

INVENTORY PINCH DECOMPOSITION
AND GLOBAL OPTIMIZATION METHODS

PLANNING AND SCHEDULING OF CONTINUOUS
PROCESSES VIA INVENTORY PINCH DECOMPOSITION
AND GLOBAL OPTIMIZATION ALGORITHMS

By PEDRO A. CASTILLO CASTILLO,
M.A.Sc. Chemical Engineering

A Thesis Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements for the Degree
Doctor of Philosophy

McMaster University

© Copyright by Pedro A. Castillo Castillo, March 2020

DOCTOR OF PHILOSOPHY (2020)

McMaster University

(Chemical Engineering)

Hamilton, Ontario

TITLE: Planning and Scheduling of Continuous Processes
Via Inventory Pinch Decomposition and Global
Optimization Algorithms

AUTHOR: Pedro A. Castillo Castillo
M.A.Sc. Chemical Engineering (McMaster
University)

SUPERVISOR: Professor Vladimir Mahalec

NUMBER OF PAGES: x, 216

Lay Abstract

Optimal planning and scheduling of production systems are two very important tasks in industrial practice. Their objective is to ensure optimal utilization of raw materials and equipment to reduce production costs. In order to compute realistic production plans and schedules, it is often necessary to replace simplified linear models with nonlinear ones including discrete decisions (e.g., “yes/no”, “on/off”). To compute a global optimal solution for this type of problems in reasonable time is a challenge due to their intrinsic nonlinear and combinatorial nature.

The main goal of this thesis is the development of efficient algorithms to solve large-scale planning and scheduling problems. The key contributions of this work are the development of: i) a heuristic technique to compute near-optimal solutions rapidly, and ii) a deterministic global optimization algorithm. Both approaches showed results and performances better or equal to those obtained by commercial software and previously published methods.

Abstract

In order to compute more realistic production plans and schedules, techniques using nonlinear programming (NLP) and mixed-integer nonlinear programming (MINLP) have gathered a lot of attention from the industry and academy. Efficient solution of these problems to a proven ε -global optimality remains a challenge due to their combinatorial, nonconvex, and large dimensionality attributes.

The key contributions of this work are: 1) the generalization of the inventory pinch decomposition method to scheduling problems, and 2) the development of a deterministic global optimization method.

An inventory pinch is a point at which the cumulative total demand touches its corresponding concave envelope. The inventory pinch points delineate time intervals where a single fixed set of operating conditions is most likely to be feasible and close to the optimum. The inventory pinch method decomposes the original problem in three different levels. The first one deals with the nonlinearities, while subsequent levels involve only linear terms by fixing part of the solution from previous levels. In this heuristic method, infeasibilities (detected via positive value of slack variables) are eliminated by adding at the first level new period boundaries at the point in time where infeasibilities are detected.

The global optimization algorithm presented in this work utilizes both piecewise McCormick (PMCR) and Normalized Multiparametric Disaggregation (NMDT), and employs a dynamic partitioning strategy to refine the estimates of the global optimum. Another key element is the parallelized bound tightening procedure.

Case studies include gasoline blend planning and scheduling, and refinery planning. Both inventory pinch method and the global optimization algorithm show promising results and their performance is either better or on par with other published techniques and commercial solvers, as exhibited in a number of test cases solved during the course of this work.

Preface

Chapters 2–8 contain multi-authored work previously published in peer-reviewed scientific journals. My individual contributions to each of those chapters consisted of the following:

- Implementing the corresponding mathematical models in GAMS.
- Developing the steps of the solution algorithms.
- Implementing the algorithms (MPIP, MPIP-C, and deterministic global optimization method) using GAMS, Python, and MATLAB.
- Running the examples and gathering numerical results.
- Analyzing the numerical results.
- Writing the initial draft and final version of each manuscript.

Contributions from Dr. Vladimir Mahalec in Chapters 2–8 included:

- Providing insightful discussions about planning and scheduling problems, potential solution strategies, and during the analysis of the numerical results.
- Approving numerical data used in the examples.
- Proofreading and editing each manuscript.

Contributions from Dr. Pedro M. Castro in Chapters 6 and 7 included:

- Providing insightful discussions about piecewise linear relaxations, bound tightening techniques, and during the analysis of the numerical results.
- Proofreading and editing each manuscript.

Acknowledgments

I would like to thank my supervisor Dr. Vladimir Mahalec for all his support, guidance, and patience during the last five years. Dr. Mahalec is a great professor and a person that really cares about his students beyond their academic performance. Since the beginning, he always encouraged me to be the best version of myself. I would like to thank him for the time and expertise he provided me, which were key elements to make each step of my journey a success. My sincere gratitude and utmost respect to him.

I would also like to thank my thesis committee: Dr. Christopher L. E. Swartz, from the Chemical Engineering department, and Dr. Antoine Deza, from the Computing and Software department. I really appreciated their advice, questions, and suggestions during our committee meetings. In addition, I would like to thank Dr. Pedro M. Castro for collaborating with me and Dr. Vladimir Mahalec during the development of our global optimization algorithm.

My sincere thanks to the always supportive and amazing administrative staff in the Chemical Engineering department: Ms. Michelle Whalen, Ms. Kristina Trollip, Ms. Lynn Falkiner, and Ms. Cathie Roberts.

For their financial support, I would like to show my gratitude to the Chemical Engineering department, the McMaster Advanced Control Consortium, the International Ontario Graduate Scholarship (OGS) Program, and the Engineering Research Council of Canada (NSERC).

Thank you to all the people that were part of my life during this time, especially to my friends from my research group, the Chemical Engineering department, the Organization of Latin American Students (OLAS), McMaster University, Hamilton and Toronto. I will never forget the time we spent together discussing optimization techniques, going to scientific conferences, playing sports all year round, going to Toronto FC matches, enjoying the good times, and supporting each other in difficult moments.

Finally, I would like to say thank you to my family, especially to my parents. They were my main motivation and their love and support were invaluable to me. Thank you to my grandparents for all their blessings. Thank you to my brothers, aunts, my uncle, and all my cousins, always putting a smile on my face when I went back to visit them and during our telephone calls.

“It is more important to ask the right questions than it is to have the right answers”

Table of Contents

Lay Abstract	iii
Abstract	iv
Preface	v
Acknowledgments	vi
Table of Contents	vii
List of Abbreviations	ix
Declaration of Academic Achievement	x
Chapter 1: Introduction	1
1.1. Supply chain optimization	2
1.2. Planning and scheduling of oil refinery operations	4
1.3. The inventory pinch approach for production planning and scheduling	8
1.4. Deterministic global optimization techniques	10
1.5. Objectives of the thesis	13
1.6. Thesis Outline	13
1.7. References	15
Chapter 2: Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part I: Gasoline Blend Planning	23
Chapter 3: Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part II: Gasoline Blend Scheduling	46
Chapter 4: Inventory Pinch-Based Multi-Scale Model for Refinery Production Planning	71
Chapter 5: Improved Continuous-Time Model for Gasoline Blend Scheduling	79
Chapter 6: Inventory Pinch Gasoline Blend Scheduling Algorithm Combining Discrete- and Continuous-Time Models	101
Chapter 7: Global Optimization Algorithm for Large-Scale Refinery Planning Models with Bilinear Terms	119
Chapter 8: Global Optimization of Nonlinear Blend-Scheduling Problems	140
Chapter 9: Global Optimization of MIQCPs with Dynamic Piecewise Relaxations	156
Chapter 10: Concluding Remarks	184

10.1.	Key Findings and Contributions	185
10.2.	Future Work Outlook	186
Appendix A: Supporting Information for Chapters 2 and 3		188
Appendix B: Supporting Information for Chapters 5, 6, and 8		192
Appendix C: Supporting Information for Chapter 7 and 9		199

List of Abbreviations

CATP	Cumulative average total production
CCR	Continuous catalytic reforming unit
CDU	Crude distillation unit
CTD	Cumulative total demand
DHT	Diesel hydrotreating unit
FBBT	Feasibility-based bound tightening
FCC	Fluid catalytic cracking unit
GAMS	General algebraic modeling system
GOHT	Gasoil hydrotreating unit
HC	Hydrocracking unit
LP	Linear programming
MILP	Mixed-integer linear programming
MINLP	Mixed-integer nonlinear programming
MPIP	Multiperiod inventory pinch
MPIP-C	Multiperiod inventory pinch with continuous-time scheduling model
NHT	Naphtha hydrotreating unit
NLP	Nonlinear programming
NMDT	Normalized multiparametric disaggregation technique
OBBT	Optimality-based bound tightening
PMCR	Piecewise McCormick relaxation
RHT	Residue hydrotreating unit

Declaration of Academic Achievement

I, Pedro Alejandro Castillo Castillo, declare that my contributions to this research work are the following:

- i) I provided the main ideas to develop the algorithms introduced in this work,
- ii) I implemented the required mathematical models in GAMS,
- iii) I implemented the proposed algorithms (MPIP, MPIP-C, and deterministic global optimization method) using Python, GAMS, and MATLAB,
- iv) I developed a Python script to use Dia Diagram Editor as a graphical user interface to model production processes as nodes in a network,
- v) I solved the case studies presented in this work and gathered the numerical results, and
- vi) I wrote the initial draft and final version of each manuscript presented here.

In addition, I declare that Dr. Vladimir Mahalec and Dr. Pedro M. Castro provided ideas and guidance to enhance such algorithms, proofread and edited the manuscripts in which each one of them collaborated.

Sincerely,

Pedro Alejandro Castillo Castillo

Chapter 1: Introduction

Planning and scheduling of production systems are two activities in supply chain optimization that increase profit margins of the plant sites by utilizing raw materials, intermediate components, storage capacity, and production equipment in the best way possible along a given time horizon, considering current market conditions and forecasts. Planning and scheduling software-based tools have become necessary for most companies, especially those that operate on economic markets with fast dynamics, face strict environmental regulations, and/or have low profit margins (e.g., commodity producers) [1].

Current trend in planning and scheduling techniques is to increase the accuracy of the mathematical models employed to represent processing units and operational policies (taking into account their scalability), as well as the development of advanced algorithms to efficiently solve these models to optimality.

It is often the case that the nature of the production process is inherently nonlinear, and operational policies usually rely on discrete decisions (e.g., “yes/no”, “on/off”). Therefore, to compute more realistic production plans and schedules, techniques using nonlinear programming (NLP) and mixed-integer nonlinear programming (MINLP) are required. The challenges associated with nonlinear planning and scheduling problems are the following:

1. Possible nonconvexities, which can introduce multiple local and global optima
 - Traditional gradient-based optimization methods can stop at a local optimum. Global optimization techniques are thus needed to understand the quality of the solution and make better decisions.
2. Potential need of a large number of partitions to represent the time domain, which can result in a model containing thousands or more variables
 - The larger the number of time periods or time slots, the larger the number of nonconvex terms and discrete variables, thus the higher computational cost involved to solve the problem to optimality.

This thesis summarizes a project focused on the development of two algorithms to solve planning and scheduling problems: a heuristic decomposition approach based on the inventory pinch concept, and a deterministic global optimization method based on dynamic partitioning of piecewise linear relaxations and optimality-based bound tightening.

In this Chapter, different concepts used throughout this report are briefly described. In addition, the objectives and outline of this thesis are presented.

1.1. Supply chain optimization

A supply chain consists of all different entities and activities necessary to produce and distribute a product to the final customer. These activities include procurement of raw materials, transformation and/or purification of the raw materials into intermediate and final products, storage and distribution of intermediate and final products, and demand forecasting and satisfaction. The physical elements of a supply chain include warehouses, distribution centers, production sites, retailers, etc. Supply chain optimization consists of determining the best possible flow of materials and information among these elements that maximize the performance of the supply chain. The performance of the supply chain is defined according to the company's goals; e.g., increase profit, market share, customer satisfaction, and/or decrease costs, lead time, etc.

Different type of decisions in the supply chain optimization problem can be identified based on business functionalities, timeframe, geographical scope, and hierarchical levels. The most common classification is shown in Figure 1. There are three basic decision levels: strategic, tactical and operational [2–6]. Long-term strategic level defines the structure and capacity of the supply chain considering a time horizon of several months or years. Medium-term tactical level assigns production and distribution targets to the different facilities usually on a weekly or monthly basis. Short-term operational level determines the assignment and sequencing of tasks to equipment units for the next few hours or days. These three levels are interconnected because the decisions made at one of them directly affect others [2, 5, 6].

In the automation pyramid (Figure 2) there are two more layers below the short-term operational level (i.e., scheduling level): real-time optimization and control. The control layer involves all the sensors, actuators, and equipment required to meet and follow process setpoints, as well as safety and alarm systems. The frequency of the calculations required by the control layer is on the order of seconds or even less. The real-time optimization (RTO) level provides setpoints to the control layer every few hours. The RTO setpoints correspond to a steady-state of the process that is optimal for the current production targets and/or market conditions.

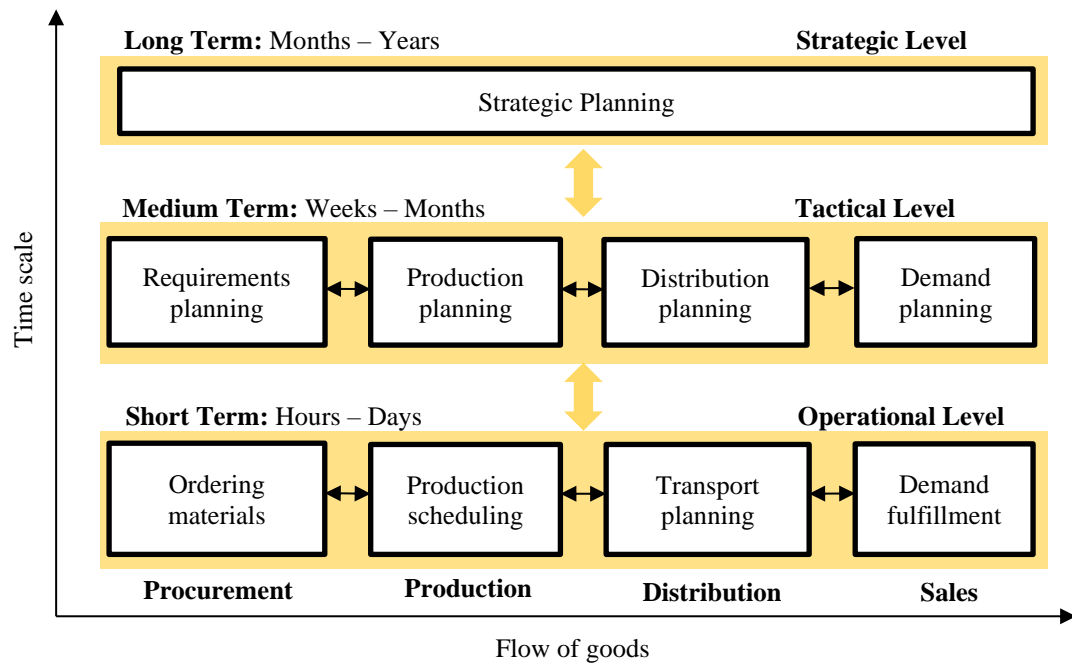


Figure 1. Supply chain planning tasks classified based on business functionalities and time scope

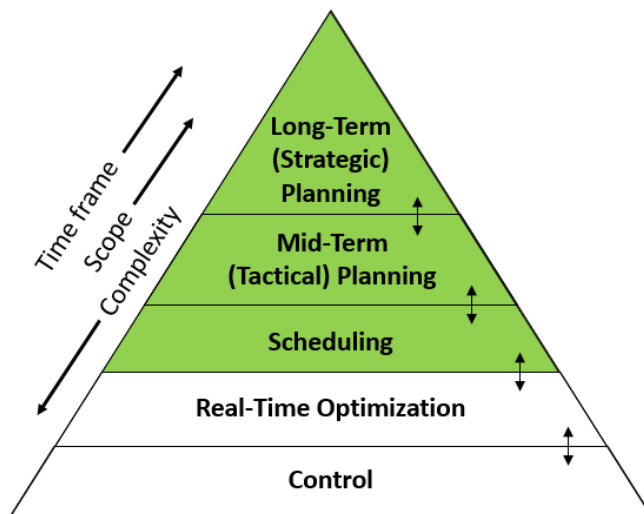


Figure 2. Automation pyramid

Computational tools based on mathematical programming and simulation techniques have become very common in modern industry for supply chain optimization. Mathematical models derived from engineering first principles (i.e., material and energy balances, thermodynamic relationships, reaction kinetics, etc.) or from historical plant data (i.e., data-driven models) are used to represent supply chain elements. These models also include operational constraints such as maximum and minimum production, storage, and transportation capacities, product demand, product specifications, availability of raw materials, inventory policies, etc. A model must be robust, reliable, and relatively easy to maintain. Model formulation is key to be able to compute realistic and optimal solutions (i.e., plans and schedules) in a reasonable amount of time (depending on the application).

Given the complexity of modeling an entire supply chain, as well as the high computational cost required to solve such model to optimality, supply chain optimization is usually carried out by solving smaller optimization problems. It is very common to use the scheme shown in Figure 1 (plus geographical scope) to define these smaller problems.

For production planning and scheduling problems, formulations can be classified based on the process type (continuous, batch) and the time representation employed (discrete, continuous, and their variants). Models can be classified as well according to their mathematical structure (linear, nonlinear, mixed-integer, etc.). Extensive reviews can be found in the literature [7–9]. Another key aspect is the algorithm used to solve the optimization problem. The solution algorithms can be classified as deterministic, stochastic, and heuristic methods. Based on their optimality guarantees, they are classified into local and global optimization methods.

Research efforts have been directed to integrate several decision levels. By taking into account the interactions between them, the efficiency of the supply chain can be increased. Model formulations and solution algorithms that exploit the structure of the integrated problems have been developed in the last decades [10–13], but there is still an ongoing research work in this area.

In section 1.2, an overview of advances and challenges in planning and scheduling of oil refinery operations is presented.

1.2. Planning and scheduling of oil refinery operations

Crude oil is a mixture of different hydrocarbons and, to a lesser extent, other organic and inorganic compounds. Most common types of hydrocarbons found in crude oil are alkanes, naphthenes, and aromatics. Crude oils from different reservoirs have different

attributes (i.e., quality properties or qualities), e.g., density, aromatics, sulfur, and metals content, etc. Oil refineries transform crude oil into more valuable products such as liquefied petroleum gas, gasoline, diesel, jet fuel, and other hydrocarbon products which can be used as either fuels or feedstocks for other chemical processes. The petroleum refining industry is still the largest source of energy products in the world [14].

A petroleum refinery plant is commonly divided into three main sections: crude oil unloading and blending, production units, and blending and shipping of final products [15, 16]. The crude oil is transported to the plant by tankers or through pipelines, where it is unloaded into storage tanks. From these storage tanks, crude oils are then transferred into charging tanks where they are mixed. The crude oil mix is fed to the crude distillation units (CDUs) where the crude mix is separated into different fractions based on their boiling temperature range. The crude oil fractions go through a hydrodesulfurization process to remove most of their sulfur content (because sulfur can poison the catalysts of downstream units). Subsequently, the crude oil fractions go through corresponding chemical processes: i) Catalytic reforming converts low-octane naphthas into high-octane reformat, ii) hydrocracking employs hydrogen to break long-chain hydrocarbons into simpler compounds (mostly diesel and jet fuel), and iii) fluid catalytic cracking transforms heavy crude oil fractions into higher value products (mostly gasoline and light olefins). Finally, the intermediate products are blended into final products, which are shipped through pipelines or distributed by tanker trucks. The final products must meet associated minimum and maximum quality specifications. Figure 3 shows a simplified scheme of an oil refinery plant with one CDU, one continuous catalytic reformer (CCR), one hydrocracker (HC), one fluid catalytic cracker (FCC), four different hydrotreaters (NHT, DHT, GOHT, RHT), and the gasoline and diesel blending sections. Given the complexity of the processes involved and their interconnections, a lot of work in the literature has been dedicated to oil refinery planning and scheduling problems.

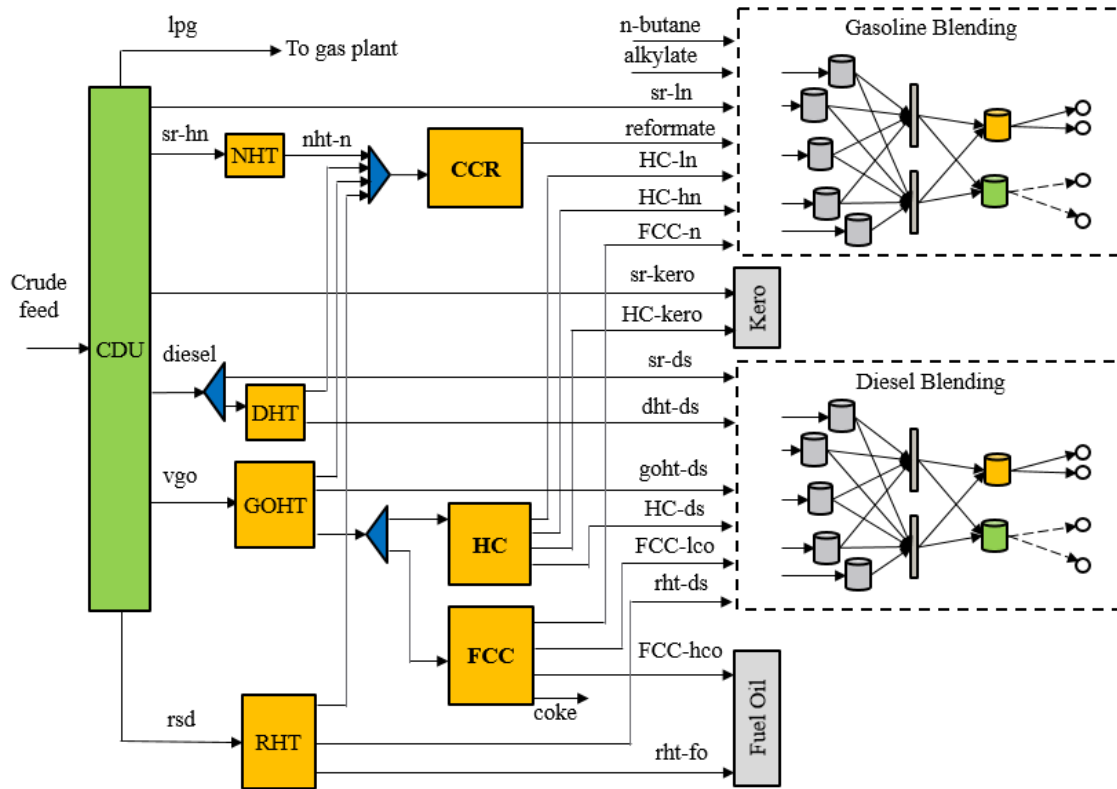


Figure 3. Simplified scheme of an oil refinery plant

Production planning in petroleum industries started to use linear programming in the 1950s [17]. Nonlinear models have gathered more attention since the late 1990s because of the technological advances in nonlinear optimization solvers. The general modelling framework for a processing unit in a refinery [18] considers i) the flowrate of each product stream as a function of the feed flowrate, the feed properties, and unit operating conditions, and ii) the properties of each product stream as a function of the feed properties, and unit operating conditions. Particular frameworks for storage tanks, blenders, and pipelines in a refinery system have been developed too [19, 20]. Discrete-time formulations are usually employed for planning models [20–24]. The time periods in which the planning horizon is discretized are denoted as big-bucket periods [2, 14] because the goal of planning models is to provide production and inventory targets for each time period, not to exactly define the start and end times of all the tasks involved to meet those targets. Mathematical models based on engineering first principles and/or empirical correlations, as well as artificial neural networks, have been developed for crude distillation units [25–28], hydrocracking units [29], and fluid catalytic cracking

units [30–32]. Currently, there exist a renewed interest in data-driven models due to the improvements in big-data applications [21, 33, 34].

Current research trend is to formulate planning models that consider more upstream and downstream operations in the supply chain (i.e., enterprise-wide optimization) [14, 35, 36], integrate more scheduling decisions [2, 10, 12, 13, 37, 38], and that take into account the uncertainty in demand, supply, and price forecasts [39–42], while keeping the model computationally tractable or developing efficient solution algorithms tailored to model formulations. More recently, pinch analysis for production planning has been developed [43–45]. This topic is described in section 1.3.

Production scheduling in oil refineries is usually carried out by scheduling the three refinery sections separately [15, 46–49], but solution strategies that account for their interdependence have recently been published [37, 50–52]. Compared to planning models, scheduling models include more constraints associated with operational policies and logistics. These constraints often involve discrete decisions (e.g., yes-no, on-off); therefore, most refinery scheduling formulations are mixed-integer linear models. Solution strategies for this type of models rely on the branch-and-bound methodology. Scheduling decisions are the following: i) To specify the number of tasks required to meet production and inventory targets, ii) to associate those tasks to specific units, iii) to select the appropriate operating modes of the units, and iv) to determine the sequence of these tasks that incurs in the less number of product changeovers in the tanks with low or null demurrages (see Figure 4). Discrete-time and continuous-time models have been developed for refinery scheduling problems [18, 53–55].

Current research trend is to develop scheduling formulations with reduced number of discrete variables [56, 57], that provide a tight relaxation [58], and that take into consideration mode transitions in the processing units [53]. By formulating scheduling models of tractable size with strong relaxations, the solution of the refinery-wide scheduling problem can be simplified and longer scheduling horizons can be considered. Also, integration of planning and scheduling decisions is an ongoing research topic.

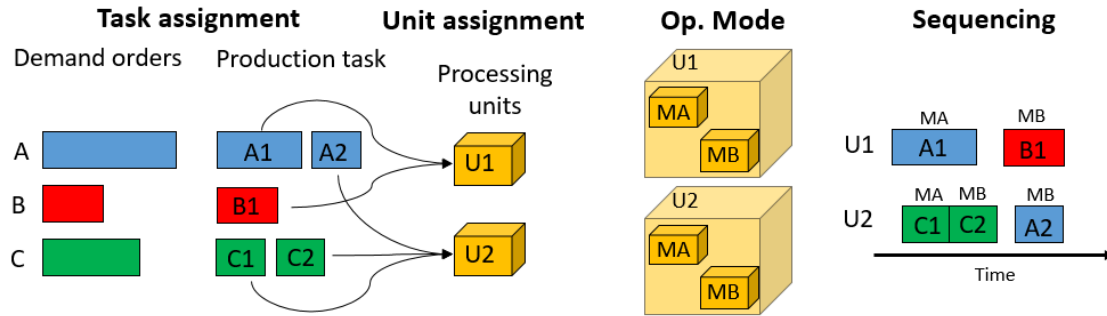


Figure 4. Scheduling decisions: task assignment, unit assignment, selection of operating mode, and task sequencing

1.3. The inventory pinch approach for production planning and scheduling

Pinch analysis was first introduced by Bodo Linhoff during the late 1970's to calculate the minimum amount of heat and cold utilities required in a heat exchanger network [59, 60]. The concept was quickly adapted to the general case of energy consumption minimization and it constitutes one of the first process integration techniques [61, 62]. The general idea is to determine the hot and cold composite curves based on the energy available at the different temperatures present in the process network, and then identify the point at which the two curves are separated by the minimum temperature difference allowed (ΔT^{min}). The reason why the two curves should not touch is because as ΔT^{min} tends to zero, the heat exchanger area required increases to infinity. Once the two curves are separated by ΔT^{min} , the minimum external hot and cold utility requirements (or energy targets) can be easily determined (see Figure 5). To achieve these targets, three rules must be followed: i) heat must not be transferred across the pinch, ii) there must be no external cooling above the pinch, and iii) there must be no external heating below the pinch.

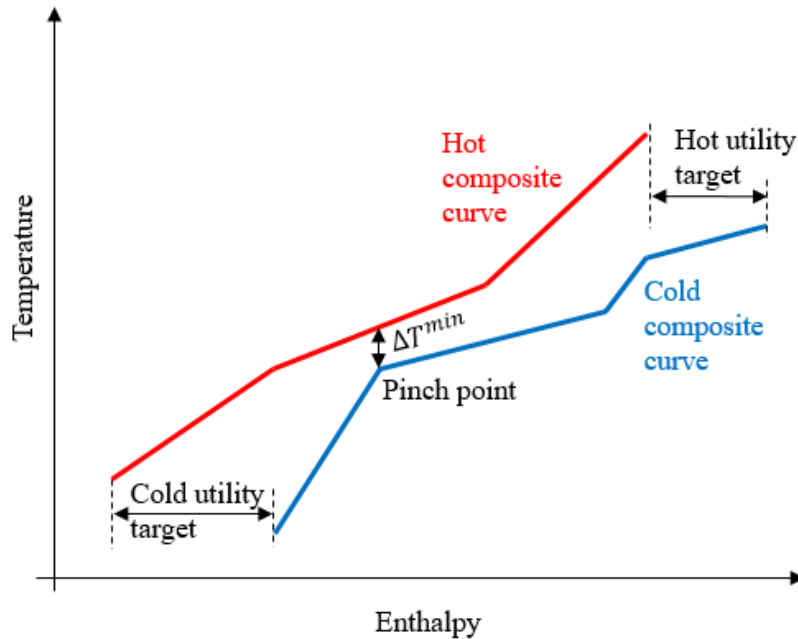


Figure 5. Pinch point in energy consumption minimization

Pinch analysis techniques have been developed for a wide range of applications: water network synthesis [63–65], carbon-constrained energy sector planning [66], and financial management [67]. Pinch analysis has been used in production planning too. Singhvi and Shenoy [44, 43] used the demand and production composite curves to define how much product is necessary to be produced between pinch points. In this case, pinch points are defined as the points where the two composite curves touch (i.e., there is no minimum separation equivalent to ΔT^{min}).

Castillo et al. [45] developed a different approach to use pinch analysis in production planning. Castillo et al. [45] defined an inventory pinch point as the point where the cumulative total demand (CTD) curve and the cumulative average total production (CATP) curve touch (see Figure 6). The CTD curve is constructed based on the demand data. The CATP curve is defined by the minimum number of straight-line segments whose initial and last points touch the CTD curve; except for the first segment, which starts at the initial total inventory available at the beginning of the planning horizon. The inventory pinch points delineate time periods where constant operating conditions are likely to be feasible [45].

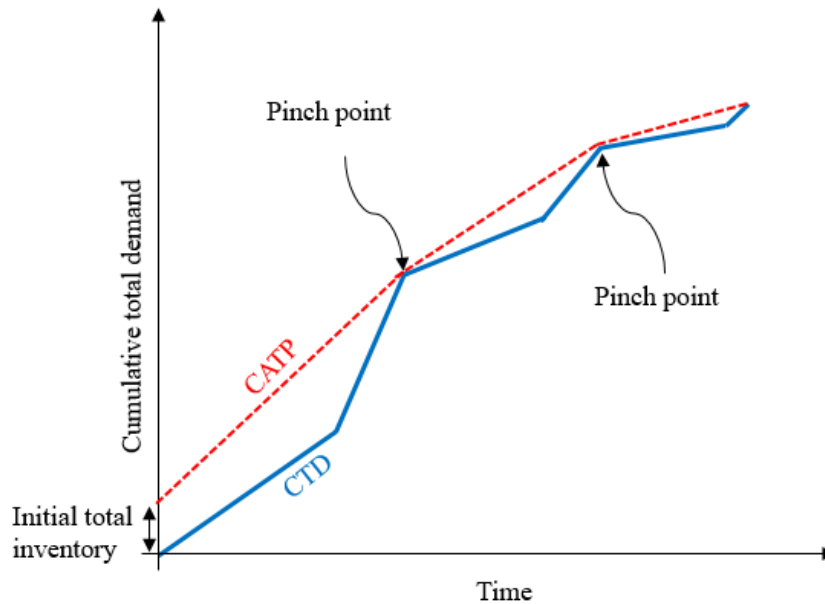


Figure 6. CTD and CATP curves, and inventory pinch points

Castillo et al. [45] developed an iterative approach:

1. To optimize operating conditions for pinch-delineated periods, and
2. To eliminate infeasibilities if they are encountered.

The inventory pinch approach is very useful when the number of pinch-delineated periods is smaller than the original time discretization of the planning problem. This dimensionality reduction makes the problem formulation smaller, thus requiring less computational effort to solve it to optimality. It also produces optimal or near-optimal solutions with operating conditions that remain constant as much as possible, which is something desirable from an operational point of view. Chapters 2, 3, and 5 contain more details on this methodology.

The inventory pinch approach is a heuristic technique which does not guarantee globally optimal solutions. In section 1.4, a brief review of rigorous global optimization methods is presented.

1.4. Deterministic global optimization techniques

Deterministic global optimization focuses on developing and improving mathematical theories, algorithms, and computational tools in order to find a global minimum of the

objective function f subject to the set of constraints S by computing lower and upper bounds of the objective function f that are valid for the whole feasible region S . The goal of deterministic global optimization is to compute an ε -global optimal solution with theoretical guarantees, where $\varepsilon > 0$ refers to the desired relative difference between the upper and lower bounds.

Consider a minimization problem. To compute lower bounds, deterministic global optimization algorithms relax the original nonconvex nonlinear problem into either a linear (LP), a mixed-integer linear (MILP), or a convex nonlinear program (NLP). The relaxation can be derived using one or a combination of the following methodologies: convex envelopes [68–70], piecewise linear relaxations [71–73], α BB underestimators [74, 75], the reformulation-linearization technique [76], outer-approximation [77, 78], by removing integrality constraints, and other techniques. To iteratively improve the relaxation (i.e., make it tighter or closer to the original model), one can rely on spatial branch-and-bound [71] (see Figure 7), cutting planes [79], bound tightening [80, 81], interval elimination strategies [82], and further partitioning in piecewise relaxations [83]. To compute upper bounds (i.e., feasible solutions), information from the relaxation is often used by single/multistart NLP strategies and other heuristic techniques.

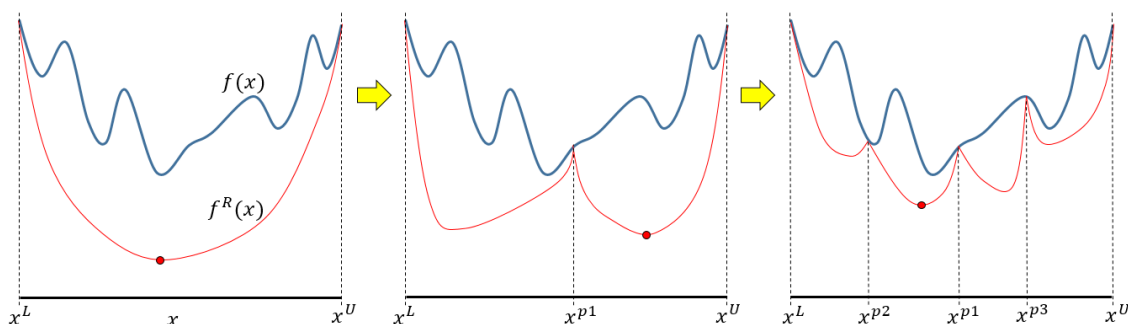


Figure 7. Sketch of a nonconvex function $f(x)$ (blue curve) and some possible relaxations $f^R(x)$ (red curves). By partitioning the domain of variable x , the relaxations become closer to the original function, and the best possible solution (red dot) increases.

Bound tightening (or range reduction) techniques reduce the domain of the variables involved in nonlinear terms. There are two main categories: Feasibility-based bound tightening (FBBT), and optimality-based bound tightening (OBBT). FBBT is an iterative procedure that employs the model constraints and interval arithmetic to imply bounds on

the variables [84]. Although FBBT is not the most effective method to reduce the bounds of the variables, it does not require too much computational effort and it is very common in most global optimization algorithms. On the other hand, OBBT involves solving one minimization and one maximization problem for each variable [80]. The minimization problem yields a lower bound of the variable, and the maximization problem gives an upper bound. These optimization problems can be solved sequentially [85] or in parallel [86].

In a branch-and-bound algorithm, it has been shown that is useful to apply OBBT at each node instead of only at the root node, in order to reduce the number of nodes to explore and the final optimality gap [87]. Since OBBT is very effective but requires significant computational effort, accelerating and approximation techniques have been proposed for OBBT in a branch-and-bound framework [88].

A different strategy is to not use a branch-and-bound framework at all. In this case, piecewise linear relaxations are employed and the number of partitions is increased in each iteration [83, 86]. By increasing the number of partitions, the relaxation becomes tighter. However, increasing the number of partitions results in larger MILP models and the difficulty to solve them to optimality (due to the addition of extra binary variables). In order to tighten the relaxation and avoid a rapid increase in model size, OBBT can be applied before increasing the number of partitions. By reducing the domain of the variables, the same number of partitions will yield a tighter relaxation. Given the large number of variables involved in bilinear terms (and that each variable requires two optimization problems), parallel implementation of OBBT is necessary to develop efficient algorithms.

Global commercial solvers employ a variety of all the previous discussed techniques and methodologies. BARON [89] relies heavily on spatial branch-and-bound and linear relaxations, but newer versions are moving towards a more significant use of piecewise linear relaxations. ANTIGONE [90] relies more on OBBT, cutting planes, and piecewise linear relaxations. Currently, there is no commercial solver that will outperform the others if using a wide variety of test examples for comparison. In general, for bilinear programs, most of the research on global optimization has been done on formulating tighter MINLP model formulations, improving piecewise relaxation techniques, and novel algorithmic developments. Applications of global optimization methods to refinery planning are described in Chapters 6 and 7.

1.5. Objectives of the thesis

The focus of this thesis is the development of efficient algorithms to solve planning and scheduling problems that can be formulated as mixed-integer nonlinear programs, with nonlinearities strictly due to bilinear and/or quadratic terms. More specifically:

1. The generalization of the inventory pinch decomposition method to scheduling problems, and
2. The development of a deterministic global optimization method based on dynamic partitioning of piecewise linear relaxations and optimality-based bound tightening.

Thus, this thesis work explores both heuristic and rigorous optimization approaches, their particular advantages and disadvantages, and how can they complement each other.

1.6. Thesis Outline

Chapter 1: Introduction. This chapter summarizes the literature review and the fundamental principles related to this project. It also includes the research objectives and the thesis outline.

Chapter 2: “Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part I: Gasoline Blend Planning”. This chapter presents more details about the inventory pinch concept for production planning, and it describes the multiperiod inventory pinch (MPIP) algorithm for blend planning problems. MPIP is a heuristic technique that decomposes the planning problem into two levels. The 1st level optimizes blend recipes, and the 2nd level computes blend plan. Both levels are formulated using discrete-time representation. This work has been published in the *AIChE Journal*.

Chapter 3: “Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part II: Gasoline Blend Scheduling”. This chapter describes the MPIP algorithm for blend scheduling problems. For this type of problems, MPIP employs a three level decomposition. The 1st and 2nd levels are constructed as in Chapter 2, while the 3rd level is a multiperiod MILP model with fixed blend recipes. All three levels are formulated using discrete-time representation. This work has been published in the *AIChE Journal*.

Chapter 4: “Inventory Pinch-Based Multi-Scale Model for Refinery Production Planning”. In this chapter, the MPIP algorithm from Chapter 2 is applied to a refinery

planning problem. In this example, the inventory pinch points are defined for each blending pool, e.g., gasoline and diesel.

Chapter 5: “Improved Continuous-Time Model for Gasoline Blend Scheduling”. This chapter presents a continuous-time blend scheduling model that includes more operational constraints than previously published model, but it requires smaller number of binary variables. This work has been published in the *Computers & Chemical Journal*.

Chapter 6: “Inventory Pinch Gasoline Blend Scheduling Algorithm Combining Discrete- and Continuous-Time Models”. This chapter introduces the MPIP-C algorithm which is an improved version of the MPIP method. By employing the continuous-time blend scheduling model from Chapter 5, MPIP-C requires smaller execution times than MPIP and computes better solutions (less switching operations). This work has been published in the *Computers & Chemical Journal*.

Chapter 7: “Global Optimization Algorithm for Large-Scale Refinery Planning Models with Bilinear Terms”. This chapter describes the deterministic global optimization algorithm designed for mixed-integer bilinear programs. This algorithm computes estimates of the global solution by solving MILP relaxations of the original model derived using either Piecewise McCormick or Normalized Multiparametric Disaggregation. The estimates of the global solution are refined by increasing the number of partitions and reducing the domain of the variables involved in bilinear terms. This work has been published in the *Industrial & Engineering Chemistry Research Journal*.

Chapter 8: “Global Optimization of Nonlinear Blend-Scheduling Problems”. This chapter presents the results obtained for nonlinear blend-scheduling problems using both MPIP-C and the global optimization algorithm from Chapter 7. This work has been published in the *Engineering Journal*.

Chapter 9: “Global Optimization of MIQCPs with Dynamic Piecewise Relaxations”. This chapter describes an enhanced version of the algorithm presented in Chapter 7. This global optimization algorithm aims to reduce as much as possible the domain of the variables involved in bilinear terms by using optimality-based bound tightening more extensively. The algorithm also increases or decreases the number of partitions depending on the last iteration execution time, optimality gap improvement, and average domain reduction. The algorithm switches from piecewise McCormick to Normalized Multiparametric Disaggregation when the number of partitions is greater or equal to 10. This work has been published in the *Journal of Global Optimization*.

Chapter 10: Concluding Remarks. The final chapter explores main conclusions, major contributions and future work for this research project.

Appendix A, B, and C: Supporting information for Chapters 2 to 9.

1.7. References

1. Papageorgiou, L.G.: Supply chain optimisation for the process industries: Advances and opportunities. *Comput. Chem. Eng.* (2009). doi:10.1016/j.compchemeng.2009.06.014
2. Maravelias, C.T., Sung, C.: Integration of production planning and scheduling: Overview, challenges and opportunities. *Comput. Chem. Eng.* 33, 1919–1930 (2009). doi:10.1016/j.compchemeng.2009.06.007
3. Fleischmann, B., Meyr, H.: Supply Chain Operations Planning Hierarchy, Modeling and Advanced Planning Systems.
4. Fleischmann, B., Meyr, H., Wagner, M.: Advanced Planning. In: Stadtler, H., Kilger, C., and Meyr, H. (eds.) *Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies*. pp. 71–95. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
5. Shobrys, D.E., White, D.C.: Planning, scheduling and control systems: why can they not work together. *Comput. Chem. Eng.* 24, 163–173 (2000)
6. Noz, E.M., Capón-García, E., Laínez-Aguirre, J.M., Esp Na, A., Puigjaner, L.: Supply chain planning and scheduling integration using Lagrangian decomposition in a knowledge management environment. *Comput. Chem. Eng.* 72, 52–67 (2015). doi:10.1016/j.compchemeng.2014.06.002
7. Floudas, C.A., Lin, X.: Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput. Chem. Eng.* 28, 2109–2129 (2004)
8. Sundaramoorthy, A., Maravelias, C.T.: Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind. Eng. Chem. Res.* 50, 5023–5040 (2011)
9. Harjunkski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Méndez, C., Sand, G., Wassick, J.: Scope for industrial applications of production scheduling models and solution methods. *Comput. Chem. Eng.* 62, 161–193 (2014)
10. Li, Z., Ierapetritou, M.G.: Production planning and scheduling integration through augmented Lagrangian optimization. *Comput. Chem. Eng.* (2010). doi:10.1016/j.compchemeng.2009.11.016

11. Li, Z., Ierapetritou, M.G.: Integrated production planning and scheduling using a decomposition framework. *Chem. Eng. Sci.* (2009). doi:10.1016/j.ces.2009.04.047
12. Terrazas-Moreno, S., Trotter, P.A., Grossmann, I.E.: Temporal and spatial Lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers. *Comput. Chem. Eng.* (2011). doi:10.1016/j.compchemeng.2011.01.004
13. Terrazas-Moreno, S., Grossmann, I.E.: A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chem. Eng. Sci.* (2011). doi:10.1016/j.ces.2011.03.017
14. Shah, N.K., Li, Z., Ierapetritou, M.G.: Petroleum refining operations: Key issues, advances, and opportunities. *Ind. Eng. Chem. Res.* (2011). doi:10.1021/ie1010004
15. Jia, Z., Ierapetritou, M.: Efficient short-term scheduling of refinery operations based on a continuous time formulation. In: *Computers and Chemical Engineering* (2004)
16. Méndez, C.A., Grossmann, I.E., Harjunkoski, I., Kaboré, P.: A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. *Comput. Chem. Eng.* 30, 614–634 (2006)
17. Bodington, C.E., Baker, T.E.: A History of Mathematical Programming in the Petroleum Industry. *Interfaces* (Providence). 20, 117–127 (1990). doi:10.1287/inte.20.4.117
18. Pinto, J.M., Joly, M., Moro, L.F.L.: Planning and scheduling models for refinery operations. *Comput. Chem. Eng.* (2000). doi:10.1016/S0098-1354(00)00571-8
19. Neiro, S.M.S., Pinto, J.M.: A general modeling framework for the operational planning of petroleum supply chains. In: *Computers and Chemical Engineering* (2004)
20. Neiro, S.M.S., Pinto, J.M.: Multiperiod optimization for production planning of petroleum refineries. *Chem. Eng. Commun.* (2005). doi:10.1080/00986440590473155
21. Alhajri, I., Elkamel, A., Albahri, T., Douglas, P.L.: A nonlinear programming model for refinery planning and optimisation with rigorous process models and product quality specifications. *Int. J. Oil, Gas Coal Technol. J. Oil, Gas Coal Technol.* 1, 283–307 (2008)
22. Pongsakdi, A., Rangsunvigit, P., Siemanond, K., Bagajewicz, M.J.: Financial risk management in the planning of refinery operations. *Int. J. Prod. Econ.* (2006). doi:10.1016/j.ijpe.2005.04.007

23. Leiras, A., Hamacher, S., Elkamel, A.: Petroleum refinery operational planning using robust optimization. *Eng. Optim.* (2010). doi:10.1080/03052151003686724
24. Zhen, G., Lixin, T., Hui, J., Nannan, X.: An Optimization Model for the Production Planning of Overall Refinery*. *Chinese J. Chem. Eng.* 16, 67–70 (2008)
25. Menezes, B.C., Kelly, J.D., Grossmann, I.E.: Improved swing-cut modeling for planning and scheduling of oil-refinery distillation units. *Ind. Eng. Chem. Res.* (2013). doi:10.1021/ie4025775
26. Kumar, V., Sharma, A., Chowdhury, I.R., Ganguly, S., Saraf, D.N.: A crude distillation unit model suitable for online applications. *Fuel Process. Technol.* (2001). doi:10.1016/S0378-3820(01)00195-3
27. Dave, D.J., Dabhiya, M.Z., Satyadev, S.V.K., Ganguly, S., Saraf, D.N.: Online tuning of a steady state crude distillation unit model for real time applications. In: *Journal of Process Control* (2003)
28. Alattas, A.M., Grossmann, I.E., Palou-Rivera, I.: Integration of nonlinear crude distillation unit models in refinery planning optimization. *Ind. Eng. Chem. Res.* (2011). doi:10.1021/ie200151e
29. Alhajree, I., Zahedi, G., Manan, Z.A., Zadeh, S.M.: Modeling and optimization of an industrial hydrocracker plant. *J. Pet. Sci. Eng.* (2011). doi:10.1016/j.petrol.2011.07.019
30. Li, W., Hui, C.W., Li, A.: Integrating CDU, FCC and product blending models into refinery planning. *Comput. Chem. Eng.* (2005). doi:10.1016/j.compchemeng.2005.05.010
31. Kangas, I., Nikolopoulou, C., Attiya, M.: Modeling & Optimization of the FCC Unit to Maximize Gasoline Production and Reduce Carbon Dioxide Emissions in the Presence of CO₂ Emissions Trading Scheme. (2013)
32. Long, J., Mao, M.S., Zhao, G.Y.: Refinery Planning Optimization Integrating Rigorous Fluidized Catalytic Cracking Unit Models. *Pet. Sci. Technol.* (2015). doi:10.1080/10916466.2015.1076846
33. Li, J., Boukouvala, F., Xiao, X., Floudas, C.A., Zhao, B., Du, G., Su, X., Liu, H.: Data-Driven Mathematical Modeling and Global Optimization Framework for Entire Petrochemical Planning Operations. *AIChE J.* (2016)
34. Cuiwen, C., Xingsheng, G., Zhong, X.: A data-driven rolling-horizon online scheduling model for diesel production of a real-world refinery. *AIChE J.* (2013). doi:10.1002/aic.13895
35. Grossmann, I.E.: Advances in mathematical programming models for enterprise-

- wide optimization. *Comput. Chem. Eng.* (2012).
doi:10.1016/j.compchemeng.2012.06.038
36. Grossmann, I.E.: Challenges in the application of mathematical programming in the enterprise-wide optimization of process industries. *Theor. Found. Chem. Eng.* (2014). doi:10.1134/S0040579514050182
37. Mouret, S., Grossmann, I.E., Pestiaux, P.: A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Comput. Chem. Eng.* (2011). doi:10.1016/j.compchemeng.2011.03.026
38. Shah, N.K., Ierapetritou, M.G.: Integrated production planning and scheduling optimization of multisite, multiproduct process industry. *Comput. Chem. Eng.* (2012). doi:10.1016/j.compchemeng.2011.08.007
39. Li, W., Hui, C.-W., Li, P., Li, A.-X.: Refinery Planning under Uncertainty. doi:10.1021/ie049737d
40. Yang, Y., Barton, P.I.: Integrated crude selection and refinery optimization under uncertainty. *AIChE J.* (2016). doi:10.1002/aic.15075
41. Al-Othman, W.B.E., Lababidi, H.M.S., Alatiqi, I.M., Al-Shayji, K.: Supply chain optimization of petroleum organization under uncertainty in market demands and prices. *Eur. J. Oper. Res.* (2008). doi:10.1016/j.ejor.2006.06.081
42. LUO, C., RONG, G.: A Strategy for the Integration of Production Planning and Scheduling in Refineries under Uncertainty. *Chinese J. Chem. Eng.* (2009). doi:10.1016/S1004-9541(09)60042-2
43. Singhvi, A., Madhavan, K.P., Shenoy, U. V: Pinch analysis for aggregate production planning in supply chains. *Comput. Chem. Eng.* 28, 993–999 (2004). doi:10.1016/j.compchemeng.2003.09.006
44. Singhvi, A., Shenoy, U.V.: Aggregate Planning in Supply Chains by Pinch Analysis. *Chem. Eng. Res. Des.* (2002). doi:10.1205/026387602760312791
45. Castillo, P.A.C., Mahalec, V., Kelly, J.D.: Inventory pinch algorithm for gasoline blend planning. *AIChE J.* 59, 3748–3766 (2013)
46. Wu, N.Q., Bai, L.P., Zhou, M.C.: An Efficient Scheduling Method for Crude Oil Operations in Refinery with Crude Oil Type Mixing Requirements. *IEEE Trans. Syst. Man, Cybern. Syst.* (2016). doi:10.1109/TSMC.2014.2332138
47. Reddy, P.C.P., Karimi, I.A., Srinivasan, R.: A new continuous-time formulation for scheduling crude oil operations. *Chem. Eng. Sci.* (2004). doi:10.1016/j.ces.2004.01.009
48. Li, J., Xiao, X., Floudas, C.A.: Integrated gasoline blending and order delivery

- operations: Part I. Short-term scheduling and global optimization for single and multi-period operations. *AICHE J.* (2016)
49. Li, J., Karimi, I.A., Srinivasan, R.: Recipe determination and scheduling of gasoline blending operations. *AICHE J.* (2010). doi:10.1002/aic.11970
 50. Luo, C., Rong, G.: Hierarchical approach for short-term scheduling in refineries. *Ind. Eng. Chem. Res.* (2007). doi:10.1021/ie061354n
 51. Shah, N.K., Sahay, N., Ierapetritou, M.G.: Efficient Decomposition Approach for Large-Scale Refinery Scheduling. *Ind. Eng. Chem. Res.* (2015). doi:10.1021/ie504835b
 52. Shah, N.K., Ierapetritou, M.G.: Lagrangian decomposition approach to scheduling large-scale refinery operations. *Comput. Chem. Eng.* (2015). doi:10.1016/j.compchemeng.2015.04.021
 53. Shi, L., Jiang, Y., Wang, L., Huang, D.: Refinery production scheduling involving operational transitions of mode switching under predictive control system. *Ind. Eng. Chem. Res.* (2014). doi:10.1021/ie500233k
 54. Karuppiyah, R., Furman, K.C., Grossmann, I.E.: Global optimization for scheduling refinery crude oil operations. *Comput. Chem. Eng.* (2008). doi:10.1016/j.compchemeng.2007.11.008
 55. Gao, X., Jiang, Y., Chen, T., Huang, D.: Optimizing scheduling of refinery operations based on piecewise linear models. *Comput. Chem. Eng.* (2015). doi:10.1016/j.compchemeng.2015.01.022
 56. Kolodziej, S.P., Grossmann, I.E., Furman, K.C., Sawaya, N.W.: A discretization-based approach for the optimization of the multiperiod blend scheduling problem. *Comput. Chem. Eng.* (2013). doi:10.1016/j.compchemeng.2013.01.016
 57. Mouret, S., Grossmann, I.E., Pestiaux, P.: A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. *Ind. Eng. Chem. Res.* (2009). doi:10.1021/ie8019592
 58. Castro, P.M.: New MINLP formulation for the multiperiod pooling problem. *AICHE J.* (2015). doi:10.1002/aic.15018
 59. Linnhoff, B., Flower, J.R.: Synthesis of heat exchanger networks: I. Systematic generation of energy optimal networks. *AICHE J.* 24, 633–642 (1978). doi:10.1002/aic.690240411
 60. Linnhoff, B., Hindmarsh, E.: The pinch design method for heat exchanger networks. *Chem. Eng. Sci.* 38, 745–763 (1983). doi:https://doi.org/10.1016/0009-2509(83)80185-7

61. Smith, R.: State of the art in process integration. *Appl. Therm. Eng.* 20, 1337–1345 (2000). doi:[https://doi.org/10.1016/S1359-4311\(00\)00010-7](https://doi.org/10.1016/S1359-4311(00)00010-7)
62. Klemeš, J.J., Kravanja, Z.: Forty years of Heat Integration: Pinch Analysis (PA) and Mathematical Programming (MP). *Curr. Opin. Chem. Eng.* 2, 461–474 (2013). doi:<https://doi.org/10.1016/j.coche.2013.10.003>
63. Foo, D.C.Y.: State-of-the-Art Review of Pinch Analysis Techniques for Water Network Synthesis. *Ind. Eng. Chem. Res.* 48, 5125–5159 (2009). doi:10.1021/ie801264c
64. Tan, Y.L., Manan, Z.A., Foo, D.C.Y.: Retrofit of Water Network with Regeneration Using Water Pinch Analysis. *Process Saf. Environ. Prot.* 85, 305–317 (2007). doi:<https://doi.org/10.1205/psep06040>
65. Jacob, J., Kaibe, H., Couderc, F., Paris, J.: Water network analysis in pulp and paper processes by pinch and linear programming techniques. *Chem. Eng. Commun.* 189, 184–206 (2002). doi:10.1080/00986440211836
66. Tan, R.R., Foo, D.C.Y.: Pinch analysis approach to carbon-constrained energy sector planning. *Energy.* 32, 1422–1429 (2007). doi:<https://doi.org/10.1016/j.energy.2006.09.018>
67. Zhelev, T.K.: On the integrated management of industrial resources incorporating finances. *J. Clean. Prod.* 13, 469–474 (2005). doi:<https://doi.org/10.1016/j.jclepro.2003.09.004>
68. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Math. Program.* 10, 147–175 (1976)
69. Liberti, L., Pantelides, C.C.: Convex Envelopes of Monomials of Odd Degree. *J. Glob. Optim.* 25, 157–168 (2003). doi:10.1023/A:1021924706467
70. Meyer, C.A., Floudas, C.A.: Convex envelopes for edge-concave functions. *Math. Program.* 103, 207–224 (2005). doi:10.1007/s10107-005-0580-9
71. Quesada, I., Grossmann, I.E.: Global optimization of bilinear process networks with multicomponent flows. *Comput. Chem. Eng.* (1995). doi:10.1016/0098-1354(94)00123-5
72. Misener, R., Thompson, J.P., Floudas, C.A.: APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Comput. Chem. Eng.* 35, 876–892 (2011)
73. Teles, J.P., Castro, P.M., Matos, H.A.: Multi-parametric disaggregation technique for global optimization of polynomial programming problems. *J. Glob. Optim.*

- (2013). doi:10.1007/s10898-011-9809-8
74. Androulakis, I.P., Maranas, C.D., Floudas, C.A.: c BB: A Global Optimization Method for General Constrained Nonconvex Problems. *J. Glob. Optim.* 7, 337–363 (1995)
 75. Adjiman, C.S., Androulakis, I.P., Maranas, C.D., Floudas, C.A.: A GLOBAL OPTIMIZATION METHODs aBB, FOR PROCESS DESIGN. *Comput. chem. Engng.* 20, 419–424 (1996)
 76. Sherali, H.D., Alameddine, A.: A new reformulation-linearization technique for bilinear programming problems. *J. Glob. Optim.* 2, 379–410 (1992)
 77. Duran, M.A., Grossmann, I.E.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Program.* 36, 307–339 (1986)
 78. Bergamini, M.L., Grossmann, I., Scenna, N., Aguirre, P.: An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Comput. Chem. Eng.* (2008). doi:10.1016/j.compchemeng.2007.03.011
 79. Karuppiyah, R., Furman, K.C., Grossmann, I.E.: Global Optimization for Scheduling Refinery Crude Oil Operations. (2007)
 80. Castro, P.M., Grossmann, I.E.: Optimality-based bound contraction with multiparametric disaggregation for the global optimization of mixed-integer bilinear problems. *J. Glob. Optim.* (2014). doi:10.1007/s10898-014-0162-6
 81. Yang, Y., Barton, P.I.: Refinery optimization integrated with a nonlinear crude distillation unit model. In: *IFAC-PapersOnLine* (2015)
 82. Faria, D.C., Bagajewicz, M.J.: A new approach for global optimization of a class of MINLP problems with applications to water management and pooling problems. *AIChE J.* 58, 2320–2335 (2012)
 83. Castro, P.M.: Normalized multiparametric disaggregation: an efficient relaxation for mixed-integer bilinear problems. *J. Glob. Optim.* (2016). doi:10.1007/s10898-015-0342-z
 84. Belotti, P., Cafieri, S., Lee, J., Liberti, L.: Feasibility-Based Bounds Tightening via Fixed Points. In: Wu, W. and Daescu, O. (eds.) *Combinatorial Optimization and Applications: 4th International Conference, COCOA 2010, Kailua-Kona, HI, USA, December 18-20, 2010, Proceedings, Part I*. pp. 65–76. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
 85. Nagarajan, H., Lu, M., Yamangil, E., Bent, R.: Tightening McCormick Relaxations for Nonlinear Programs via Dynamic Multivariate Partitioning. In: Rueher, M.

- (ed.) Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings. pp. 369–387. Springer International Publishing, Cham (2016)
86. Castillo Castillo, P., Castro, P.M., Mahalec, V.: Global Optimization Algorithm for Large-Scale Refinery Planning Models with Bilinear Terms. *Ind. Eng. Chem. Res.* 56, 530–548 (2017). doi:10.1021/acs.iecr.6b01350
 87. Castro, P.M.: Spatial Branch-and-Bound Algorithm for MIQCPs featuring Multiparametric Disaggregation.
 88. Gleixner, A.M., Berthold, T., Müller, B., Weltge, S.: Three enhancements for optimization-based bound tightening. *J. Glob. Optim.* (2017). doi:10.1007/s10898-016-0450-4
 89. Tawarmalani, M., Sahinidis, N. V.: A polyhedral branch-and-cut approach to global optimization. In: *Mathematical Programming* (2005)
 90. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *J. Glob. Optim.* 59, 503–526 (2014)

Chapter 2: Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part I: Gasoline Blend Planning

This chapter has been published in the AIChE Journal. Complete citation:

Castillo Castillo, P. A., & Mahalec, V. (2014). Inventory pinch based, multiscale models for integrated planning and scheduling-part I: Gasoline blend planning.” *AIChE Journal*, 60(6), 2158–2178. *Wiley Online Library*, doi: 10.1002/aic.14423

Permission from © American Institute of Chemical Engineers.

In Chapter 2, the inventory pinch concept for production planning is revisited and the multiperiod inventory pinch (MPIP) algorithm is introduced for blend planning problems. MPIP relies on a two level decomposition of the original problem. At the 1st level, the blend recipes are determined by solving a multiperiod NLP model with periods delineated by inventory pinch points. The 2nd level is a multiperiod MILP model (with original number of periods defined by the planner) with fixed blend recipes. Both levels are formulated using discrete-time representation. One of the key features of the MPIP approach is that produces solutions with less variations in blend recipes.

The MPIP for blend planning is the base for the MPIP algorithm for blend scheduling presented in Chapter 3.

Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part I: Gasoline Blend Planning

Pedro A. Castillo Castillo and Vladimir Mahalec

Dept. of Chemical Engineering, McMaster University, Hamilton, ON, Canada L8S 4L8

DOI 10.1002/aic.14423

Published online March 7, 2014 in Wiley Online Library (wileyonlinelibrary.com)

A two-level algorithm to compute blend plans that have much smaller number of different recipes, much shorter execution times, and the same cost as the corresponding multiperiod mixed-integer nonlinear programming is introduced. These plans become a starting point for computation of approximate schedules, which minimize total number of switches in blenders and swing tanks. The algorithm uses inventory pinch points to delineate time periods where optimal blend recipes are likely constant. At the first level, nonlinear blend models are optimized via nonlinear programming. The second level uses fixed recipes (from the first level) in a multiperiod mixed-integer linear programming to determine optimal production plan followed by an approximate schedule. Approximate schedules computed by the multiperiod inventory pinch algorithm in most of the case studies are slightly better than those computed by global optimizers (ANTIGONE, GloMIO) while requiring significantly shorter execution times. Such schedules provide constraints for subsequent detailed scheduling in Part II. © 2014 American Institute of Chemical Engineers AIChE J, 60: 2158–2178, 2014

Keywords: gasoline blend planning, inventory pinch, recipe optimization, minimum number of recipes, multiscale planning model

Introduction

A supply chain is a system of all the activities required to produce and distribute a product starting from raw materials to the final product delivery to the end customer. Supply chain optimization involves making decisions at different temporal, spatial, and process scales (see Figure 1). Strategic planning determines decisions such as supplier selection, plant location, production system selection, distribution structure, and sales programs. Medium-term planning defines, in a weekly or monthly basis, the production targets of each plant site, the necessary stock levels of the inventories at various locations, and how much of each product is going to be transported from each plant to each warehouse or storage depot. Short-term planning is similar to medium-term planning but it is carried out for a shorter time horizon with smaller time scale (e.g., daily basis). Short-term production planning is usually called production scheduling and it deals with the assignment and sequencing of tasks for specific production units. Integration of these different decision levels is one of the main issues in the petroleum supply chain management due to the large physical supply network and the complex operations in an oil refinery.¹

The refinery is the main element of the petroleum supply chain. The final section of a crude-oil refinery consists of the blending units to produce the final liquid products and shipping operations. Minimization of the gasoline blending costs

has a major impact on profitability of crude-oil refineries.^{2,3} A sample gasoline blending system is shown in Figure 2. Accurate optimization of gasoline blends requires nonlinear models. Nonlinearities may appear in the blending model due to (1) nonlinear blending properties, (2) inclusion of the qualities of future product inventory in the multiperiod model, and (3) other attributes of the pooling problem. Kelly⁴ pointed out that formulation of planning models with nonlinearities is becoming more spread in practice due to: (1) recent complex government regulations, (2) raw materials being more expensive and of poorer quality, (3) new and more sophisticated production processes, and (4) higher energy, chemical, and utility costs.

One approach to gasoline blending is to compute blend recipes and detailed blend schedules simultaneously via continuous time model.^{5,6} Due to complexity of the mixed integer formulation, such approaches have been limited to using linear blending models. Even with linear models, this approach often leads to very large computational times. Another approach is to decompose the problem into planning and scheduling. Gasoline blend planning determines the ratios of blend components to be used to produce specific products (i.e., the blend recipes) and the volume to blend of each product along the planning horizon in order to meet product quality specifications, product demand requirements, and minimum blend cost. This has the advantage that nonlinear models can be used at the planning, as integer variables appear only with the minimum blend threshold constraint. As blend plans are feasible at the time period boundaries, the constraints they provide for detailed blend scheduling may be intraperiod infeasible. Hence, integration of planning and scheduling is required to ensure feasibility.

Additional Supporting Information may be found in the online version of this article.

Correspondence concerning this article should be addressed to V. Mahalec at mahalec@mcmaster.ca.

© 2014 American Institute of Chemical Engineers

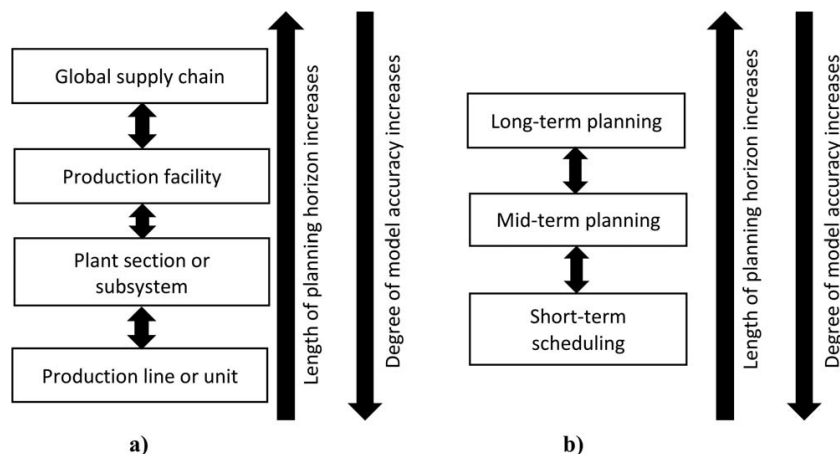


Figure 1. General scheme of hierarchical production planning with respect to (a) spatial scale and (b) time scale.

One problem that practitioners face when using nonlinear, multiperiod blend planning models is the blend recipe variations from one time period to another along the planning horizon, at the solution of the optimization model. In general, blend recipes vary for every blend instance and every time period, and they differ as well from one solver to another (even though the value of the economic objective function of the solutions is the same). Current practice is to avoid such variations by penalizing deviations of blend recipes in every period from some preferred blend recipe (e.g., Mendez et al.⁷). The disadvantage of this technique is that the final blend recipe and the cost will depend on the value given to the penalty coefficients. Another option is to use a set of preferred blend recipes (e.g., Jia and Ierapetritou²); however, such blends may not be the lowest cost blends.

This work presents an inventory pinch based, two-level algorithm to solve two problems:

1. Compute a blend plan based on a nonlinear blend model, such that the plan has a minimum number of optimal blend recipes along the planning horizon, and
2. Integrate nonlinear blending with approximate scheduling.

At the first level, we use the inventory pinch concept to determine minimal number of periods, each one of them having a single blend recipe for each grade of gasoline. The inventory pinch points delineate the initial time periods in a nonlinear programming (NLP) model used to compute these optimal blend recipes. At the second level, we use these optimal blend recipes from the first level to formulate a fixed-recipe multiperiod mixed-integer linear programming (MILP) to compute:

1. Blend plan: how much of each grade to produce and when to produce it, and when to change swing tanks from service to another. This model provides feasible inventory profiles at the second level period boundaries.
2. Approximate schedule: determine the production sequence that minimizes blender switches and switches of the swing tanks from one service to another along the planning horizon. Period boundaries at the second level are typically 1/2 day or 1 day long; that duration is set based on the minimum time a swing tank is expected to be in a given service. We call this an approximate schedule because it is

computed by an aggregated model, which does not consider minimum changeover times and other logistic rules that may be in place. This is similar to the definition of approximate scheduling model used by Maravelias and Sung.⁸

The second level MILP model contains slack variables to obtain a numerical solution even if it is not physically feasible. If the second level MILP slack variables have non-zero values, we subdivide the corresponding period at the first level NLP model, recompute the blend recipes and then recompute the second level. Hence, we increase number of recipes only when such a change is required to ensure optimality and feasibility of the solution.

For blend planning problems, we compare our solutions to the corresponding full-space mixed-integer nonlinear programming (MINLP) model. The results show that the blend costs computed by the full-space MINLP and our algorithm are the same. Our solutions have substantially smaller number of distinct blend recipes and also require much shorter

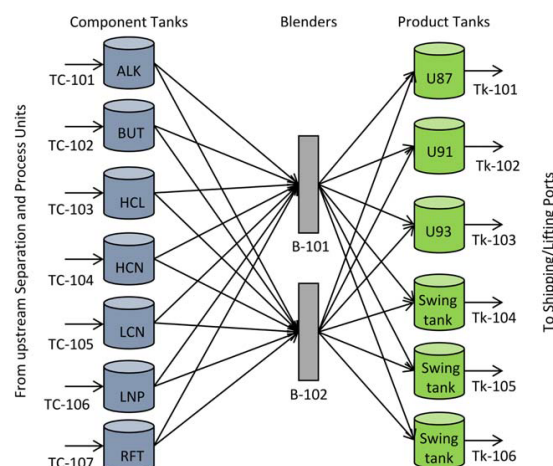


Figure 2. Sample gasoline blending system.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

execution times. Approximate schedules computed by our algorithm define constraints for detailed scheduling. Part II of this article integrates planning and detailed scheduling by introducing a third decomposition level, which computes detailed schedule.

Problem Statement

Blend planning

In this work, we address the gasoline blend planning problem stated as follows:

Given.

1. A planning horizon $[0, H]$.
2. A set of components, their properties, initial inventories, costs, and flow rates along the planning horizon (i.e., supply profile).
3. A set of products (i.e., gasoline grades) with prescribed minimum and maximum quality specifications, their initial inventories, and corresponding initial quality.
4. A set of delivery orders for each product along the planning horizon (i.e., demand profile).
5. The maximum blending capacity of each blender.
6. A set of storage tanks and their minimum and maximum hold ups.
7. Nonlinear blending model.

The Objective is to Determine.

1. The products that the blenders should produce in each time period and how much.
2. The best blend recipes for each product.
3. The inventory profiles of all the component and product tanks.
4. Swing tanks allocation in each time period.

Minimizing.

1. The blend cost associated with the amount of blend components used.

Subject to.

1. Extending the use of the same blend recipe as long as possible.
2. If a blender is to produce a product, it must blend at least a minimum amount.

Assuming.

1. Flow rate profile of each component from the upstream process is piecewise constant.
2. Component quality profile is piecewise constant.
3. Perfect mixing occurs in the blender.
4. Changeover times between product runs on the blender are product dependent but sequence independent.
5. Each order involves only one product and is completed during the respective time delivery window (otherwise, the problem is considered infeasible).
6. Swing tanks are allocated to a particular service throughout duration on a given period at the second level of the algorithm.

Approximate blend scheduling

Blend planning problem is extended to approximate blend scheduling by:

Minimizing.

1. The blend cost associated with the amount of blend components used plus the cost of switching blender operations plus the cost of switching swing tanks to different service.

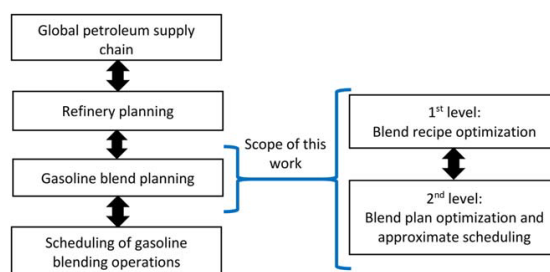


Figure 3. Proposed decomposition of the gasoline blend planning problem.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Solution Approach

Although production planning optimization of the global supply chain of an enterprise can be carried out by a single MILP and MINLP model, a hierarchical framework is preferred because: (1) it solves more tractable and smaller-size models and (2) it does not require large amount of forecast data, thus reducing the magnitude of the forecast errors in the planning process.⁹ In a hierarchical planning system, decisions at the upper levels are made first and they impose constraints on the decisions at the lower levels. Lower levels' solutions provide feedback to evaluate the upper levels' decisions. Each hierarchical level has its own characteristics, including length of the planning horizon, level of detail of the required information and forecasts (see Figure 1), and scope of the planning activity.⁹

In this work, we solve the gasoline blend planning problem using the hierarchical framework shown in Figure 3. The first level computes the optimal blend recipes and when it should be blended using the recipes from the first level. We assume that the flow rates of the refinery intermediate products arriving to the blending system (i.e., blend components) are given by the solution of the refinery planning level; therefore, the inventory cost need not to be included in the gasoline blend planning model as the refinery will carry the inventories as either blend components or as finished gasoline grades.

Aggregate planning

To reduce the disadvantages of a detailed formulation (e.g., large execution times, intractable models, large effect of the forecast errors and so forth), models at the upper levels are usually aggregated formulations of the lower levels.⁹ The adjective "aggregated" indicates that (1) some resources or tasks have been grouped into corresponding families and/or (2) the availability of the resources in one time interval is the cumulative amount available during that interval. Several aggregation/disaggregation techniques for production planning have been discussed in the literature. Nam and Logendran¹⁰ presented a survey of methodologies to formulate and solve aggregate production planning problems.

Bitran and Hax⁹ proposed a hierarchical planning model of a multiproduct batch plant. Feedback and interactions among the hierarchical levels are carried out by incorporating conditions not included initially in the models.

Axsater and Jonsson¹¹ presented a planning model for a manufacturing company where items and machines are

grouped according to the ratio between capacity requirements and available capacity, or according to the flows of items. Disaggregation is carried out by simulation using two different heuristic procedures.

Verderame and Floudas¹² proposed a multiperiod MILP planning model of a multisite production and distribution network. The model computes the daily production profile for each facility and the shipment profiles from the facilities to the distribution centers. Key assumptions in their model are: (1) sequencing constraints are not included (i.e., the solution of the planning model will be used as a basis to compute the detailed schedule), (2) more than one task can be executed by the same unit in the same facility and the same time period, and (3) unused time of the production units is transferred to future time periods. Therefore, the time periods, even as small as one day, can be considered as aggregated periods. Assumption (1) reduces the number of discrete variables, and Assumptions (2) and (3) eliminate unnecessary downtime of the processing units due to finite discretization of the time horizon. Comparison with the Kallrath planning model for multisite production (Timpe and Kallrath¹³) shows that Verderame and Floudas¹² model yields a tighter upper bound on the true production capacity.

Thakral and Mahalec¹⁴ developed an algorithm that optimizes blend recipes using a multiperiod MINLP planning model, minimizes blender switches through the use of a genetic algorithm, and detects infeasibilities via agent-based simulation. The time periods of the MINLP model are considered as aggregated as the same resources (e.g., blenders) are shared for different tasks (e.g., blends of different products) in the same time period; therefore, intraperiod infeasibilities may appear. If an infeasibility is encountered, the corresponding period is subdivided and the blend recipes are recomputed. The iterative algorithm stops when no infeasibilities appear.

Pinch analysis was used in aggregate production planning by Singhvi and Shenoy.¹⁵ They presented the demand and production composite (cumulative) curves, as well as a grand composite curve that showed the inventory levels as a function of time. They define the pinch as the point where the production composite touches the demand composite. They considered only one product and their main objective was to determine the production targets along the planning horizon from the pinch analysis, and then use these targets to solve a MILP model that minimizes the material, inventory, and labor costs, subject to material balance equations and production capacity constraints. The inventory levels are penalized in the objective function but no inventory capacity constraints are included. This methodology aims to provide at least a near-optimal solution. Singhvi et al.¹⁶ extended the pinch analysis to the scenario with multiple products and presented an algorithm to determine the production sequence based on some heuristic rules and assuming that the demand must be met only at the end of the horizon. Ludwig et al.¹⁷ applied the pinch analysis of Singhvi and Shenoy¹⁵ to a process with seasonal supply. The composite supply curve is constructed similarly to the composite demand curve and the composite production curve must not cross either curve to avoid supply shortages or product stock-outs, respectively.

Foo et al.¹⁸ implemented the graphical pinch methodology from Singhvi and Shenoy¹⁵ into an algebraic technique which can be easily programmed into a spreadsheet. They include minimum and maximum product inventory constraints and the capability of scheduling a plant shut down.

However, their algorithm considers only one product. They pointed out that further improvements of this technique were required to handle more complex supply chain planning problems; they expressed a belief that pinch analysis may provide a deeper analysis of the physical problem. They viewed the pinch point as an indicator of the time within a planning horizon when more accurate forecast data were needed because the pinch point represented the scheduling bottleneck.

Castillo et al.¹⁹ defined the inventory pinch point for the gasoline blend planning problem in a similar manner to Singhvi and Shenoy,¹⁵ but they utilized it to define the aggregated time intervals where one single blend recipe per product was likely to be used without incurring an infeasible operation. In addition, Castillo et al.¹⁹ included in their model: nonlinear product quality constraints, blending equations, availability of raw materials, time delivery windows, and capacity loss due to product switching in the blenders.

Inventory pinch concept

In this work, we use an aggregation/disaggregation approach based on inventory pinch points in order to (1) reduce execution times when compared with full-space MINLP model and (2) reduce the number of different blend recipes. A brief review of the inventory pinch concept is presented next; a more detailed explanation is found in Castillo et al.¹⁹. To explain the inventory pinch concept, we define the cumulative total demand (CTD) curve and the cumulative average total production (CATP) curve. The CTD curve is constructed by adding the cumulative demands of all products over time. The CATP curve consists of straight line segments, its starting point at time zero is the initial total product inventory available to deliver (i.e., the sum of all initial product inventories minus the minimum hold ups of the tanks), denoted as $V(0)$, and its final point at the end of the planning horizon corresponds to that of the CTD curve plus the final aggregated target inventories. The slope of each segment of the CATP curve must be equal or greater than zero and less than the maximum aggregated blending capacity (i.e., all blenders are aggregated as one single blending unit). The slope of the CATP curve can only change if the CATP is touching the CTD curve. CATP curve has the minimum number of segments required to remain above the CTD curve within the planning horizon. The inventory pinch points are defined as the points in time where the CATP curve changes its slope. Figure 4 shows examples of this concept and the grand composite curve which is computed as the difference between the cumulative total production (CTP) curve and the CTD curve. The inventory pinch points delimit the smallest number of time intervals where a constant blend recipe is likely to lead to a feasible blend plan; that is, no inventory infeasibilities appear when disaggregating the first level decisions at the second level. An inventory infeasibility is defined as the missing volume to fulfill certain demand order or the overflow (or runoff) that occurs in a storage tank.

Outline of the algorithm

At the first level, the boundaries of the time periods are delimited by the inventory pinch points, or by the times when the quality values of the blend components change, and the times when the unit cost of blend components or products vary. At the second level, the boundaries of the

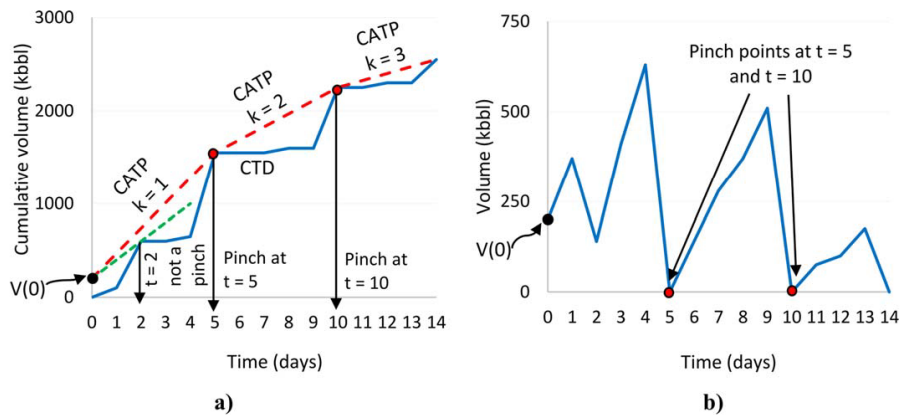


Figure 4. Inventory pinch point examples (a) on the cumulative curves and (b) on the composite curve.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

time periods are defined by the planner based on operational conditions, for example, the minimum time a storage tank will be holding a specific product or the minimum time that a blending unit will require to produce the minimum threshold amount. In addition, boundaries of the first level also become period boundaries at the second level. Notice that the length of the time periods at either level does not need to be uniform. For sake of exposition, the time periods of the first level are denoted as L1-periods and those of the second level as L2-periods. One L1-period contains one or more L2-periods; therefore, the product demand, blend component supply, and blend capacity in a L1-period are the aggregated values of the corresponding L2-periods (see Figure 5).

The inventory pinch points delineate the time periods of constant blend recipes that do not produce inventory infeasibilities due to peaks in demand because these peaks are already considered in the computation of blend recipes. However, other inventory infeasibilities can appear due to

the logistic constraints such as minimum blend runs, specific production sequences, or due to highly variable supply profiles of blend components. As these infeasibilities cannot be detected in advance, they will be accounted for in an iterative manner that refines the blend recipes if inventory infeasibilities are encountered. Hence, the inventory pinch concept allows us to start with the smallest number of time periods at the first level and increase that number only if required to eliminate inventory infeasibilities. It also provides the minimum production quantities to meet the demand of each product by solving a simple material balance for each time period delimited by pinch points: (aggregated production) = (aggregated demand) + (final inventory) – (initial inventory).

Although we consider only one storage tank per blend component, we assume that there are tanks that may store different gasoline grades at different times in the planning horizon, that is, swing tanks. At the first level, individual product tanks are aggregated into a single inventory capacity,

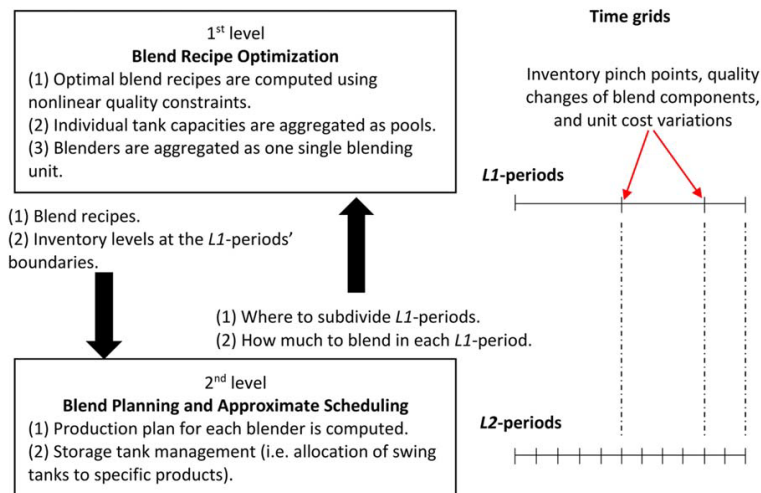


Figure 5. Inventory pinch based algorithm for gasoline blend planning and approximate scheduling.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

that is, a product pool; then at the second level, these pools are disaggregated into individual storage tanks.

Mathematical Models

The models presented here are developed based on the gasoline blend planning problem; however, they can be used for similar processes with few modifications. The following sets are used:

- A = $\{(\alpha) \mid \text{set of different supply flow rates of blend components}\}$
- Bl = $\{(\text{bl}) \mid \text{set of blenders}\}$
- E = $\{(e) \mid \text{set of quality properties}\}$
- I = $\{(i) \mid \text{set of blend components}\}$
- J = $\{(j) \mid \text{set of product storage tanks}\}$
- K = $\{(k) \mid \text{set of time periods at the first level or L1-periods}\}$
- M = $\{(m) \mid \text{set of time periods at the second level or L2-periods}\}$
- O = $\{(o) \mid \text{set of orders}\}$
- P = $\{(p) \mid \text{set of products}\}$
- BJ = $\{(\text{bl}, j) \mid \text{blender bl can feed tank } j\}$
- BP = $\{(\text{bl}, p) \mid \text{blender bl can process product } p\}$
- JP = $\{(j, p) \mid \text{tank } j \text{ can hold product } p\}$
- KA = $\{(k, \alpha) \mid \text{blend component supply profile } a \text{ occurs within L1-period } k\}$
- MA = $\{(m, \alpha) \mid \text{blend component supply profile } a \text{ occurs within L2-period } m\}$
- MK = $\{(m, k) \mid \text{L2-periods contained in each L1-period}\}$
- MKF = $\{(m, k) \mid \text{last L2-period of each L1-period}\}$

First level—Blend recipe optimization

The objective of the first level is to determine the volume fractions (i.e., blend recipes) to mix different blend components available at the refinery in such a way that the final products meet the demand and quality specifications and the blend cost is the minimum possible. The values of the product demand, blend component supply, and blend capacity are aggregated values for each of the L1-periods. At this level, product storage tanks are aggregated into product pools. The objective function defined by Eq. 1 minimizes the blend cost and the inventory infeasibilities. The penalty coefficients for the product slacks variables are much greater than the penalty coefficients for the component slacks variables (i.e., $\text{Penalty}_{\text{pool,L1}} \gg \text{Penalty}_{\text{bc,L1}}$). The blend cost is given by Eq. 2. Slack variables will be zero at the optimal solution (i.e., penalty coefficients will not affect the final blend cost); if slack variables have non-zero values, then the problem is infeasible as there are not enough blend components to blend products as required by the quality specifications or demand requirements.

Since the inventories will be at the minimum level allowed at the end of each L1-period (because the CATP curve touches the CTD curve), the inventory cost need not to be included in the objective function of the first level

$$\min Z_{L1} = \text{Blend Cost}_{L1} + \sum_{k=1}^K \left\{ \sum_{i \in I} \text{Penalty}_{\text{bc,L1}} \cdot \left(S_{\text{bc,L1}}^+(i, k) + S_{\text{bc,L1}}^-(i, k) \right) + \sum_{p \in P} \text{Penalty}_{\text{pool,L1}} \cdot \left(S_{\text{pool,L1}}^+(p, k) + S_{\text{pool,L1}}^-(p, k) \right) \right\} \quad (1)$$

$$\text{Blend Cost}_{L1} = \sum_{k=1}^K \left(\sum_{i \in I, p \in P} \text{Cost}_{\text{bc}}(i) \cdot V_{\text{comp,L1}}(i, p, k) \right) \quad (2)$$

The volume balance equations and inventory constraints for component and product tanks for every period k are given by Eqs. 3–6

$$\sum_{a \in \text{KA}} [F_{\text{bc}}(i, a) \cdot t_{\text{bc,L1}}(a, k)] + V_{\text{bc,L1}}(i, k-1) - V_{\text{bc,L1}}(i, k) - \sum_{p \in P} \quad (3)$$

$$V_{\text{comp,L1}}(i, p, k) + S_{\text{bc,L1}}^+(i, k) - S_{\text{bc,L1}}^-(i, k) = 0 \quad \forall i, k \geq 1$$

$$V_{\text{blend,L1}}(p, k) + V_{\text{pool,L1}}(p, k-1) - V_{\text{pool,L1}}(p, k) - \text{Demand}_{\text{pool,L1}}(p, k) + S_{\text{pool,L1}}^+(p, k) - S_{\text{pool,L1}}^-(p, k) = 0 \quad (4)$$

$$\forall p, k \geq 1$$

$$V_{\text{bc}}^{\min}(i) \leq V_{\text{bc,L1}}(i, k) \leq V_{\text{bc}}^{\max}(i) \quad \forall i, k \quad (5)$$

$$V_{\text{pool}}^{\min}(p) \leq V_{\text{pool,L1}}(p, k) \leq V_{\text{pool}}^{\max}(p) \quad \forall p, k \quad (6)$$

In Eq. 3, α stands for a specific supply profile of blend components. Parameter $t_{\text{bc,L1}}(\alpha, k)$ defines the aggregate duration of the time intervals when supply profile α occurs during period k . Notice that the sum of the durations of these time intervals must be equal to the length of period k

$$\sum_{a \in \text{KA}} t_{\text{bc,L1}}(a, k) = t_{L1}(k)$$

Therefore, the first term in Eq. 3 represents the aggregated availability of blend component i in period k . Equations 7 and 8 specify the initial inventories of the component tanks and product pools, respectively

$$V_{\text{bc,L1}}(i, k=0) = V_{\text{bc}}^{\text{start}}(i) \quad \forall i \quad (7)$$

$$V_{\text{pool,L1}}(p, k=0) = V_{\text{pool}}^{\text{start}}(p) \quad \forall p \quad (8)$$

Equations 9 and 10 are used to determine the blend recipes. Equation 11 sets the minimum and maximum compositions of each product

$$V_{\text{comp,L1}}(i, p, k) = r(i, p, k) \cdot V_{\text{blend,L1}}(p, k) \quad \forall i, p, k \geq 1 \quad (9)$$

$$\sum_{i \in I} r(i, p, k) = 1 \quad \forall p, k \geq 1 \quad (10)$$

$$r^{\min}(i, p) \leq r(i, p, k) \leq r^{\max}(i, p) \quad \forall i, p, k \geq 1 \quad (11)$$

Equations 12a and 12b are the linear quality constraints, that is, the quality property e is assumed to blend linearly in a volumetric basis

$$\sum_{i \in I} V_{\text{comp,L1}}(i, p, k) \cdot Q_{\text{bc}}(i, e, k) \leq Q_{\text{pr}}^{\max}(p, e) \cdot V_{\text{blend,L1}}(p, k) \quad \forall e, p, k \geq 1 \quad (12a)$$

$$\sum_{i \in I} V_{\text{comp,L1}}(i, p, k) \cdot Q_{\text{bc}}(i, e, k) \geq Q_{\text{pr}}^{\min}(p, e) \cdot V_{\text{blend,L1}}(p, k) \quad \forall e, p, k \geq 1 \quad (12b)$$

Equations 13 and 14 are the nonlinear quality constraints; they are expressed as a function of the blend recipe and the qualities of blend components

$$Q_{\text{pr}}(p, e, k) = f(r(i, p, k), Q_{\text{bc}}(i, e, k)) \quad \forall e, p, k \geq 1 \quad (13)$$

$$Q_{\text{pr}}^{\min}(p, e) \leq Q_{\text{pr}}(p, e, k) \leq Q_{\text{pr}}^{\max}(p, e) \quad \forall e, p, k \geq 1 \quad (14)$$

The only nonlinear quality constraint in our case studies is Eq. 15, which computes the Reid vapor pressure (RVP) (Sing et al.³)

$$Q_{pr}(p, e, k) = \left[\sum_{i \in I} r(i, p, k) \cdot Q_{bc}(i, e, k)^{1.25} \right]^{0.8} \quad \forall e = \text{RVP}, p, k \geq 1 \quad (15)$$

Equations 1–15 complete the NLP model for the first level. In the first iteration of the algorithm, the volumes to be blended in each L1-period are the minimum amount required to fulfill the demand. This provides a lower bound of the global blend cost.

It is important to note that discrete variables are not required at the first level to observe minimum blend size thresholds and maximum blenders' capacities. The volumes to be blended are adjusted if needed before solving the first level model (see Algorithm Steps).

Second level—Blend plan optimization and approximate blend schedule optimization

The blend plan defines (1) how much to blend of each product and in which blender in each L2-period; (2) allocation of swing tanks to specific products in each L2-period; and (3) the inventory profiles of all storage tanks along the planning horizon.

The second level uses the optimal blend recipes from the first level: the blend recipes of each L1-period are fixed in the corresponding L2-periods. The aggregated decisions of the first level are now disaggregated at the second level. This disaggregation step uses a MILP model to set constraints such as the minimum blend size threshold, and others that require discrete variables. The second level is solved first as a blend planning problem. This enables rapid computation of an optimal blend plan which is feasible with respect to the component and product availabilities and inventories. Once an optimal blend plan is known, scheduling of blenders and swing tankage is carried out (approximate scheduling).

Blend Planning—Objective Function at the Second Level. The objective function of the second level, Eq. 16, minimizes the blend cost and the inventory slack variables. Inventory slack variables will be zero at the solution of the second level if a feasible operation can be obtained using the blend recipes computed at the first level. If this is not the case, the inventory slacks will show which specific products and in which L2-periods cannot be produced in the amounts required to meet the demand. To ensure this, two things are required:

1. Penalty coefficients for the products' inventory slack variables are smaller in comparison with the penalty for the components' inventory slacks (i.e., $\text{Penalty}_{bc,L2} \gg \text{Penalty}_{pr,L2}(m) \forall m$).

2. The penalty coefficients for the product inventory slacks decrease with time (i.e., $\text{Penalty}_{pr,L2}(m) > \text{Penalty}_{pr,L2}(m+1) \forall m$) to move the inventory infeasibilities as late as possible in the planning horizon, thus maximizing the use of a given blend recipe. The penalty coefficients must decrease as fast as possible and after each L1-period boundary a significant change must take place.

If the solution of the MILP has inventory infeasibilities, it signifies that component supply or blender constraints are such that the recipes computed at the first level are not feasible within a L1-period. The algorithm will subdivide such L1-periods and reoptimize the blend recipes

$$\min Z_{L2}^{feas} = \text{Blend Cost}_{L2} + \sum_{m=1}^M \left\{ \begin{aligned} & \sum_{i \in I} \text{Penalty}_{bc,L2}(m) \cdot \left(S_{bc,L2}^+(i, m) + S_{bc,L2}^-(i, m) \right) \\ & + \sum_{p \in P} \text{Penalty}_{pool,L2}(m) \cdot \left(S_{pool,L2}^+(p, m) + S_{pool,L2}^-(p, m) \right) \\ & + \sum_{j \in J} \text{Penalty}_{pr,L2}(m) \cdot \left(S_{pr,L2}^+(j, m) + S_{pr,L2}^-(j, m) \right) \end{aligned} \right\} \quad (16)$$

Equation 17 computes the blend cost

$$\text{Blend Cost}_{L2} = \sum_{m=1}^M \left(\sum_{(bl,p) \in BP} \sum_{i \in I} \text{Cost}_{bc}(i) \cdot V_{comp,L2}(i, p, bl, m) \right) \quad (17)$$

Approximate Scheduling—Objective Function at the Second Level. After the blend recipes of the first level have been proven to generate a feasible blend plan, then, the approximate scheduling is solved. The scheduling objective function at the second level (Eq. 18) minimizes the number of possible product transitions in the blenders ($x_{e,L2}(bl, g)$) and in the product storage tanks ($ue_{L2}(j, m)$), as well as the number of blend instances ($x_{L2}(p, bl, m)$). Binary variable $x_{L2}(p, bl, m)$ defines a blend instance; that is, it determines if product p is going to be produced in blender bl during period m if its value is 1. Blend instances are penalized because a solution at the second level can suggest to blend the same product in the same blender for several adjacent L2-periods, thus not incurring in a penalty for product changeover in the blender; however, long blend runs are inefficient if the blender is not working close to full capacity. Therefore, it is better to have the minimum blend instances, and then reduce the expected number of product changeovers (i.e., $\text{PenaltyBR}_{L2}(bl) \gg \text{PenaltyBS}_{L2}$, for all bl). Because the inventory slacks are zero at the blend planning solution, we know that the blend recipes and inventory targets from the first level solution can yield a feasible blend plan; then, if those are fixed at the second level, the blend cost need not to be included in Eq. 18

$$\min Z_{L2}^{opt} = \sum_{m=1}^M \left\{ \begin{aligned} & \sum_{(bl,p) \in BP} \text{PenaltyBR}_{L2}(bl) \cdot x_{L2}(p, bl, m) \\ & + \sum_{j \in J} \text{PenaltyTS}_{L2}(j) \cdot ue_{L2}(j, m) \end{aligned} \right\} + \sum_{g=1}^G \left(\sum_{bl \in B1} \text{PenaltyBS}_{L2} \cdot x_{e,L2}(bl, g) \right) \quad (18)$$

Thus, the total cost at the second level is given by $Z_{L2} = Z_{L2}^{feas} + Z_{L2}^{opt}$. Given the assumption that the flow rates of blend components are given by the solution of the refinery planning level, the inventory cost need not to be included in the gasoline blend planning level as the refinery will carry the total inventories as either blend components or as finished gasoline grades.

Constraints at the Second Level. The volume balance equations for blend component tanks for every L2-period are given by Eq. 19

$$\begin{aligned}
& \sum_{a \in \text{MA}} F_{bc}(i, a) \cdot t_{bc,L2}(a, m) + V_{bc,L2}(i, m-1) \\
& - V_{bc,L2}(i, m) - \sum_{(bl, p) \in \text{BP}} V_{comp,L2}(i, p, bl, m) \\
& + S_{bc,L2}^+(i, m) - S_{bc,L2}(i, m) = 0 \\
& \forall i, m \geq 1
\end{aligned} \quad (19)$$

Once again, α stands for a specific supply profile of blend components. Parameter $t_{bc,L2}(\alpha, m)$ defines the aggregate duration of the time intervals when supply profile α occurs during period m . Notice that the sum of the durations of these time intervals within a L2-period, must be equal to the length of such L2-period

$$\sum_{a \in \text{MA}} t_{bc,L2}(a, m) = t_{L2}(m)$$

Similarly to Eq. 3, the first term in Eq. 18 represents the aggregated availability of blend component i in period m . In our case studies, there is only one interval α in each L2-period m .

Equation 20 fixes the blend recipe in its corresponding L2-periods. Note that $r(i, p, k)$ is a parameter and not a variable at the second level

$$\begin{aligned}
& V_{comp,L2}(i, p, bl, m) = r(i, p, k) \\
& \cdot V_{blend,L2}(p, bl, m) \quad \forall i, (p, bl) \\
& \in \text{BP}, (m, k) \in \text{MK}
\end{aligned} \quad (20)$$

Equation 21 establishes that a blender may only blend a certain number of products during a L2-period. $x_{L2}(p, bl, m)$ is a binary variable with value of 1 if product p is blended in bl during period m , and 0 otherwise

$$\sum_{p \in \text{BP}} x_{L2}(p, bl, m) \leq np(bl) \forall bl, m \geq 1 \quad (21)$$

Maximum blender capacity is enforced by Eq. 22. Because every product transition in the blender involves an idle time (due to sensor recalibration, equipment start-up, or other reasons), part of the blend capacity is lost. We assume that the blend capacity loss is equal to the product of the maximum blending capacity and the minimum idle time required by a blender to process product p

$$\begin{aligned}
& \sum_{p \in \text{BP}} V_{blend,L2}(p, bl, m) + F_{blend}^{\max}(bl) \\
& \cdot \sum_{p \in \text{BP}} (\text{it}_{blend}^{\min}(p, bl) \cdot x_{L2}(p, bl, m)) \\
& \leq F_{blend}^{\max}(bl) \cdot t_{L2}(m) \\
& \forall bl, m \geq 1
\end{aligned} \quad (22)$$

A blend run of a particular product must be less than or equal to the volume that the blender can process at maximum blending capacity in period m (Eq. 23), and greater than a threshold amount prespecified by the planner (Eq. 24). A blend run of any product must be greater than the volume that a blender can process at minimum blending rate in period m (Eq. 25)

$$\begin{aligned}
& V_{blend,L2}(p, bl, m) \leq F_{blend}^{\max}(bl) \cdot t_{L2}(m) \\
& \cdot x_{L2}(p, bl, m) \quad \forall (p, bl) \in \text{BP}, m \geq 1
\end{aligned} \quad (23)$$

$$\begin{aligned}
& V_{blend,L2}(p, bl, m) \geq V_{\text{MIN}}_{blend}(p, bl) \\
& \cdot x_{L2}(p, bl, m) \quad \forall (p, bl) \in \text{BP}, m \geq 1
\end{aligned} \quad (24)$$

$$\begin{aligned}
& V_{blend,L2}(p, bl, m) \geq F_{blend}^{\min}(bl) \cdot t_{blend}^{\min}(bl) \\
& \cdot x_{L2}(p, bl, m) \quad \forall (p, bl) \in \text{BP}, m \geq 1
\end{aligned} \quad (25)$$

Equations 22–25 are not enough to guarantee feasibility. The running times of each blend must be estimated and their sum plus the idle time must be equal to or less than the duration of the corresponding L2-period m . Equation 26 constrains the running time of a blend to be equal or greater than the minimum running time. Equations 27 and 28 set the lower and upper limits of a running time of a blend, respectively. Equation 29 enforces that the sum of the running times, plus product changeover times, is equal to or less than the L2-period duration

$$\begin{aligned}
& t_{blend,L2}(p, bl, m) \geq t_{blend}^{\min}(bl) \cdot x_{L2}(p, bl, m) \quad \forall (p, bl) \\
& \in \text{BP}, m \geq 1
\end{aligned} \quad (26)$$

$$\begin{aligned}
& t_{blend,L2}(p, bl, m) \geq \frac{V_{blend,L2}(p, bl, m)}{F_{blend}^{\max}(bl)} \quad \forall (p, bl) \in \text{BP}, m \\
& \geq 1
\end{aligned} \quad (27)$$

$$\begin{aligned}
& t_{blend,L2}(p, bl, m) \leq \frac{V_{blend,L2}(p, bl, m)}{F_{blend}^{\min}(bl)} \quad \forall (p, bl) \in \text{BP}, m \\
& \geq 1
\end{aligned} \quad (28)$$

$$\begin{aligned}
& \sum_{p \in \text{BP}} t_{blend,L2}(p, bl, m) + \sum_{p \in \text{BP}} \text{it}_{blend}^{\min}(p, bl) \cdot x_{L2}(p, bl, m) \\
& \leq t_{L2}(m) \quad \forall bl, m \geq 1
\end{aligned} \quad (29)$$

Equations 30 and 31 are the volumetric balance equations on the product pools and the individual storage tanks, respectively

$$\begin{aligned}
& \sum_{bl \in \text{BP}} V_{blend,L2}(p, bl, m) + V_{pool,L2}(p, m-1) \\
& - V_{pool,L2}(p, m) - \text{Deliver}_{pool,L2}(p, m) \\
& + S_{pool,L2}^+(p, m) - S_{pool,L2}(p, m) = 0 \\
& \forall p, m \geq 1
\end{aligned} \quad (30)$$

$$\begin{aligned}
& \sum_{bl \in \text{BPJ}} V_{trans,L2}(j, p, bl, m) + V_{pr,L2}(j, p, m-1) \\
& - V_{pr,L2}(j, p, m) - \text{Deliver}_{pr,L2}(j, p, m) \\
& + S_{pr,L2}^+(j, p, m) - S_{pr,L2}(j, p, m) = 0 \\
& \forall (j, p) \in \text{JP}, m \geq 1
\end{aligned} \quad (31)$$

The volume blended of product p in blender bl can only be sent to one product tank j during L2-period m , and that tank j must contain already product p or be empty (i.e., mixing of different products is not allowed in storage tanks). These constraints are represented by Eqs. 32–35. Binary variable $v_{L2}(j, p, bl, m)$ specifies if product p is transferred from blender bl to product tank j during period m . Binary variable $u_{L2}(j, p, m)$ specifies if product p is being stored in product tank j during period m

$$\begin{aligned}
& V_{trans,L2}(j, p, bl, m) \leq F_{blend}^{\max}(bl) \cdot t_{L2}(m) \cdot v_{L2}(j, p, bl, m) \\
& \forall (bl, p) \in \text{BP}, (j, bl) \in \text{BJ}, (j, p) \in \text{JP}, m \geq 1
\end{aligned} \quad (32)$$

$$\begin{aligned}
& \sum_{j \in \text{JP}} V_{trans,L2}(j, p, bl, m) = V_{blend,L2}(p, bl, m) \\
& \forall (bl, p) \in \text{BP}, (j, bl) \in \text{BJ}, m \geq 1
\end{aligned} \quad (33)$$

$$\sum_{j \in \text{JP}} v_{L2}(j, p, \text{bl}, m) \leq 1 \quad (34)$$

$$\forall (\text{bl}, p) \in \text{BP}, (j, \text{bl}) \in \text{BJ}, m \geq 1$$

$$v_{L2}(j, p, \text{bl}, m) \leq u_{L2}(j, p, m) \quad (35)$$

$$\forall (\text{bl}, p) \in \text{BP}, (j, \text{bl}) \in \text{BJ}, (j, p) \in \text{JP}, m \geq 1$$

Equation 36 states that only one product can be stored in a product tank j during a period m . Equation 37 constrains the volume stored of product p in tank j to be less than the maximum capacity of such tank. Equation 38 relates the product pool inventory with the individual tank inventories

$$\sum_{p \in \text{JP}} u_{L2}(j, p, m) = 1 \quad \forall j, m \quad (36)$$

$$V_{\text{pr}, L2}(j, p, m) \leq V_{\text{pr}}^{\max}(j) \cdot u_{L2}(j, p, m) \quad \forall (j, p) \in \text{JP}, m \quad (37)$$

$$V_{\text{pool}, L2}(p, m) = \sum_{j \in \text{JP}} V_{\text{pr}, L2}(j, p, m) \quad \forall p, m \quad (38)$$

Binary variable $ue_{L2}(j, m)$ is introduced to determine if a product transition in tank j has taken place at the beginning of period m (Eqs. 39 and 40). Equation 42 forces a product transition to occur on tank j if it is empty at the end of the previous period m

$$ue_{L2}(j, m) \geq u_{L2}(j, p, m) - u_{L2}(j, p, m-1) \quad \forall (j, p) \in \text{JP}, m \geq 1 \quad (39)$$

$$ue_{L2}(j, m) \geq u_{L2}(j, p, m-1) - u_{L2}(j, p, m) \quad \forall (j, p) \in \text{JP}, m \geq 1 \quad (40)$$

$$V_{\text{pr}, L2}(j, p, m-1) \leq V_{\text{pr}}^{\max}(j) \cdot (1 - ue_{L2}(j, m)) \quad \forall (j, p) \in \text{JP}, m \geq 1 \quad (41)$$

Equation 42 defines the products stored in the product tanks at the beginning of the scheduling horizon

$$u_{L2}(j, p, m=0) = u^{\text{start}}(j, p) \quad \forall (j, p) \in \text{JP} \quad (42)$$

Inventory constraints on storage tanks are given by Eqs. 43–45. Maximum pool capacity is no longer enforced at this level as there are constraints on the individual product tanks (see Eq. 37)

$$V_{\text{bc}}^{\min}(i) \leq V_{\text{bc}, L2}(i, m) \leq V_{\text{bc}}^{\max}(i) \quad \forall i, m \quad (43)$$

$$V_{\text{pool}, L2}(p, m) \geq V_{\text{pool}}^{\min}(p) \quad \forall p, m \quad (44)$$

$$V_{\text{pr}, L2}(j, p, m) \geq V_{\text{pr}}^{\min}(j) \cdot u_{L2}(j, p, m) \quad \forall (j, p) \in \text{JP}, m \quad (45)$$

Inventory levels at the boundaries of the L1-periods must be equal at the corresponding point in time at the second level. This is enforced by Eq. 46

$$V_{\text{pool}, L2}(p, m) = V_{\text{pool}, L1}(p, k) \quad \forall p, (m, k) \in \text{MKF} \quad (46)$$

Equations 47 and 48 set the initial state of the component tanks and product tanks, respectively

$$V_{\text{bc}, L2}(i, m=0) = V_{\text{bc}}^{\text{start}}(i) \quad \forall i \quad (47)$$

$$V_{\text{pr}, L2}(j, p, m=0) = V_{\text{pr}}^{\text{start}}(j, p) \quad \forall (j, p) \in \text{JP} \quad (48)$$

Equation 49 constrains the shipped/lifted amount of product p from tank j in period m to be equal or less than the volume that such tank can deliver at maximum delivery rate during period m . The amount of product p delivered by all product tanks is equal to that amount delivered from the cor-

responding product pool (Eq. 50). Equation 51 establishes that the demand must be met in each L2-period

$$\text{Deliver}_{\text{pr}, L2}(j, p, m) \leq D_{\text{pr}}^{\max}(j) \cdot t_{L2}(m) \cdot u_{L2}(j, p, m) \quad (49)$$

$$\forall (j, p) \in \text{JP}, m \geq 1$$

$$\text{Deliver}_{\text{pool}, L2}(p, m) = \sum_{j \in \text{JP}} \text{Deliver}_{\text{pr}, L2}(j, p, m) \quad \forall p, m \geq 1 \quad (50)$$

$$\text{Deliver}_{\text{pool}, L2}(p, m) = \text{Demand}_{\text{pr}, L2}(p, m) \quad \forall p, m \geq 1 \quad (51)$$

Equation 50 is used when the demand requirement for each L2-period is known exactly, that is, the delivery windows of the orders do not span more than one L2-period. Equation 50 can be substituted by Eqs. 52–55, which are included in order to handle delivery windows spanning several L2-periods. Equation 52 establishes that the amount delivered in period m is equal to a fraction of the total demand of order o , denoted as $of_{L2}(o, m)$. $uof_{L2}(o, m)$ is a parameter that represents if order o can be delivered in period m if its value is 1 (i.e., delivery time window of order o is contained totally or partially by L2-period m), or not if its value is zero. Equation 53 ensures that only the contracted demand for order o is shipped/lifted. Equation 54 forces completion of order o within the scheduling horizon. Equation 55 constrains the fraction delivered of order o during period m to be less than the maximum possible delivery of such order

$$\text{Deliver}_{\text{pool}, L2}(p, m) = \sum_{o \in \text{OP}} \text{Demand}(o) \cdot of_{L2}(o, m) \quad \forall p, m \geq 1 \quad (52)$$

$$of_{L2}(o, m) \leq uof_{L2}(o, m) \quad \forall o, m \geq 1 \quad (53)$$

$$\sum_{m=1}^M of_{L2}(o, m) = 1 \quad \forall o \quad (54)$$

$$\text{Demand}(o) \cdot of_{L2}(o, m) \leq D_{\text{order}}^{\max}(o) \cdot dt_{L2}(o, m) \quad \forall o, m \geq 1 \quad (55)$$

Constraints Specific to the Approximate Scheduling at the Second Level. Equations 56–61 are included in order to compute the product transitions in the blenders. The production sequence of each blender is determined by binary variable $y_{L2}(p, \text{bl}, g)$, which is computed based on the values from variable $x_{L2}(p, \text{bl}, m)$. The g index refers to position in the sequence where product p is processed by blender bl ; that is, g can be interpreted as well as a time slot to allocate the blend instances of period m to estimate the production sequence of blender bl . The number of these slots for one L2-period is equal to the number of different products. Equation 56 defines the products being processed in the blenders at the beginning of the scheduling horizon. Equation 57 ensures that only a product that is being blended in period m is allocated in a slot g corresponding to m . Equation 58 constrains that only one product can be allocated in slot g . Equation 59 allows a product being blended in period m to use more than one slot g corresponding to m . Equations 60 and 61 determine if a product transition has taken place. $xe_{L2}(\text{bl}, g)$ is a continuous variable that can only take 0–1 values due to Eqs. 60 and 61, and for being penalized in Eq. 18

$$y_{L2}(p, \text{bl}, g=0) = x^{\text{start}}(p, \text{bl}) \quad \forall (\text{bl}, p) \in \text{BP} \quad (56)$$

$$y_{L2}(p, \text{bl}, g) \leq x_{L2}(p, \text{bl}, m) \quad \forall p, \text{bl}, (g, m) \in \text{GM} \quad (57)$$

$$\sum_{p \in P} y_{L2}(p, bl, g) \leq 1 \quad \forall bl, g \geq 1 \quad (58)$$

$$\sum_{g \in m} y_{L2}(p, bl, g) \geq x_{L2}(p, bl, m) \quad \forall p, bl, m \geq 1 \quad (59)$$

$$x_{eL2}(bl, g) \geq y_{L2}(p, bl, g) - y_{L2}(p, bl, g-1) \quad \forall p, bl, g \geq 1 \quad (60)$$

$$x_{eL2}(bl, g) \geq y_{L2}(p, bl, g-1) - y_{L2}(p, bl, g) \quad \forall p, bl, g \geq 1 \quad (61)$$

It is important to note that the production sequence given by $y_{L2}(p, bl, g)$ provides a lower bound of the number of product transitions in the blenders, but the actual production sequence must still be computed. This is done in Part II of this article where detailed scheduling is integrated to the model.

Note that the second level MILP model does not deal with the blend recipe optimization and do not include the nonlinear terms corresponding to the quality constraints. Thus, all the nonlinearities intrinsic to the blending problem are solved at the first level. Appendix shows that the first and second level models are equivalent to full-space MINLP model.

Corresponding full-space MINLP model

Corresponding full-space MINLP model can be obtained from the second level model by dropping Eq. 20 and adding the following equations from the first level (such equations and its variables are written for all $m \geq 1$):

- The blend recipe computation: Eqs. 9–11; and
- The quality constraints: Eqs. 12a–15.

Although it is possible, we do not include constraints to minimize blend recipe variation to find the optimal solution of the original full-space MINLP model and compare it with the solution of our algorithm.

Special case: full-space MILP model

The second level model can be transformed into a full-space MILP planning model if:

a. Blend recipes are not required to be computed directly in the model. In this case, Eq. 20 is dropped from the second level model, and Eqs. 62 and 63 are added

$$\sum_{i \in I} V_{comp,L2}(i, p, bl, m) = V_{blend,L2}(p, bl, m) \quad (62)$$

$$\forall i, (p, bl) \in BP, m \geq 1$$

$$r^{\min}(i, p) \cdot V_{blend,L2}(p, bl, m) \leq V_{comp,L2}(i, p, bl, m) \leq r^{\max}(i, p) \cdot V_{blend,L2}(p, bl, m) \quad (63)$$

$$\forall i, (p, bl) \in BP, m \geq 1$$

a. All the quality properties are assumed to blend linearly (Eqs. 12a and 12b are added to the second level model but rewritten for all L2-periods) or if the nonlinear equations can be rewritten as linear constraints (e.g., the RVP nonlinear constraint given by Eq. 15 can be substituted by Eq. 64, which is linear)

$$\begin{aligned} Q_{pr}^{\min}(p, e)^{1.25} \cdot V_{blend,L2}(p, m) &\leq \sum_{i \in I} V_{comp,L2}(i, p, m) \\ Q_{bc}(i, e, m)^{1.25} &\leq Q_{pr}^{\max}(p, e)^{1.25} \cdot V_{blend,L1}(p, m) \end{aligned} \quad (64)$$

$$\forall e = RVP, p, m \geq 1$$

This linear transformation allows us to compare performance of our algorithm with a full-space MILP model. Not all nonlinear quality constraints can be rewritten exactly as lin-

ear constraints, and the aim of our algorithm is to handle problems with those nonlinearities.

As blend recipes are not computed directly by the model, it is only possible to minimize variations with respect to a given set of preferred blend recipes. These constraints are not included to find the optimal solution of the original full-space model.

Multiperiod Inventory Pinch Algorithm for Gasoline Blend Planning

The flowchart of the multiperiod inventory pinch (MPIP) algorithm is presented in Figure 6. Although this algorithm is based on the gasoline blend planning problem, it can be applicable to any system or process with similar features. The steps of the algorithm are the following:

Step 1: Construct the cumulative curves (CTD and CATP) and determine the pinch point(s) location.

Step 2: Set iteration counter $iter = 1$. Divide the planning horizon (at the first level) in the number of L1-periods indicated by the pinch points and the times when the quality values of the blend components change (i.e., $K^{(1)}$).

Step 3: Solve the first level Blend Recipe Optimization model.

• In the first iteration ($iter = 1$), the volumes to produce of each product in each L1-period k are the minimum amounts required to meet the aggregated demand in each L1-period

$$V_{blend,L1}(p, k) = Demand_{pool,L1}(p, k) + V_{pool,L1}^{target}(p, k) - V_{pool,L1}(p, k-1) \quad (65)$$

where $V_{pool,L1}^{target}(p, k)$ is the target inventory. Notice that if no target inventories are set, $V_{pool,L1}^{target}(p, k) = V_{pool,L1}^{\min}(p)$.

• If $V_{blend,L1}(p, k)$ violates the maximum blend capacity or the minimum blend size threshold constraints, volumes are adjusted by moving the least amount possible of volume to previous L1-periods (i.e., preblending).

• In subsequent iterations ($iter > 1$), the volumes to produce are defined according to the solution of the second level (see Step 6).

• If any inventory slack variable has a non-zero value at the solution, the problem is infeasible as the availability or quality of blend components is not sufficient to meet the product quality specifications (or to deliver the product within the delivery window). Stop.

Step 4: Solve the second level Blend Planning model.

Step 5: If the inventory slack variables from Step 4 are zero, an optimal blend plan has been found; go to Step 8. Otherwise, continue to Step 6.

Step 6: Subdivide the L1-period with the first infeasibility (see Figure 7).

• The volumes to be blended in each new L1-period are given by the solution of Step 5 plus the positive slacks minus the negative slacks

$$V_{blend,L1}(p, k) = \sum_{m \in MK} \left[\sum_{bl \in BP} V_{blend,L2}(p, bl, m) + S_{pool,L2}^+(p, m) - S_{pool,L2}(p, m) \right] \quad (66)$$

• If $V_{blend,L1}(p, k)$ violates the maximum capacity or minimum blend threshold constraints, volumes are adjusted by moving the least amount possible of volume to the previous L1-period or from the next L1-period, respectively.

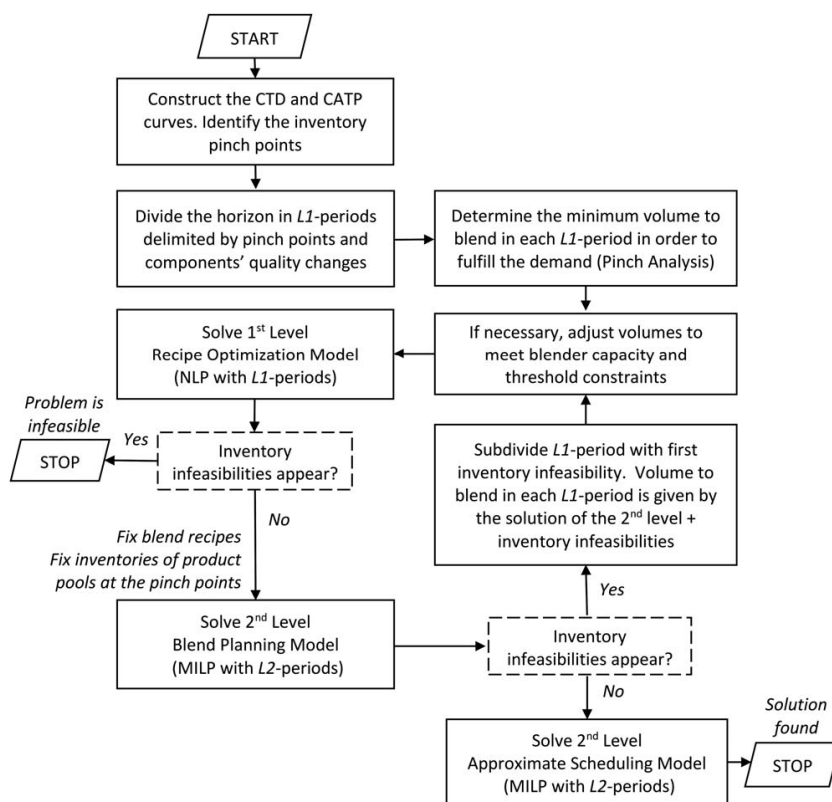


Figure 6. Flowchart for the MPIP planning algorithm.

Step 7: $K^{(iter+1)} = K^{(iter)} + 1$. $iter = iter + 1$. Go back to Step 3.

Step 8: Solve the second level Approximate Scheduling model, to reduce the number of blend instances and product transitions in the blenders and storage tanks. Stop.

Case Studies

The gasoline blending system shown in Figure 2, in configurations with one, two, and three blenders, has been studied. In all cases, the system uses seven blend components (ALK, BUT, HCL, HCN, LCN, LNP, and RFT) and produces three products (grades of gasoline U87, U91, and U93). Each component has their particular storage tank. There are three dedicated product tanks, one per each gasoline grade, and there are three swing tanks which can store any product, but only one at a given time. A planning horizon of 14 1-day L2-periods is used in all case studies ($H = 14$ days, $t_{L2}(m) = 1$ day for all m). The minimum volume to blend in each blender in each L2-period is $VMIN_{blend}(p,bl) = 30 \times 10^3$ bbl, for all p and bl. The demand orders involve a single product and their delivery time windows are assumed to be 1 day. Eight blend properties are considered: aromatic content (% by volume, ARO), benzene content (% by volume, BEN), olefin content (% by volume, OLF), motor octane number (MON), research octane number (RON), RVP (psi), specific gravity (SPG), and sulfur content (% by volume, SUL). All blend properties are assumed to blend linearly except RVP. Supporting

Information of this article contains data describing product specifications, properties, and availabilities of blend components, tankage capacities, and information required for full specification of the test cases. The cumulative curves and pinch points for each profile demand are shown in Figure 8. There is no maximum delivery rate for each order, that is, the maximum delivery rate of the tanks is the maximum limit. For the MILP volume allocation problem, the cost coefficients for component inventory slack variables are set to 1×10^9 , and the cost coefficients for product inventory slack variables appear in Supporting Information of this article. Table 1 shows data describing the blenders. At the second level, for all blenders, the penalty for a blend instance is

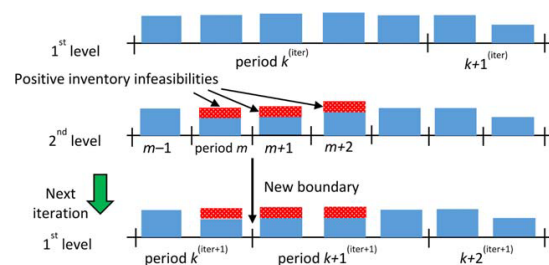


Figure 7. Subdivision of L1-periods according to the solution of the second level model.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

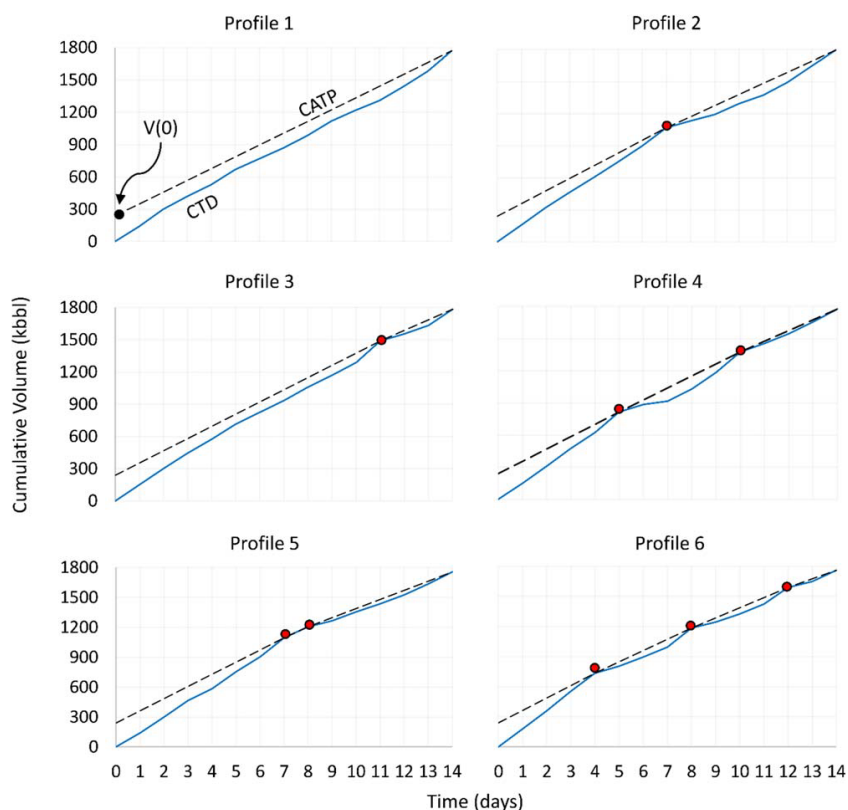


Figure 8. Cumulative curves and pinch points for the six demand profiles.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

20×10^3 \$, and for a product transition in a blender is 7×10^3 \$. Notice that a blend instance is more penalized than a product transition in a blender to use more efficiently the blenders (i.e., at higher production rates). All data are presented in English system of units as it prevails in the refining industry.

All case studies have been computed on a DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 Gb RAM) running Windows Server 2008 R2 OS. GAMS IDE 24.2.1 was used to solve each one of the case studies. The

first level NLP model was solved using IPOPT, and the second level model was solved using CPLEX 12.6. To compare the results obtained with the inventory pinch algorithm, the corresponding full-space MINLP and MILP models were solved for both the blend planning and approximate scheduling problems. DICOPT (using IPOPT version 3.8), BARON 9.3.1, ANTIGONE²⁰ 1.1, and GloMIQO^{21,22} 2.3 were the solvers used to solve the full-space MINLP models, and CPLEX 12.6 to solve the full-space MILP model. The mixed-integer quadratically constrained program (MIQCP)

Table 1. Blenders Data

Blender		A	B	C
Allowable products		U87, U91, U93	U91, U93	U87, U91
Minimum volume to blend in each L2-period ($\times 10^3$ bbl)		30	30	30
Maximum blending rate ($\times 10^3$ bbl/h)	Case study	1, 2, 3, 8, 9, 10	–	–
		4, 5, 6, 11, 12, 13	3	–
		7, 14	2.5	2
Minimum blending rate ($\times 10^3$ bbl/h)		1, 2, 3, 8, 9, 10	–	–
		4, 5, 6, 11, 12, 13	3	–
		7, 14	1.8	1.5
Minimum running time (h)	Product	U87	–	6
		U91	6	6
		U93	6	6
			6	–
Minimum idle time (h)		U87	–	2
		U91	1	1
		U93	1	1
			2	–

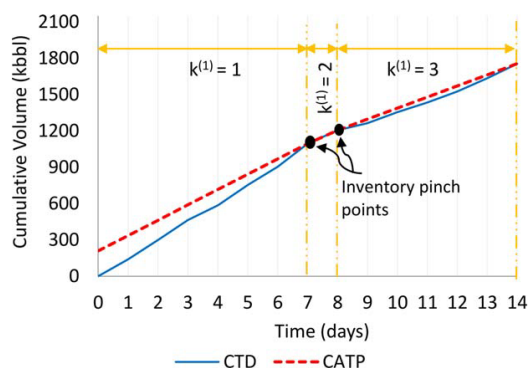


Figure 9. Case study 13—Cumulative curves, inventory pinch points, and L1-periods. Blend planning first iteration.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

required by GloMIQO is the full-space MINLP model described in the Mathematical Models section, but Eq. 64 replaces Eq. 15.

When solving the full-space MINLP model with DICOPT, starting points are important to find an optimal solution, or even a feasible one. In our case studies, the starting point of variable $V_{\text{blend},L2}(p,bl,m)$, for all p , bl , and m , was set above the minimum blend size threshold (i.e., $V_{\text{MINblend}}(p,bl)$).

As blend recipes computed by MINLP model are not exactly the same, in order to compute the number of different recipes in the MINLP solution, two recipes are considered to be different if the absolute change of composition percentage of any component is greater than

1%. As three grades of gasoline are being blended, the total number of different blend recipes for all products is divided by three.

Illustrative example with two blenders and irregular component supply

Case study 13 will be used to illustrate the steps of the MPIP planning algorithm. This case study has demand profile #5, two blenders (A and B), and the supply flow rate of components is irregular along the planning horizon.

Blend Planning: Iteration 1. The corresponding cumulative curves are shown in Figure 9, where it can be seen that there are two inventory pinch points, one at the end of Day 7, and the other at the end of Day 8. Hence, three L1-periods are required initially, $k^{(1)} = 1$, starts at the beginning of Day 1 until the first pinch point, $k^{(1)} = 2$ corresponds to Day 8, and $k^{(1)} = 3$ goes from the beginning of Day 9 to the end of the planning horizon. In the first iteration of the algorithm, the minimum production to meet the aggregated demand in each L1-period is given by Eq. 65. The blend recipes computed by the first level model are given in Table 2, and the blend cost is $\text{Blend Cost}_{L1} = 37,784.52 \times 10^3$ \$.

Using the blend recipes from the first level, the second level planning model is solved. Figure 10 shows the blend plan for each blender, but it contains inventory infeasibilities. The positive slack variables appear in L2-periods $m = 4$ (12.77×10^3 bbl) and $m = 5$ (14×10^3 bbl), both for product U91. The negative slack variables appears in period $m = 14$ for product U87 (10.42×10^3 bbl). From Figure 10 can be seen that blender A and B are working at full capacity during period $m = 5$; thus, the slack in period $m = 5$ must be preblended (see rule 6.a of the algorithm). L2-Periods before $m = 5$ have enough blending capacity

Table 2. Case Study 13—Optimal Blend Recipes (First Level)

Blend Comp.	First Iteration				Second Iteration						
	L1-Period	U87	U91	U93	L1-Period	U87	U91	U93			
ALK	$k^{(1)} = 1$	0.1747	0.245	0.1414	$k^{(2)} = 1$	0.1636	0.2422	0.1874			
BUT		0.0261	0.0368	0.0436		0.0262	0.0355	0.0435			
HCL		0.0241	0.0374	0.0327		0.027	0.0309	0.029			
HCN		0.0456	0.0615	0.0523		0.0366	0.04	0.0394			
LCN		0.2765	0.1698	0.1168		0.2458	0.1978	0.1293			
LNP		0.1708	0.0784	0.0368		0.1842	0.0854	0.031			
RFT		0.2823	0.3711	0.5765		0.3165	0.3682	0.5404			
ALK	$k^{(1)} = 2$				$k^{(2)} = 2$	0.1543	0.2486	0.1824			
BUT						0.0263	0.0373	0.0442			
HCL						0.0239	0.0465	0.0314			
HCN						0.058	0.0793	0.0529			
LCN						0.2791	0.1648	0.1174			
LNP						0.169	0.0653	0.0276			
RFT						0.2895	0.3583	0.5442			
ALK	$k^{(1)} = 3$				$k^{(2)} = 3$	0.1535	0.2434	0.1803			
BUT						0.0252	0.036	0.0434			
HCL						0.0286	0.038	0.0278	0.0289	0.0382	0.0277
HCN						0.0522	0.0606	0.0443	0.0533	0.0616	0.0443
LCN						0.3058	0.2003	0.1369	0.3056	0.1996	0.1372
LNP						0.1619	0.0702	0.0293	0.1613	0.0702	0.0289
RFT						0.2726	0.35	0.5395	0.2722	0.351	0.5381
ALK	$k^{(1)} = 3$				$k^{(2)} = 4$	0.0152	0.1363	0.09			
BUT						0.0197	0.0349	0.0416			
HCL						0	0	0			
HCN						0.0011	0.0021	0.0017	0.0011	0.0021	0.0017
LCN						0.427	0.1617	0.1496	0.4255	0.1566	0.1564
LNP						0.2016	0.1554	0.0825	0.2022	0.1571	0.0801
RFT						0.3355	0.5096	0.6346	0.3371	0.5144	0.628

The significance of the bold characters is that they represent the L1-periods, where k represents the elements of the set $K = (\text{L1-periods})$. The superscript inside the parenthesis represent the iteration number.

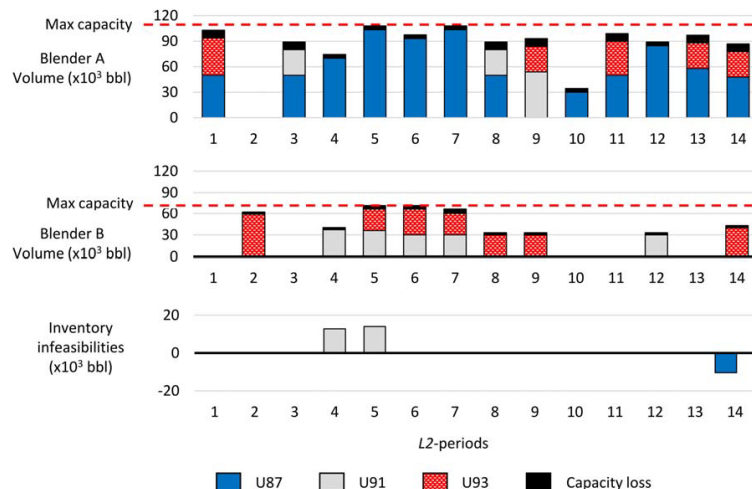


Figure 10. Case study 13—Blend plan with inventory infeasibilities. Blend planning first iteration.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

available to blend both infeasibilities. Therefore, the new boundary in the first level must be placed at the end of day 4.

The new period boundaries at the first level are shown in Figure 11; however, this time the CATP curve is replaced with the actual CTP curve that is constructed by aggregating the total production [i.e., $\sum_{p,bl} V_{blend,L2}(p,bl,m)$] over time. When the CTP curve is below the CTD, there are inventory infeasibilities. This is easier to see on the grand composite curve.

Blend Planning: Iteration 2. In this iteration, the production targets for each L1-period are computed using Eq. 66 and rule 6.a of the algorithm. The blend recipes computed by the first level model are given in Table 2, and the blend cost is $\text{Blend Cost}_{L1} = 37,784.52 \times 10^3$ \$, the same cost as in the first iteration.

Using the blend recipes from the second iteration, the second level planning model is solved. An optimal blend plan is found with a cost equal to $\text{Blend Cost}_{L2} = 37,784.52 \times 10^3$ \$, the same as that of the first level. Because there are no inventory infeasibilities this time, we proceed to solve the second level approximate scheduling model.

Approximate Scheduling. The approximate schedule with reduced number of blend instances and product transitions in the blenders is shown in Figure 12 and its total cost is

$Z_{L2} = 38,522.52 \times 10^3$ \$. Idle times between blend runs are not shown since the actual production schedule is not determined at this level. It is also determined that in this case no product changeovers are required in the swing tanks.

The component and product inventory profiles corresponding to the second level approximate scheduling (Figure 13) are at the allowed minimums at the inventory pinch points (i.e., at the end of the seventh and eighth day). Additionally, it is observed that the inventory levels of some blend components are at the minimum allowed at some points in the planning horizon. In this particular case study, this is observed at the inventory pinch points. In general, these “pinch points on the components’ inventory profiles” will appear at least for one blend component at the end of the planning horizon and at least at one inventory pinch point. There are two reasons for the appearance of these pinch points on the components’ inventory profiles: (a) cheaper components are used as much as possible, and (b) components with the necessary qualities to produce products under specification are scarce. If the “components’ pinch” occurs because of reason “(a),” trying to blend more volume (i.e., increased production) before a components’ pinch might result in a blend cost increase (i.e., more expensive materials are used since cheaper components are not available). If this

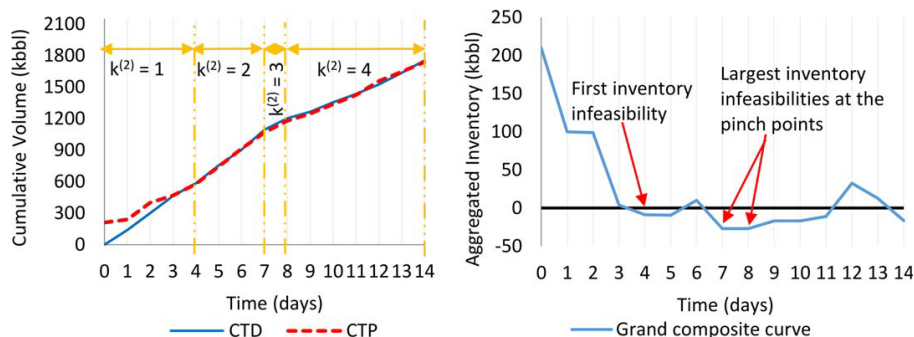


Figure 11. Case study 13—Cumulative curves and L1-periods.

Blend planning second iteration. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

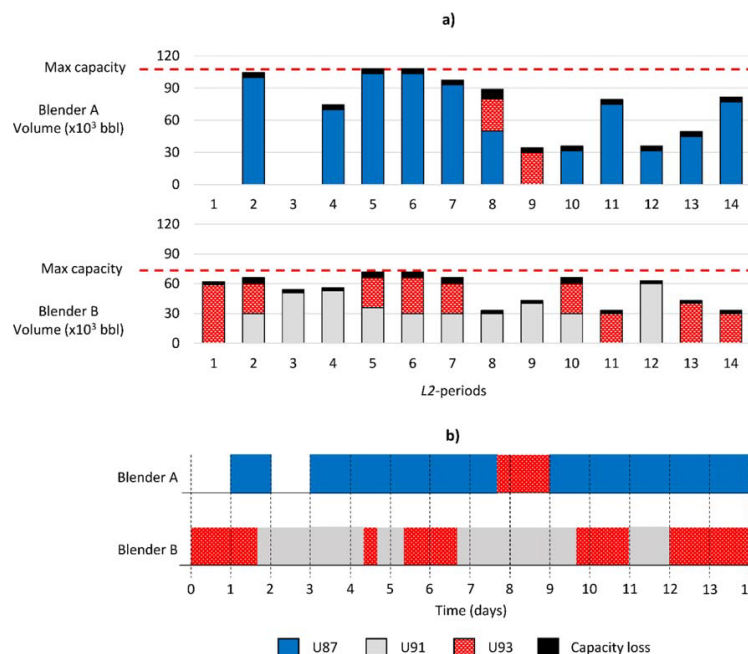


Figure 12. Case study 13—(a) Blend plan and (b) production sequence.

Approximate scheduling. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

type of pinch appears because of reason “(b),” then trying to increase production before the components’ pinch is not possible (first level will present non-zero slack variables) since the blend cannot reach the quality specifications. In this illustrative example, the components’ pinch occurs because of reason “(b)” there is not enough ALK and RFT available to blend more before Day 8. ALK and RFT have the highest RON and MON values from among the blend components; other components have RON and MON values smaller than the minimum spec, or if they have a greater RON and MON value, they have RVP and SPG value above the maximum spec. Therefore, no more than the production targets given in can be blended before Day 7 or Day 8 in Case study 13.

Results and Discussion

To evaluate the performance of the MPIP algorithm, we compared the solutions from our algorithm with those provided by the full-space models and the execution times required to obtain them.

Blend planning

All problems were solved to an optimality gap less than or equal to 0.001%. The MPIP algorithm and the full-space models found the same optimal objective function value (see Table 3). The solution times of the MPIP planning algorithm (using IPOPT for the NLP model and CPLEX for the MILP model) are much smaller than those required by DICOPT to solve the corresponding MINLP model (almost two orders of magnitude lower). We chose to compare with DICOPT solver because it provided the smallest execution times when compared with the other selected MINLP solvers. We know that the solutions found by DICOPT and by MPIP algorithm are globally optimal, as they are equal to the solution of full-space MILP (which in this special case, via transforma-

tions, is able to solve our test problems as linear MILP models). It is interesting to note that our algorithm leads to execution times which are about the same as the times required to solve the equivalent MILP model. In addition, our algorithm requires a small number of iterations (in most of our case studies only one iteration is required). Moreover, our approach leads to a smaller number of different blend

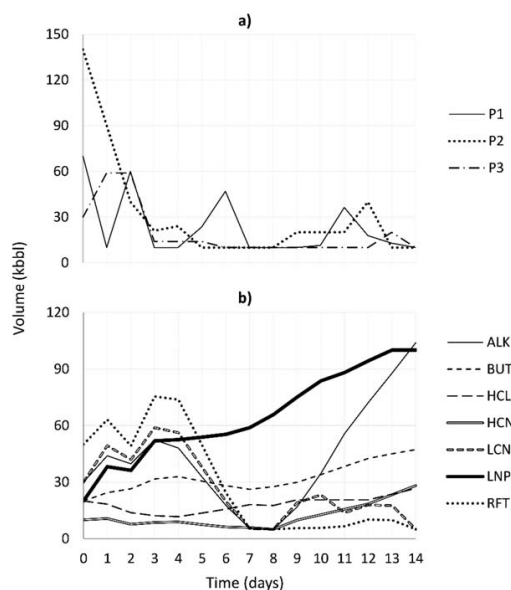


Figure 13. Case study 13—(a) Product and (b) component inventory profiles.

Approximate scheduling.

Table 3. Blend Planning—Comparison Between Solutions of Full-Space Models and the MPIP Planning Algorithm

Case Study ID	Demand Profile	# Pinch Points	# Blenders	Component Supply	Blend Cost ^a ($\times 10^3$ \$)	Full-Space MINLP (DICOPT)			Full-Space MILP (CPLEX)			MPIP Algorithm (NLP: IPOPT, MILP: CPLEX)		
						# Recipes ^b	Time ^c (s)	# Recipes ^d	Time ^c (s)	# Recipes ^d	Time ^c (s)	# Recipes ^d	Time ^c (s)	# Iterations
1	1	0	1	Constant	37,542.5	6	152.2	6	0.822	1	0.625	1		
2	2	1	1	Constant	38,309.8	6	240.5	5	0.752	2	0.778	1		
3	3	1	1	Constant	37,991.1	7	227.9	7	1.068	2	0.969	1		
4	3	1	2	Constant	37,991.1	8	192.9	7	1.754	2	0.938	1		
5	4	2	2	Constant	37,681.1	8	312.6	8	0.733	3	0.960	1		
6	5	2	2	Constant	37,324.3	9	258.5	8	0.969	6	10.634	4		
7	6	3	3	Constant	37,377.5	10	366.4	10	1.524	4	1.277	1		
8	1	0	1	Irregular	37,943.4	6	119.7	6	0.829	1	0.505	1		
9	2	1	1	Irregular	38,754.2	7	176.8	6	0.858	2	0.766	1		
10	3	1	1	Irregular	38,405.2	6	165.2	7	0.547	2	0.768	1		
11	3	1	2	Irregular	38,405.2	8	213.3	8	1.001	2	0.730	1		
12	4	2	2	Irregular	38,073.4	9	501.3	8	0.768	3	0.750	1		
13	5	2	2	Irregular	37,784.5	9	281.7	8	0.973	4	1.836	2		
14	6	3	3	Irregular	37,796.4	11	842.8	10	1.208	4	0.906	1		

^aAll approaches computed the same blend cost.

^bBlend recipe is considered different if the absolute change of composition percentage of any component is greater than 1%.

^cCPU time.

^dRepeated blend recipes are exactly the same.

Table 4. Approximate Scheduling—Comparison between Solutions of Full-Space Models and the MPIP Planning Algorithm

Case Study ID	Demand Profile ID	# Pinch Points	# Blenders	Supply Profile	Global Optimum Full-Space MILP (CPLEX)			Full-Space MINLP ANTIGONE		Full-Space MIOCP GLOMIO		MPIP Algorithm (IPOPT + CPLEX)	
					Total Cost ($\times 10^3$ \$)	Gap (%)	Best Lower Bound ($\times 10^3$ \$)	CPU Time (s)	Total Cost ($\times 10^3$ \$)	Total Cost ($\times 10^3$ \$)	Total Cost ($\times 10^3$ \$)	Total Cost ($\times 10^3$ \$)	CPU Time (s)
1	1	0	1	C	38,004	0.17	37,939	10,800	38,109	38,096	38,007	10,800	
2	2	1	1	C	38,782	0.00	38,782	3,264	38,891	38,827	38,896	83	
3	3	1	1	C	38,503	0.02	38,497	10,800	38,599	38,558	38,543	76	
4	3	1	2	C	38,529	0.11	38,486	10,800	38,703	38,598	38,563	10,800	
5	4	2	2	C	38,253	0.16	38,193	10,800	38,401	38,358	38,306	817	
6	5	2	2	C	37,916	0.09	37,883	10,800	38,060	37,951	37,981	66	
7	6	3	3	C	38,076	0.14	38,024	10,800	38,323	38,249	38,157	3,183	
8	1	0	1	I	38,387	0.12	38,342	10,800	38,498	38,512	38,423	647	
9	2	1	1	I	39,220	0.00	39,220	1,995	39,301	39,248	39,420	17	
10	3	1	1	I	38,917	0.09	38,882	10,800	38,998	38,992	39,004	21	
11	3	1	2	I	38,951	0.17	38,884	10,800	39,144	39,042	39,023	313	
12	4	2	2	I	38,621	0.12	38,577	10,800	38,788	38,745	38,698	649	
13	5	2	2	I	38,370	0.06	38,346	10,800	38,562	38,533	38,496	15	
14	6	3	3	I	38,497	0.13	38,447	10,800	38,738	38,710	38,596	9,017	
Average difference with respect to best lower bound ($\times 10^3$ \$)									186	137	115	—	

Stop at 10^{-5} with Gap Greater than 0.2%

All Runs Stopped at 10,800 s

Stop at 10^{-5} Gap or 10,800 s

Stop at 10^{-5} Gap or 10,800 s

Table 5. Approximate Scheduling—Results for MPIP Planning Algorithm

Case Study ID	MPIP Algorithm (IPOPT + CPLEX)			
	Total Cost ($\times 10^3$ \$)	Blend Cost ($\times 10^3$ \$)	Switching Cost ($\times 10^3$ \$)	CPU Time (s)
1	38,014.5	37,542.5	472.00	2809.3
2	38,901.8	38,309.8	592.00	18.5
3	38,550.1	37,991.1	559.00	22.5
4	38,569.1	37,991.1	578.00	1871.8
5	38,312.6	37,680.6	632.00	72.4
6	37,988.3	37,324.3	664.00	26.2
7	38,156.5	37,377.5	779.00	118.7
8	38,422.9	37,943.4	479.50	140.2
9	39,427.2	38,754.2	673.00	8.7
10	39,004.2	38,405.2	599.00	14.3
11	39,030.2	38,405.2	625.00	119.6
12	38,705.4	38,073.4	632.00	83.4
13	38,522.5	37,784.5	738.00	7.5
14	38,603.4	37,796.4	807.00	1104.1
Average difference of the total cost with respect to best lower bound ($\times 10^3$ \$)				122

Stopping criteria: 0.1% optimality gap or 10,800 s.

recipes per product compared to the solution of the corresponding full-space models (see Table 3). Significant reduction is seen in the cases with few number of pinch points and blenders; this is because the full-space models can have different recipes for the same product for different blenders in the same time period. In addition, repeated recipes from the solution of the full-space models may or may not be used in adjacent periods; thus, the number of recipe switching is equal or greater than the number of different recipes.

Approximate scheduling

The approximate schedules computed by the MPIP planning algorithm were compared with the full-space models. The termination criteria was 10,800 s (CPU time) or an optimality gap less than or equal to 0.001%. DICOPT was not used in this evaluation since it requires to solve the MILP subproblems to optimality to guarantee a local optimum; however, it was not possible to achieve that within the maximum allocated time of 3 h. Results from BARON are not shown because it was able to find feasible integer solutions only for some of our case studies and with optimality gaps greater than 3%.

Table 4 shows the results obtained by CPLEX, ANTIGONE, GloMIQO, and the MPIP planning algorithm. The results from full-space MILP were used to determine the global optimum and the best lower bound. In the majority of our case studies, MPIP algorithm computed better solutions than ANTIGONE and GloMIQO solvers and in much shorter execution times. All ANTIGONE and GloMIQO runs stopped at 10,800 s with

optimality gaps larger than 0.2%. Only in two test cases (no. 1 and no. 4) MPIP algorithm stopped at 10,800 s time limit. In both of these cases, the results from MPIP are slightly better than the results computed by GloMIQO and ANTIGONE. In test case no. 9, MPIP algorithm converged after 17 s, but the solution is 0.4% higher than GloMIQO.

Although the MILP model at the second level achieves optimality gaps smaller than or equal to 0.001%, it is noted that the solutions of the MPIP planning algorithm are higher than the best integer solution given by the full-space MILP. The reason for this difference is that the blend recipes define the feasible region at the second level and they are only being redefined to account for inventory feasibility, but there is no feedback that would cause the computation of new recipes that will minimize the number of product transitions in the blenders and swing tanks. Nevertheless, the difference between the MPIP algorithm solution and the true optimum will only be significant when the penalty for such transitions is much greater than the unit costs of the blend components. For our case studies, the average relative difference between the MPIP solution and the best lower bound is less than 0.33%. As another test of MPIP algorithm performance, we solved all test cases with 0.1% optimality gap at the second level. The execution times are even shorter and the function values are on average still closer to the best lower bound than those computed by ANTIGONE and GloMIQO (compare data in Tables 4 and 5).

Table 6 shows the number of equations, continuous variables, discrete variables, and non-zero elements in the full-

Table 6. Model Size Comparison

Model	# Equations	# Continuous Variables	# Discrete Variables	# Non-Zeros	# Nonlinear Terms
Full-space MINLP model (one blender)	5928	2753	609	16,646	1176
Full-space MINLP model (two blenders)	7906	3655	837	23,455	1960
Full-space MILP model (one blender)	5550	2375	609	15,554	0
Full-space MILP model (two blenders)	7600	3306	837	23,052	0
First level NLP model (MPIP algorithm, two L1-periods)	326	162	0	955	42
First level NLP model (MPIP algorithm, three L1-periods)	473	237	0	1416	63
Second level MILP model (MPIP algorithm, one blender, two fixed recipes)	4541	2375	609	11,017	0
Second level MILP model (MPIP algorithm, two blenders, two fixed recipes)	6003	3306	837	15,575	0

space models and our decomposition approach for blend planning case studies. Nonlinear aspects of the blending model are solved at the first level, where number of equations and variables are significantly smaller than in the full-space MINLP model. The second level model MILP is approximately the same size as the full-space models. Separation of the nonlinear blend optimization from linear model-based production planning enables MPIP algorithm to optimize blend plans or approximate blend schedules much faster than the corresponding full-space MINLP models.

Conclusions

We have presented a new inventory pinch based, two-level decomposition approach which (1) incorporates nonlinear blending models into integrated planning and approximate scheduling of gasoline blends, (2) includes blender switching and swing tankage management, (3) computes optimal blend plans with significantly smaller number of distinct blend recipes than the fine grid multiperiod MINLP models, (4) computes approximate schedules with similar or lower cost than those computed by global MINLP solvers used in this study, and (5) achieves much shorter execution times.

Rapid computation of optimal blend plans is accomplished by decomposing the blend planning optimization model into two levels: the first level handles the constraints related to operating conditions (e.g., quality constraints and blend recipe computation) by solving a NLP model, and the second level determines the actual production plan and allocation of swing tankage using optimal blend recipes from the first level in a MILP model, subject to availability of blend components, inventory storage limits, and minimum blend threshold constraints. These blend plans are then used to compute approximate blend schedules, which minimize total number of blend switches and product changeovers in the swing tanks.

Case studies with one, two, and three blenders have been presented. Our case studies have been constructed in such a way that after some transformations they can also be solved as an MILP model. That has enabled us to compute rapidly the global optimum and confirm that the blend plans computed by MPIP are also globally optimal. In 10 out of 14 case studies, MPIP algorithm finds a better solution for the approximate scheduling problem than ANTIGONE or GloMIQO and typically in much shorter execution times.

Our results show that the solutions computed by the MPIP planning algorithm are exactly the same as the optimum solutions computed by the corresponding full-space MINLP and MILP models when the objective function of the second level contains only variables that are aggregated at the first level (e.g., volumes to blend); and close-to-optimum solutions when the objective function of the second level contains variables that are not aggregated at the first level (e.g., switching variables) with penalty coefficients similar to the unit costs of blend components.

Part II of this article deals with detailed scheduling based on the approximate blend schedule computed at the second level of the MPIP algorithm.

Acknowledgments

Support by Ontario Research Foundation and McMaster Advanced Control Consortium is gratefully acknowledged. The authors thank Dr. Xiang Li (Queens University) for suggesting that they transform the example problems into MILP models in order to verify global optimality. Moreover, the

authors thank GAMS Corporation for granting them a license to use ANTIGONE and GloMIQO in their tests.

Notation

Subscripts

bc = refers to a variable or parameter related to the blend component tanks
 blend = refers to a variable or parameter related to the blenders
 comp = refers to a variable or parameter related to the transfer of volume between component tanks and blenders
 L1 = refers to a variable or parameter of the first level
 L2 = refers to a variable or parameter of the second level
 order = refers to a variable or parameter related to the product orders
 pool = refers to a variable or parameter related to the product pools
 pr = refers to a variable or parameter related to the individual product tanks
 trans = refers to a variable or parameter related to the transfer of volume between blenders and product tanks or pools

Superscripts

max = refers to a maximum value that a variable may have
 min = refers to a minimum value that a variable may have if different from zero
 start = refers to the initial value at the beginning of the planning horizon that a variable may have
 target = refers to a target value for a variable

Parameters

$Cost_{bc}(i)$ = cost of blend component i
 $D_{pr}^{max}(j)$ = maximum delivery rate of tank j
 $Demand(o)$ = demand of order o for the complete scheduling horizon
 $Demand_{pool,L1}(p,k)$ = aggregated demand of product p in L1-period k
 $Demand_{pr,L2}(p,m)$ = demand of product p in L2-period m
 $dt_{L2}(o,m)$ = time available to deliver order o during L2-period m
 $F_{bc}(i,\alpha)$ = supply flow rate of blend component i during time interval α
 $F_{blend}^{max}(bl)$ = maximum blending rate of blender bl
 $F_{blend}^{min}(bl)$ = minimum blending rate of blender bl
 H = length of the planning horizon
 $it_{blend}^{min}(p,bl)$ = minimum idle time required by blender bl before processing product p
 $np(bl)$ = number of products that can be produced in a L2-period in blender bl
 $Penalty_{bc,L1}$ = penalty for the inventory slack variables of blend component tanks
 $Penalty_{pool,L1}$ = penalty for the inventory slack variables of product pools
 $Penalty_{bc,L2}(m)$ = penalty for the inventory slack variables of component i in L2-period m
 $Penalty_{pool,L2}(m)$ = penalty for the inventory slack variables of product pool p in L2-period m
 $Penalty_{pr,L2}(m)$ = penalty for the inventory slack variables of product tank j in L2-period m
 $Penalty_{BR_{L2}}(bl)$ = penalty for a blend instance processed in blender bl during a L2-period m
 $Penalty_{BS_{L2}}$ = penalty for a product transition in a blender at the second level
 $Penalty_{TS}(j)$ = penalty for a product transition in a swing tank at the second level
 $Q_{bc}(i,e,k)$ = quality e of blend component i during L1-period k
 $Q_{pr}^{max}(p,e)$ = maximum requirement of quality e in grade p
 $Q_{pr}^{min}(p,e)$ = minimum requirement of quality e in grade p
 $r_{pr}^{max}(i,p)$ = maximum composition specification of product p regarding blend component i
 $r_{pr}^{min}(i,p)$ = minimum composition specification of product p regarding blend component i
 $t_{bc,L1}(\alpha,k)$ = duration of time interval where blend component supply flow rate α occurs in L1-period k
 $t_{bc,L2}(\alpha,m)$ = duration of time interval where blend component supply flow rate α occurs in L2-period m
 $it_{blend}^{min}(p,bl)$ = minimum running time required by blender bl when processing product p

$t_{L1}(k)$ = duration of L1-period k
 $t_{L2}(m)$ = duration of L2-period m
 $u_{L2}^{\text{start}}(j,p)$ = product p stored in tank j at the beginning of the planning horizon
 $u_{ofL2}(o,m)$ = parameter which value is 1 if order o may be delivered during L2-period m (i.e., delivery window of order o spans L2-period m) and 0 otherwise
 $V_{bc}^{\text{max}}(i)$ = maximum holdup of tank with blend component i
 $V_{bc}^{\text{min}}(i)$ = minimum holdup of tank with blend component i
 $V_{bc}^{\text{start}}(j,p)$ = volume of blend component i stored at the beginning of the planning horizon
 $V_{\text{blend,L1}}(p,k)$ = volume of product p to blend in L1-period k
 $V_{\text{pool}}^{\text{max}}(p)$ = maximum holdup of product pool p
 $V_{\text{pool}}^{\text{min}}(p)$ = minimum holdup of product pool p
 $V_{\text{pool}}^{\text{start}}(p)$ = total volume of product p stored at the beginning of the planning horizon
 $V_{\text{pool,L1}}^{\text{target}}(p,k)$ = target inventory for product pool p in L1-period k
 $V_{pr}^{\text{max}}(j)$ = maximum holdup of tank j
 $V_{pr}^{\text{min}}(j)$ = minimum holdup of tank j
 $V_{pr}^{\text{start}}(j)$ = volume stored in tank j at the beginning of the planning horizon
 $V_{\text{MINblend}}(p,bl)$ = minimum volume allowed to blend of product p in blender bl during each L2-period

$t_{\text{blend,L2}}(p,bl,m)$ = estimated time to process product p in blender bl in L2-period m
 $V_{bc,L1}(i,k)$ = volume stored in component tank i at the end of L1-period k
 $V_{bc,L2}(i,m)$ = volume stored in component tank i at the end of L2-period m
 $V_{\text{blend,L2}}(p,bl,m)$ = volume of product p to process in blender bl in L2-period m
 $V_{\text{comp,L1}}(i,p,k)$ = volume of blend component i into product p in L1-period k
 $V_{\text{comp,L2}}(i,p,bl,m)$ = volume of blend component i into product p in blender bl in L2-period m
 $V_{\text{pool,L1}}(p,k)$ = volume stored in product pool p at the end of L1-period k
 $V_{\text{pool,L2}}(p,m)$ = volume stored in product pool p at the end of L2-period m
 $V_{pr,L1}(j,k)$ = volume stored in product tank j at the end of L1-period k
 $V_{pr,L2}(j,m)$ = volume stored in product tank j at the end of L2-period m
 $V_{\text{trans,L2}}(j,p,bl,m)$ = volume transferred of product p from blender bl to tank j in L2-period m
 $xe_{L2}(p,bl,m)$ = 0–1 continuous variable that indicates if a state transition has occurred in blender bl at the beginning of L2-period m
 Z_{L1} = objective function value at the first level
 Z_{L2} = total cost at the second level
 Z_{L2}^{feas} = objective function value at the second level, blend planning
 Z_{L2}^{opt} = objective function value at the second level, approximate scheduling

Binary variables

$u_{L2}(j,p,m)$ = binary variable that indicates if tank j is storing product p in L2-period m
 $ue_{L2}(j,m)$ = binary variable that indicates if there is a product transition in tank j at the beginning of L2-period m
 $v_{L2}(j,p,bl,m)$ = binary variable that indicates if tank j is receiving product p from blender bl in L2-period m
 $y_{L2}(p,bl,g)$ = binary variable that indicates if product p occupies slot g in order to estimate the production sequence of blender bl at the second level
 $x_{L2}(p,bl,m)$ = binary variable that indicates if product p is processed in blender bl in L2-period m

Continuous variables

Blend = Cost_{L1} = total blend cost at the first level
 Blend = Cost_{L2} = total blend cost at the second level
 $\text{Deliver}_{pr,L2}(j,m)$ = volume of tank j shipped/lifted at the end of L2-period m
 $\text{Deliver}_{\text{pool,L2}}(p,m)$ = volume of product p shipped/lifted at the end of L2-period m
 $of_{L2}(o,m)$ = fraction of order o to be delivered during L2-period m
 $Q_{pr,L1}(p,e,k)$ = quality level of property e of product p in L1-period k
 $r(i,p,k)$ = volume of blend component i into product p in L1-period k (it becomes a parameter in the second level model)
 $S_{bc,L1}^+(i,k)$ = positive inventory slack variable of blend component i at L1-period k
 $S_{bc,L1}^-(i,k)$ = negative inventory slack variable of blend component i at L1-period k
 $S_{bc,L2}^+(i,m)$ = positive inventory slack variable of blend component i at L2-period m
 $S_{bc,L2}^-(i,m)$ = negative inventory slack variable of blend component i at L2-period m
 $S_{\text{pool,L1}}^+(p,k)$ = positive inventory slack variable of product pool p at L1-period k
 $S_{\text{pool,L1}}^-(p,k)$ = negative inventory slack variable of product pool p at L1-period k
 $S_{\text{pool,L2}}^+(p,m)$ = positive inventory slack variable of product pool p at L2-period m
 $S_{\text{pool,L2}}^-(p,m)$ = negative inventory slack variable of product pool p at L2-period m
 $S_{pr,L1}^+(i,k)$ = positive inventory slack variable of product tank j at L1-period k
 $S_{pr,L1}^-(i,k)$ = negative inventory slack variable of product tank j at L1-period k
 $S_{pr,L2}^+(i,m)$ = positive inventory slack variable of product tank j at L2-period m
 $S_{pr,L2}^-(i,m)$ = negative inventory slack variable of product tank j at L2-period m

Literature Cited

- Shah NK, Li Z, Ierapetritou MG. Petroleum refining operations: key issues, advances, and opportunities. *Ind Eng Chem Res.* 2011;50:1161–1170.
- Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Ind Eng Chem Res.* 2003;42:825–835.
- Sing A, Forbes JF, Vermeer PJ, Woo SS. Model-based real-time optimization of automotive gasoline blending operations. *J Process Control.* 2000;10:43–58.
- Kelly JD. Formulating production planning models. *Chem Eng Prog.* 2004;100:43–50.
- Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. *AIChE J.* 2010;56:441–465.
- Li J, Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res.* 2011;50:9156–9174.
- Mendez CA, Grossman IE, Harjunkoski I, Kabore P. A simultaneous optimization approach for off-line blending and scheduling of oil refinery operations. *Comput Chem Eng.* 2006;30:614–634.
- Maravelias C, Sung C. Integration of production planning and scheduling: overview, challenges and opportunities. *Comput Chem Eng.* 2009;33:1919–1930.
- Bitran GR, Hax AC. On the design of hierarchical production planning systems. *Decis Sci.* 1977;8(1):28–55.
- Nam S, Logendran R. Aggregate production planning—a survey of models and methodologies. *Eur J Oper Res.* 1992;61:255–272.
- Axsater S, Jonsson H. Aggregation and disaggregation in production planning. *Eur J Oper Res.* 1984;17:338–350.
- Verderame PM, Floudas CA. Operational planning framework for multisite production and distribution networks. *Comput Chem Eng.* 2009;33:1036–1050.
- Timpe CH, Kallrath J. Optimal operational planning in large multisite production networks. *Eur J Oper Res.* 2000;126:422–435.
- Thakral A, Mahalec V. Composite planning and scheduling algorithm addressing intra-period infeasibilities of gasoline blend planning models. *Can J Chem Eng.* 2013;91:1244–1255.
- Singhvi A, Shenoy UV. Aggregate planning in supply chains by pinch analysis. *Chem Eng Res Des.* 2002;80:597–605.
- Singhvi A, Madhavan KP, Shenoy UV. Pinch analysis for aggregate production planning in supply chains. *Comput Chem Eng.* 2004;28:993–999.
- Ludwig J, Treitz M, Rentz O, Geldermann J. Production planning by pinch analysis for biomass use in dynamic and seasonal markets. *Int J Prod Res.* 2009;47(8):2079–2090.

18. Foo DCY, Ooi MBL, Tan RR, Tan JS. A heuristic-based algebraic targeting technique for aggregate planning in supply chains. *Comput Chem Eng.* 2008;32:2217–2232.
19. Castillo PAC, Kelly JD, Mahalec V. Inventory pinch algorithm for gasoline blend planning. *AIChE J.* 2013;59(10):3748–3766.
20. Misener R, Floudas CA. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *J Glob Optim.* In press.
21. Misener R, Floudas CA. Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCP) through piecewise-linear and edge-concave relaxations. *Math Program.* 2012;136:155–182.
22. Misener R, Floudas CA. GloMIQO: global mixed-integer quadratic optimizer. *J Glob Optim.* 2013;57(1):3–50.
23. Li Z, Ierapetritou MG. Integrated production planning and scheduling using a decomposition framework. *Chem Eng Sci.* 2009;64:3585–3597.

Appendix A: Equivalence of the Full-Space MINLP/MILP and the MPIP Planning Models

In this appendix, we will show how the first and second level models described in the article are can be obtained from the full-space model. The derivation presented here uses reasoning similar to Li and Ierapetritou.²³ Let us consider the following full-space optimization model (A1), also known as a single level model

$$\begin{aligned} \min_x \quad & cx \\ \text{s.t.} \quad & x \in X \\ & x \geq 0 \end{aligned} \quad (\text{A1})$$

x represents all the variables (continuous and integer), and X represents the feasible space given by all the sets of constraints on x (linear and nonlinear). Therefore, (A1) may represent a full-space MINLP or MILP model. Let us consider that we have a planning horizon originally discretized in M time periods ($M = \{m\}$, i.e., the L2-periods). Suppose we can aggregate over time some of the variables in the model. This aggregation is carried out over K nonoverlapping time intervals that span the entire planning horizon ($K = \{k\}$, i.e., the L1-periods). Let us define x in terms of variables that can be aggregated ($x_2(m)$), variables that may have a fixed value during an aggregated time interval ($y_2(m)$), variables that cannot be aggregated ($z_2(m)$), the aggregated variables ($x_1(k)$), and variables representing the fixed property/condition during an aggregated interval ($y_1(k)$)

$$x \equiv \{x_1(k), y_1(k), x_2(m), y_2(m), z_2(m)\} \quad (\text{A2})$$

By definition

$$x_1(k) = \sum_{m \in k} x_2(m) \quad \forall k \quad (\text{A3})$$

$$y_1(k) = y_2(m) \quad \forall (m, k) \in \text{MK}(m, k) \quad (\text{A4})$$

where $\text{MK}(m, k)$ is the set that indicates which adjacent time periods m are aggregated as the interval k . It is noted that $x_1(k)$ can only represent continuous variables, whereas $y_1(k)$ can represent continuous and discrete variables. In terms of the gasoline blend planning problem, $x_1(k)$ represents the aggregated production volumes at the first level, and $y_1(k)$ represents the blend recipes and the quality values of the blends. Let us rewrite model (A1) as (A5), which is still the original full-space model

$$\begin{aligned} \min_{x_2, y_2, z_2} \quad & \sum_m [c_1 x_2(m) + c_2 y_2(m) + c_3 z_2(m)] \\ \text{s.t.} \quad & x_2(m), y_2(m), z_2(m) \in X \quad \forall m \\ & x_2(m), y_2(m), z_2(m) \geq 0 \quad \forall m \end{aligned} \quad (\text{A5})$$

Model (A6) is obtained by substituting the aggregated variables into Model (A5). The constraints can now be divided into two sets. One set will contain all the equations regarding the $x_1(k)$ and

$y_1(k)$ variables, and the associated feasible space is denoted as X_1 . The second set of constraints will consist of all the equations containing the $x_2(m)$, $y_2(m)$, and $z_2(m)$ variables, with X_2 as the associated feasible space

$$\begin{aligned} \min_{x_1, y_1, z_2} \quad & \sum_k [c_1 x_1(k) + c_2 i(k) y_1(k)] + \sum_m c_3 z_2(m) \\ \text{s.t.} \quad & x_1(k), y_1(k) \in X_1 \quad \forall k \\ & \sum_{m \in k} x_2(m) = x_1(k) \quad \forall k \\ & y_2(m) = y_1(k) \quad \forall (m, k) \in \text{MK}(m, k) \\ & x_2(m), y_2(m), z_2(m) \in X_2 \quad \forall m \\ & x_1(k), y_1(k), x_2(m), y_2(m), z_2(m) \geq 0 \quad \forall k, m \end{aligned} \quad (\text{A6})$$

Model (A6) can be written as a two-level problem given by (A7), which is analogous to our first level model plus the second level approximate scheduling model of the MPIP planning algorithm

$$\begin{aligned} \min_{x_1, y_1} \quad & \sum_k [c_1 x_1(k) + c_2 i(k) y_1(k)] + f_2 \\ \text{s.t.} \quad & x_1(k), y_1(k) \in X_1 \quad \forall k \\ & x_1(k), y_1(k) \geq 0 \quad \forall k \end{aligned}$$

where

$$\begin{aligned} f_2 = \min \quad & \sum_m c_3 z_2(m) \\ \text{s.t.} \quad & \sum_{m \in k} x_2(m) = x_1(k) \quad \forall k \\ & y_2(m) = y_1(k) \quad \forall (m, k) \in \text{MK}(m, k) \\ & x_2(m), y_2(m), z_2(m) \in X_2 \quad \forall m \\ & x_2(m), y_2(m), z_2(m) \geq 0 \quad \forall m \end{aligned} \quad (\text{A7})$$

It can be noted that X_1 is a relaxation of X because not all the original equations are considered in it. X_2 is a contraction of X because all the original equations are considered (except those corresponding to $y_2(m)$, but because $y_1(k) \in X_1$, then $y_2(m)$ is feasible) plus some new restrictions [i.e., (A3) and (A4)]. Therefore, the feasible space X_2 depends on the values of $x_1(k)$ and $y_1(k)$. It is possible that X_2 will include the optimum x^* from the original X when the number of aggregated intervals K is greater than some minimum number k^* ; in other words, when $K \geq k^*$ the values of $x_1(k)$ and $y_1(k)$ are such that the feasible space X_2 will contain the optimum x^* from X . Finding this minimum number k^* is the objective of the MPIP planning algorithm.

If we assume that the objective function only contains aggregated variables, that is, $c_3 = 0$, Model (A7) can be expressed as Model (A8)

$$\begin{aligned} \min_{x_1, y_1} \quad & \sum_k [c_1 x_1(k) + c_2 i(k) y_1(k)] + f_2 \\ \text{s.t.} \quad & x_1(k), y_1(k) \in X_1 \quad \forall k \\ & x_1(k), y_1(k) \geq 0 \quad \forall k \end{aligned}$$

where

$$\begin{aligned} f_2 = 0 \\ \text{s.t.} \quad & \sum_{m \in k} x_2(m) = x_1(k) \quad \forall k \\ & y_2(m) = y_1(k) \quad \forall (m, k) \in \text{MK}(m, k) \\ & x_2(m), y_2(m), z_2(m) \in X_2 \quad \forall m \\ & x_2(m), y_2(m), z_2(m) \geq 0 \quad \forall m \end{aligned} \quad (\text{A8})$$

Therefore, it can be seen that the second level model is just a feasibility problem. Model (A8) is analogous to the first level plus the second level blend planning model of our MPIP planning algorithm. We can include slack variables ($s_1(k)$ and $s_2(m)$) at each level of the model to obtain a numerical feasible solution even when some constraints are violated. P represents the penalty coefficients. A physically feasible solution will have all the slack variables equal to zero. From Model (A8), it can be seen that the blend cost value is fixed according to $x_1(k)$ and $y_1(k)$; however, we observed that the inclusion of the blend cost term in our MPIP second level model formulation speeded up the solution of the feasibility problem.

We can define the lower and upper bounds (LB and UB, respectively) as follows

$$\text{LB} = \arg \left\{ \min_{x_1, y_1 \in X_1} \sum_k [c_1 x_1(k) + c_2 y_1(k) + P s_1(k)] \right\} \quad (\text{A9})$$

$$\text{UB} = \arg \left\{ \min_{x_2, y_2 \in X_2} \sum_m [c_1 x_2(m) + c_2 y_2(m) + P s_2(m)] \right\} \quad (\text{A10})$$

The MPIP algorithm finds the minimum number of aggregated time intervals, k^* , by subdividing the intervals at the first level at each iteration (iter) if $\text{UB}^{(\text{iter})} > \text{LB}^{(\text{iter})}$. For blend planning (Model A8), the MPIP algorithm computes optimal solutions since the pinch points define the minimum production target and the rules to eliminate infeasibilities preblend as low as possible. For approximate scheduling (Model A7), current solutions may only be considered near-optimal as long as the coefficients denoted by c_3 are similar or smaller in magnitude to coefficients c_1 and c_2 . This is due to term f_2 not being estimated at the first level by our algorithm.

Manuscript received Oct. 24, 2013, and revision received Feb. 4, 2014.

Chapter 3: Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part II: Gasoline Blend Scheduling

This chapter has been published in the AIChE Journal. Complete citation:

Castillo Castillo, P. A., & Mahalec, V. (2014). Inventory pinch based, multiscale models for integrated planning and scheduling-part II: Gasoline blend scheduling. *AIChE Journal*, 60(7), 2475–2497. *Wiley Online Library*, doi: 10.1002/aic.14444

Permission from © American Institute of Chemical Engineers.

In Chapter 3, the multiperiod inventory pinch (MPIP) algorithm is introduced for blend scheduling problems. In this case, MPIP decomposes the original problem into three levels. The 1st and 2nd levels are constructed based on the methodology presented in Chapter 2, with some modifications to the 2nd level MILP model to include a few scheduling decisions. The 3rd level is a multiperiod MILP model (with original number of periods defined by the scheduler) with fixed blend recipes. All three levels are formulated using discrete-time representation. Due to their large size, the 3rd level model is solved using a rolling horizon strategy.

Inventory Pinch Based, Multiscale Models for Integrated Planning and Scheduling-Part II: Gasoline Blend Scheduling

Pedro A. Castillo Castillo and Vladimir Mahalec

Dept. of Chemical Engineering, McMaster University, Hamilton, ON, Canada L8S 4L8

DOI 10.1002/aic.14444

Published online April 1, 2014 in Wiley Online Library (wileyonlinelibrary.com)

Integration of planning and scheduling optimizes simultaneous decisions at both levels, thereby leading to more efficient operation. A three-level discrete-time algorithm which uses nonlinear models and integrates planning and detailed scheduling is introduced: first level optimizes nonlinear blend models via multiperiod nonlinear programming (NLP), where period boundaries are initially determined by the inventory pinch points; second level uses fixed recipes (from the first level) in a multiperiod mixed-integer linear program to determine first an optimal production plan and then to optimize an approximate schedule which minimizes the total number of switches in blenders and swing tanks; third level computes detailed schedules that adhere to inventory constraints computed in the approximate schedule. If inventory infeasibilities appear at the second or the third level, the first-level periods are subdivided and blend recipes are reoptimized. Algorithm finds the same or better solutions and is substantially faster than previously published full-space continuous-time model. © 2014 American Institute of Chemical Engineers AIChE J, 60: 2475–2497, 2014

Keywords: gasoline blend planning, scheduling, inventory pinch, recipe optimization, minimum number of recipes, multi-scale models

Introduction

In an oil refinery, gasoline can account for 60–70% of the total profit^{1–3}; therefore, minimization of blending operation costs represents a huge opportunity to increase profit margins. Oil refineries have to deal with unsteady product demands and crude prices, as well as continuously stricter environmental regulations.⁴ In such a context, computational supply chain optimization tools based on mathematical models have become very important for companies to plan and schedule their operations in the best way possible with the objective to reduce costs and maximize revenues.⁵ Market opportunities in the short term should be exploited without compromising the objectives in the long term.⁶ Optimization of blending operations (as the production of gasoline in an oil refinery) involves the fulfillment of product quality specifications and demand requirements at the minimum cost, subject to raw materials availability and production and storage capacity limits. Kelly⁷ pointed out that quality and quantity details are not the only elements comprising the optimization problem of blending systems, and he described several logistical details that, if incorporated into the blend scheduling problem, will provide a more accurate production schedule. Some of the logic constraints listed by Kelly⁷ are the following: (1) a flow (i.e., the total volume pumped/processed or a flow/production rate) must be between its lower and upper bounds, (2) a flow must be equal across contiguous time periods if a specific task has not finished, (3) processing units can only execute one task at a time, (4) a tank

can only store a new material when its holdup is less than or equal to the quantity specified (this type of tanks are called swing tanks as they can switch to a different material service), (5) minimum and maximum bounds on the running time (up-time) and idle time (down-time) of a unit, task, or operation mode, may be specified, (6) the use of sequence-dependent and -independent changeover down-times may be considered, and (7) an order (internal or external) cannot be fulfilled outside its specified time window, but material flow may be specified to be constant or allowed to be intermittent within the window.

Many authors have worked on the integration of production planning and scheduling optimization problems, either developing easier-to-converge models or designing more efficient algorithms to solve them. Maravelias and Sung⁸ reviewed the opportunities and challenges of integrating the planning and scheduling levels, pointing at the importance of developing more computationally effective models for complex process systems, improving decomposition and iterative algorithms, and the possibility to solve the scheduling problem more efficiently by developing hybrid methods using different solution techniques. A broad classification of scheduling formulation approaches is also found in Maravelias,⁹ Maravelias and Sung,⁸ and Mendez et al.¹⁰ Gasoline blending falls into the category of network-based formulations due to the continuous nature of the process. With respect to the time representation used, the formulated problems can be classified as discrete-, continuous-, or mixed-time models. In the discrete-time models, the scheduling horizon is divided in a given number of time periods whose duration is known *a priori*, whereas in the continuous-time models the length of these periods is not known in advance (for this reason, the word “time slot” is preferred when referring to this type

Corresponding concerning this article should be addressed to V. Mahalec at mahalec@mcmaster.ca.

of time intervals). An in-depth review of advantages and disadvantages of discrete- and continuous-time formulations can be found in Floudas and Lin¹¹ and Sundaramoorthy and Maravelias.¹² Joly and Pinto¹³ developed a discrete-time mixed-integer linear programming (MILP) model for the scheduling of fuel oil and asphalt production, and they pointed out that, although continuous-time formulations may decrease significantly the combinatorial feature of a model, discrete-time models may still be a good option because (1) the resource constraints are easier to handle (e.g., products between flow rates and time intervals are linear) and (2) discrete-time models provide tight formulations in general. In mixed-time formulations, the time grid is fixed but the durations of the tasks are variable.

In principle, to solve an integrated planning and scheduling problem it is enough to write a discrete- or continuous-time model, that is, the full-space model, and solve it; however, for real-life, large-scale problems, this will lead to intractable mixed-integer nonlinear programming (MINLP) or MILP models. Given their large-scale combinatorial nature, scheduling problems are at least nondeterministic polynomial time (NP) complete.⁶ In the last decade, researchers have been working on improving full-space model formulations in order to avoid prohibited execution times.

Jia and Ierapetritou¹ solved simultaneously the gasoline blend scheduling and distribution problem. They presented a continuous-time event-based MILP model for the scheduling problem. The model includes multipurpose product tanks (tank switching), delivery of the same order from multiple product tanks, and one product tank delivering multiple orders. A set of preferred blend recipes is given (i.e., blend recipes are not optimized). Their largest problem (one blender, four products, 11 product tanks, nine blend components, 45 orders, and a scheduling horizon of 8 days) was solved to proven optimality in 5 CPU hours.

Mendez et al.³ introduced an iterative algorithm to optimize blend recipes and schedule blending operations. Nonlinear quality constraints are modeled as linear constraints by using correction factors. At the end of each iteration, these correction factors for the product properties are calculated according to the blend recipes computed. The algorithm stops when the correction factors converge and the products properties fulfill the specifications. Minimum blend run constraints and multipurpose tanks are features not included in the model. Due to the assumption made in their case studies that each blender produces only one particular gasoline grade (i.e., the sequencing problem is avoided), their computational times are very small (less than 2 s).

Li et al.² presented a continuous-time slot-based MILP model that uses process slots. This model includes the blend recipe optimization, inventory constraints, blender capacity constraints, and delivery scheduling for the demand orders. Blend indices are used instead of the actual quality properties in order to avoid nonlinear constraints. Their model also includes parallel nonidentical blenders, multipurpose tanks, and other attributes and constraints found in industrial practice. After the model is solved, a schedule adjustment step is required to ensure that each blend run has a constant blending rate. Although their formulation incorporates many details of the industrial systems, computational times of more than 20 h were required to solve examples for a blending system of significant size (e.g., 3 blenders, 9 and 11 component and product storage tanks, respectively) and a

scheduling horizon of 8 days; nevertheless, their solutions were better than those provided by DICOPT and BARON solving the corresponding MINLP model in the same time.

Li and Karimi⁴ replaced process slots with unit slots and expanded the model by Li et al.² to include blender setup times, limited inventory of components, and simultaneous receipt/delivery by the product tanks. Due to the reduction in the number of discrete variables when using unit slots instead of process slots, computational times improved significantly for small- and medium-size problems; however, large-scale problems (e.g., 2–3 blenders, 5 products, 9 components, 9 properties, 11 product tanks, 35–45 orders, and a planning horizon of 8 days) used all the allocated CPU time (46,800–118,800 s, depending on the problem) as the solution did not meet the stopping criteria.

Decomposition techniques have been used to solve more efficiently the integrated planning and scheduling problem, as well as the scheduling problem itself. There is usually a trade off between shorter execution times and the quality of the solution obtained depending of the decomposition method used.

Bassett et al.¹⁴ reviewed several time-based decomposition approaches to solve the scheduling problem; these decompositions are based on subdividing the scheduling horizon in smaller intervals, solving the corresponding subproblems in a specific sequence, and applying heuristics methods to combine the solutions.

Elkamel et al.¹⁵ presented a spatial and a temporal decomposition to schedule batch processes in a general chemical plant. The spatial decomposition is based on grouping units which perform similar tasks and their corresponding orders. In the temporal decomposition, the product orders are grouped according to their due dates, and the last due date of each group delineates the time where the scheduling horizon is subdivided. Global optimality is only guaranteed if all the subproblems are independent.

Munawar and Gudi¹⁶ proposed a three-level hierarchical approach to integrate the planning and scheduling decisions in the multistage hybrid flowshop problem for a single facility. The first level consists of the midterm planning model and its solution provides the production targets for each time period. Stopping losses are assumed at this level and the objective function maximizes production in the initial periods in order to have production capacity available in later periods to handle unexpected events (e.g., demand variations, machine breakdowns etc). At the second level, the schedule is computed using a continuous-time model (which is solved sequentially for each of the planning periods) that maximizes profit and penalizes product changeover and inventory costs. Actual stopping losses are calculated at this level, and inventory upper bounds are overestimated based on heuristics or previous process knowledge. The third level determines the detailed product-to-tank assignments using a heuristic algorithm to find the minimum number of tanks required to manage the inventory levels provided by the second level. This algorithm is based on slicing the inventory profiles along the scheduling horizon at the given capacity of the individual tanks, and generating subprofiles to determine the points in time that a tank is free to be reused.

Li and Ierapetritou¹⁷ developed a bilevel decomposition algorithm to solve separately the production planning and scheduling levels. The planning level is modeled using discrete-time representation while the scheduling subproblems (one per planning period) are formulated as continuous-time

models. The algorithm aims to close the difference between the objective function of both levels. At the planning level, an underestimation term of the production costs associated with the scheduling level is included and computed through Lagrangian relaxation. At the end of each iteration, the algorithm contracts the bounds of the planning decision variables.

Mouret et al.¹⁸ introduced a new algorithm to solve a MINLP model for refinery planning and scheduling of crude-oil operations using Lagrangian decomposition, as well as a new hybrid algorithm to solve the associated dual problem which combines subgradient and cutting plane methods at different steps. The authors pointed out that, in this case, the planning and scheduling problems are only linked by the crude distillation unit (CDU) feedstock quantities; that is, the planning model is not an aggregated formulation of the scheduling problem. For that reason, a spatial Lagrangian decomposition is a better option to solve the problem than a hierarchical approach.

For gasoline blend planning and scheduling, Glismann and Gruhn¹⁵ used a two-level approach: at the top level, a discrete-time NLP model computes blend recipes and, at the lower level, a discrete-time MILP model solves the short-term scheduling problem. In this approach, the time periods of the NLP model are defined by product liftings and other specific planning priorities while the scheduling MILP model time periods are defined to be 2-hours long. If a feasible solution cannot be found at the lower level, or if deviations from the goals determined at the top level cannot be accepted, blend recipes are recomputed by NLP model but this time including the information from MILP solution through the addition of constraints regarding the blend components consumption (this step is not clearly described in their article). The new blend recipes can be chosen as alternatives to the previous ones at the MILP scheduling model which contains constraints to enforce a minimum running time for a single recipe on a blender.

The work presented in this article introduces a new method to solve the gasoline blend planning and scheduling problem, using the inventory pinch concept to reduce the number of different blend recipes. It is based on the gasoline blend planning decomposition shown in Part I of this article. We use a three-level decomposition:

1. At the first level, blend recipes are optimized by solving a discrete-time multiperiod NLP model. The boundaries of the time periods in this NLP model are initially given by the inventory pinch points and break points (if applicable) in the components' qualities or unit costs.

2. At the second level, a blend plan using the recipes from the first level is computed. The blend plan defines the swing tanks allocation to each product and the volumes to produce in each blender in each second-level time period along the horizon. A discrete-time multiperiod MILP model is solved for the entire horizon in two phases:

- a. Blend planning. The objective function minimizes the blend cost and inventory infeasibilities. Hence, this phase uses the blend recipes from the first level to compute an optimal blend plan.

- b. Approximate scheduling. The objective function minimizes the number of blend runs and the number of product transitions in the blenders and swing tanks. It is assumed that the length of the time periods at the second level is such that a swing tank can only be used for one product during any second-level time period. As the blend recipes and inventory levels from the first level have been proven feasible in the previous phase, they are fixed and the blend cost is not included in the objective function.

3. At the third level, scheduling of the blending operations is carried out based on the decisions from the second level. Production and delivery rates, as well as start and end times of the tasks are computed. Product-tank allocation is fixed as computed at the second level. At the third level, the scheduling horizon is divided in several time intervals, and each one is solved using a discrete-time multiperiod MILP. These small subintervals are solved in two sequences: a forward sequence that computes an initial solution considering the initial conditions of the system, and a reverse sequence that merges blend runs, if possible, to obtain a final production schedule with smaller number of switches. Each sequence is solved in two phases:

- a. Feasibility phase. The objective function minimizes the blend cost and inventory infeasibilities. During the forward sequence, this phase determines if the blend recipes from the first level and the blend plan from the second level can provide a feasible schedule; while during the reverse sequence, this phase determines the minimum number of blend runs that can be achieved.

- b. Optimization phase. The objective function minimizes the number of blend runs, penalizes long blend runs, variations in the delivery rates, late deliveries, and variations in the destination tank for the product of a blend run.

If the solutions at the second level or at the third level forward sequence contain slack variables with nonzero values, the current set of blend recipes from the first level leads to infeasible solutions. In such a case, we subdivide the corresponding period at the first-level NLP model and resolve all levels. Hence, we increase the number of time periods (and the number of corresponding recipes) only when such a change is required to ensure the optimality and feasibility of the solution.

The contents of this article are organized as follows. We start with the problem description, and then, the proposed solution approach is explained. The mathematical models and solution algorithm are presented next, followed by the numerical case studies and by the comparison of our solutions with those from the literature. Finally, we conclude with the discussion of the results, summary of the algorithm performance, and outline of the future work.

Problem Statement

The integrated gasoline blend planning and scheduling problem addressed in this work is stated as follows:

Given

1. A scheduling horizon $[0, H]$;
2. A set of blend components and profiles of their quality levels along the horizon;
3. A set of tanks to store the blend components, their initial inventories, limits on their holdups, and the flow profiles of feeds into the tanks;
4. A set of products and their quality and composition specification limits;
5. A set of blenders, the products that each blender can process, minimum running times of these blenders for a specific product, and limits on their blending rates;
6. A set of tanks to store the products, the products that each tank can store, limits on their holdups, the products and inventories at time zero, and their maximum delivery rates;
7. A set of orders, their constituent products, amounts, and delivery time windows; and
8. Unit cost of the blend components.

Determine

1. The blend recipes (i.e., the volume fractions of the blend components that compound one unit of each product);
2. The blenders that each component tank should feed over time, and their feed rates;
3. The products that each blender should produce over time, and their production rates;
4. The products that each product tank should receive over time, from which blender, and at what flow rates;
5. The orders that each product tank should deliver over time, their amounts, and delivery rates; and
6. The inventory profiles of component and product tanks.

Minimizing

The operating cost that includes the blended materials cost and the cost associated with the number of blend runs and product transitions in the blenders and storage tanks.

Subject to the following constraints

1. A blender can process, at most, one product at any time. Once it begins processing a product, it must operate for some minimum time before it can switch to another product.
2. A blender can feed, at most, one product tank at any time (industrial practice).

Assuming

1. Flow rate profile of each component from the upstream process is piecewise constant;
2. Component quality profile is also piecewise constant;
3. There is only one tank for a given blend component;
4. Mixing in each blender is perfect;
5. Changeover times between products are negligible for product tanks;
6. Changeover times between product runs on blenders are product-dependent but sequence-independent;
7. Each order involves only one product (one original order involving different products can be broken into orders of each specific product); and
8. Each order is completed during the scheduling horizon.

Allowing

1. A component tank may receive and feed components at the same time;
2. A component tank may feed some or all blenders simultaneously;
3. Multiple component tanks may feed a blender at the same time;
4. A product tank may receive and feed products at the same time;
5. A product tank may deliver multiple orders at the same time; and
6. Multiple product tanks may deliver an order at the same time.

Solution Approach

A discrete-time formulation is adopted in our models as it leads to simple time-related structure of the equations when compared to the continuous-time representation. As previously stated, one of the main advantages of the discrete-time formulation is the linearity of the terms involving flow rates and time intervals. Although a discrete-time model represents an approximation of the real-life problem,¹¹ our goal is to obtain close-to-optimal solutions for large-scale problems

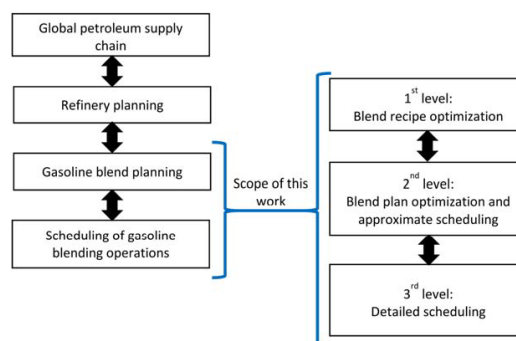


Figure 1. Proposed decomposition of the gasoline blend planning and scheduling problem.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

with short execution times. For comparison purposes, some continuous-time model examples from the literature were solved using our discrete-time approach.

We use the hierarchical framework shown in Figure 1 and our decomposition approach is described in Figure 2. The first level optimizes the blend recipes; the second level uses the blend recipes from the first level to determine how much, when and in which blender a product should be processed, and the allocation of swing tanks to specific products along the horizon; finally, the third level computes the delivery and production rates, and the start and end times of all tasks that minimize the number of blend runs, using the decisions from the second level. We assume that the production rates of the blend components are given by the solution of the refinery planning level; therefore, the inventory cost does not need to be included at any of these three levels as the refinery will carry the inventories as either blend components or as finished gasoline grades. The first and second levels are a decomposition of the gasoline blend planning problem. By solving the recipe optimization separately, nonlinear models can be solved more efficiently since the problem is modeled as a NLP instead of a MINLP, as illustrated in Part I of this work.

To arrive at short execution times, a temporal decomposition technique can be applied to solve the third level (i.e., the detailed scheduling model). As the second level is solved for the entire scheduling horizon, we know that the constraints imposed by the second-level solution constrain the third-level scheduling problem close to the optimal solution (if the constraints of the second level contain a feasible schedule). Hence, we have decided to use a forward and reverse rolling window techniques to compute the schedules at different steps of the algorithm. The scheduling horizon is divided in small subintervals, denoted as L -intervals, which represent the width of the rolling window. The forward rolling window sequence is used to generate an initial solution that takes into consideration the conditions at the beginning of the horizon. Once the initial solution is computed, the reverse rolling window sequence is used to determine if the number of blend runs can be reduced.

Inventory pinch concept

The inventory pinch concept used in this work is defined in detail in Castillo et al.,²⁰ and a brief review can be found

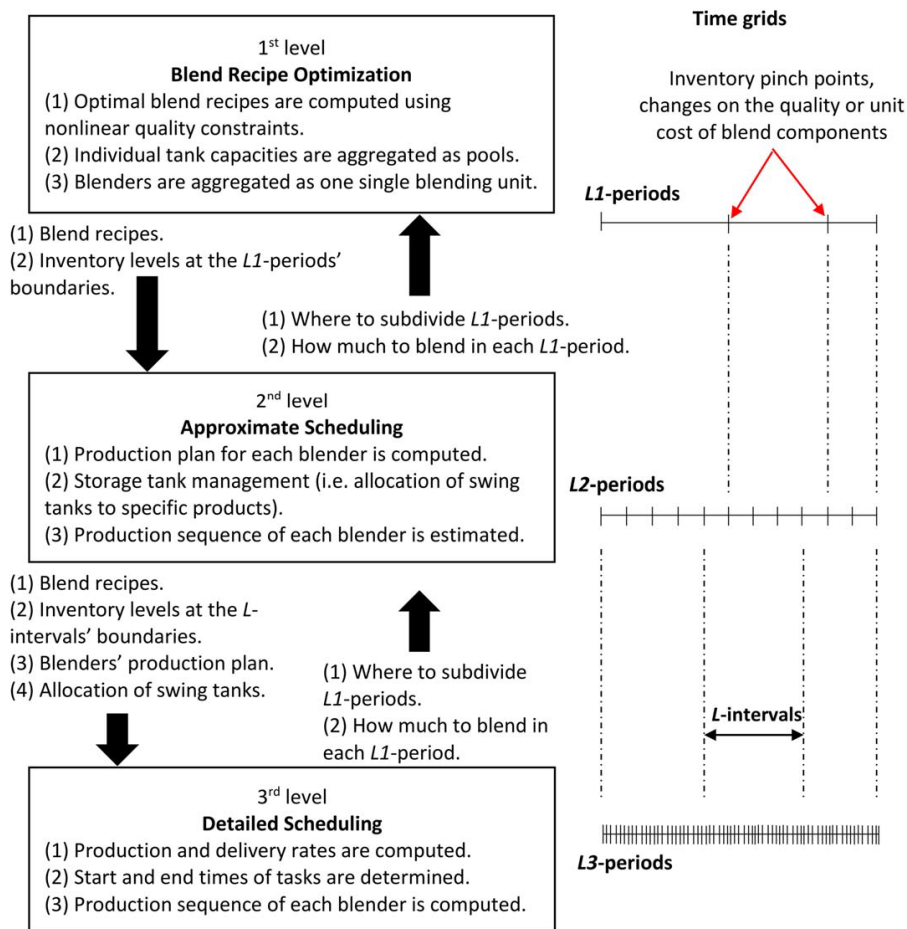


Figure 2. Inventory pinch-based algorithm for gasoline blend scheduling.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

in Part I of this article. The inventory pinch points are the times where the cumulative average total production (CATP) curve changes its slope, in order to remain above the cumulative total demand (CTD) curve, the minimum possible number of times, and at the closest distance to the CTD. These cumulative curves are also known as composite curves.²¹ $V(0)$ represents the initial inventory available. The concept can be applied directly to multiple products (i.e., the demand of all products is aggregated in such case), and minimum inventory limits and target inventories can be incorporated easily. The inventory pinch points can be seen as well in the grand composite curve (i.e., CATP-CTD values) as those points in time where the inventory goes to zero. The cumulative curves provide the minimum production quantities required to meet the demand in each interval where the CATP has a constant slope; however, the grand composite curve given by CATP-CTD does not represent the actual total inventory profile nor the slopes of the CATP curve are the actual production rates; those values will be calculated through our Multiperiod Inventory Pinch (MPIP) algorithm.

The inventory pinch points define the times within the planning/scheduling horizon where the product inventories

are at the minimum allowed limits, and they are used to define the time grids of each level.

Time grids construction of each level

At the first level, the boundaries of the time periods are initially delimited by the inventory pinch points, the times when the quality of blend components changes, and the times when the unit cost of blend components or products vary.

At the second level, the boundaries of the time periods are defined by the planner taking into consideration the following:

- The boundaries of the first level. All boundaries of the first level must exist as well at the second level.
- The minimum time a storage tank will be holding a specific product. The smaller the time periods at the second level, the better the time resolution of assigning the swing tanks to specific products. In our case studies, we use one-day periods at the second level, but it is possible to use 1/2 or 1/4 of the day as the duration of these time periods as dictated by operational considerations.
- The minimum time that a blending unit will require to produce the minimum threshold amount. In this case, larger

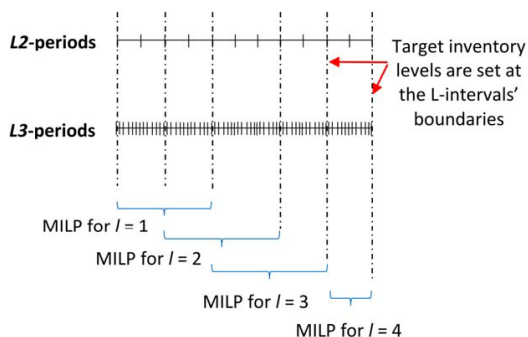


Figure 3. L-intervals at the third level.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

periods are preferred to avoid inventory infeasibilities (i.e., slack variables with nonzero values) due to the blending units not being able to fulfill the minimum production threshold constraint.

- The delivery windows. The higher the number of the period boundaries that correspond to the start and end times of the delivery windows, the fewer iterations the algorithm will require. The second-level periods are still aggregated periods; all tasks are assumed to be completed at the end boundary. Therefore, inventory infeasibilities may appear if a delivery window ends before the conclusion of the second-level period. As a heuristic rule, an original delivery window is narrowed when it spans less than the half of a second-level period, if and only if the order is possible to be met at maximum order delivery rate within the reduced window; otherwise, another second-level period should be considered.

At the third level, the time periods are small enough that only one task in a given unit can take place (e.g., a blender can only blend one specific gasoline grade). They are usually 1- or 2-hours long.

The length of the time periods at any level does not need to be uniform, and their boundaries are not required to coincide with the start of a calendar day, month and so forth. For example, it is possible to use a few smaller periods at the second level if it is known that no blender will operate at some intervals of the horizon (e.g., after the last delivery window if it ends before the conclusion of the horizon). For sake of exposition, the time periods of the first, second, and third level, are denoted as L1-periods, L2-periods, and L3-periods, respectively. One L1-period contains one or more L2-periods, and one L2-period contains several L3-periods; therefore, the product demand, blend component supply, and blend capacity at any level are the aggregated values of the corresponding periods of the next lower level.

Mathematical Models

The mathematical models are presented next. The following sets are used:

- A = $\{(\alpha) \mid \text{set of different supply flow rates of blend components}\}$
- B1 = $\{(\text{bl}) \mid \text{set of blenders}\}$
- E = $\{(e) \mid \text{set of quality properties}\}$
- G = $\{(g) \mid \text{set of time slots for blend run allocation}\}$
- I = $\{(i) \mid \text{set of blend components}\}$
- J = $\{(j) \mid \text{set of product storage tanks}\}$
- K = $\{(k) \mid \text{set of time periods at the first level or L1-periods}\}$

- L = $\{(l) \mid \text{set of time intervals in which the scheduling horizon is solved}\}$
- M = $\{(m) \mid \text{set of time periods at the second level or L2-periods}\}$
- N = $\{(n) \mid \text{set of time periods at the third level or L3-periods}\}$
- O = $\{(o) \mid \text{set of orders}\}$
- P = $\{(p) \mid \text{set of products}\}$
- BJ = $\{(\text{bl}, j) \mid \text{blender bl can feed tank } j\}$
- BP = $\{(\text{bl}, p) \mid \text{blender bl can process product } p\}$
- JP = $\{(j, p) \mid \text{tank } j \text{ can hold product } p\}$
- MK = $\{(m, k) \mid \text{L2-periods contained in each L1-period}\}$
- NA = $\{(n, \alpha) \mid \text{blend component supply profile } \alpha \text{ occurs within L3-period } n\}$
- OP = $\{(o, p) \mid \text{order } o \text{ consists of product } p\}$
- JO = $\{(j, o) \mid \text{tank } j \text{ can deliver order } o\}$
- GM = $\{(g, m) \mid \text{slots } g \text{ for blend allocation is contained in period } m\}$
- ML = $\{(m, l) \mid \text{L2-periods contained in interval } l\}$
- MLE = $\{(n, m) \mid \text{last L2-period contained in interval } l\}$
- NM = $\{(n, m) \mid \text{L3-periods contained in each L2-period}\}$
- NMF = $\{(n, m) \mid \text{all L3-periods, except the first one, contained in each L2-period}\}$
- NL = $\{(n, l) \mid \text{L3-periods contained in interval } l\}$
- NLE = $\{(n, l) \mid \text{last L3-period contained in interval } l\}$
- NLF = $\{(n, l) \mid \text{all L3-periods, except the first one, contained in interval } l\}$
- NLO = $\{(n, l, o) \mid \text{L3-periods contained in interval } l \text{ when order } o \text{ can be delivered}\}$
- JON = $\{(j, o, n) \mid \text{tank } j \text{ may deliver order } o \text{ during L3-period } n\}$
- JPN = $\{(j, p, n) \mid \text{L3-periods when tank } j \text{ can store product } p\}$
- NLOA = $\{(n, l, o) \mid \text{L3-period contained in interval } l \text{ when order } o \text{ can start to be delivered}\}$

In principle, the models presented here can be adapted to any scheduling problem with similar characteristics. Numbering of the equations continues from that of Part I of this article.

First level—Blend recipe optimization

The first-level objective is to minimize the blend cost by determining the optimum volume fractions (i.e., blend recipes) to mix the blend components available into final products that meet quality specifications and demand requirements. At this level, the values of the product demand, blend component supply, and blend capacity are aggregated values for each of the L1-periods, and product storage tanks are aggregated into product pools. The mathematical model of the first level presented in Part I of this article is used without modifications. The objective function defined by Eq. 1 minimizes the blend cost (Eq. 2) and the inventory infeasibilities. The penalty coefficients for the product slacks variables are much greater than the unit cost coefficients of the blend components. Slack variables will be zero at the optimal solution (i.e., penalty coefficients will not affect the final blend cost); if they have nonzero values, it means that the problem is infeasible because there is not enough blend components to blend products as required by the quality specifications or demand requirements. Inventory cost is not included in the objective function as inventory levels will be at the minimum allowed at the end of each L1-period (because the CATP curve touches the CTD curve).

In the first iteration of the algorithm, the volumes to be blended in each L1-period are the minimum amount required to fulfill the demand; thus, the solution of the first level model is a lower bound of the global blend cost. Discrete variables are not required at the first level because compliance of minimum blend size thresholds and maximum blenders' capacity constraints are handled by adjusting the volumes to be blended in each L1-period by moving the minimum possible amount of volume to the previous L1-periods.

$$\min Z_{L1} = \text{BlendCost}_{L1} + \sum_{k \in K} \left\{ \begin{array}{l} \sum_{i \in I} \text{Penalty}_{bc,L1} \cdot (S_{bc,L1}^+(i, k) + S_{bc,L1}(i, k)) \\ + \sum_{p \in P} \text{Penalty}_{pool,L1} \cdot (S_{pool,L1}^+(p, k) + S_{pool,L1}(p, k)) \end{array} \right\} \quad (1)$$

$$\text{BlendCost}_{L1} = \sum_{k \in K} \left(\sum_{i \in I, p \in P} \text{Cost}_{bc}(i) \cdot V_{comp,L1}(i, p, k) \right) \quad (2)$$

Second level—Blend plan optimization and approximate scheduling

The blend plan consists on determining (1) how much to blend of each product and in which blender in each L2-period; (2) allocation of swing tanks to specific products in each L2-period; and (3) the inventory profiles of all storage tanks along the planning horizon.

The second level computes the optimal blend plan using the blend recipes from the first level. The blend recipes of each L1-period are fixed in the corresponding L2-periods. The second level is basically a disaggregation step of the first-level decisions. A MILP model is used to deal with constraints as the minimum blend size threshold, and others that require discrete variables. The second level is solved in two phases: blend planning optimization followed by approximate scheduling. In this way, inventory infeasibilities are detected in less time.

Blend planning optimization at the second level minimizes the blend cost and the inventory slack variables (see Eq. 16). If a feasible operation can be obtained using the blend recipes from the first level, inventory slack variables will be zero at the solution of the second level; otherwise, the inventory slacks will show which specific products, by how much, and in which L2-periods they cannot be produced in the amounts required. To ensure that the nonzero slacks will appear on the product tanks, the product inventories at the L1-period boundaries are fixed instead of setting the production targets from the first level, and:

1. Penalty coefficients for the products' inventory slack variables are smaller in comparison with the penalty for the components' inventory slacks (i.e., $\text{Penalty}_{bc,L2} \gg \text{Penalty}_{pr,L2}(m) \forall m$; $\text{Penalty}_{pool,L2}(m) = \text{Penalty}_{pr,L2}(m) \forall m$).

2. The penalty coefficients for the product inventory slacks decrease with time (i.e., $\text{Penalty}_{pr,L2}(m) > \text{Penalty}_{pr,L2}(m+1) \forall m$) in order to move the inventory infeasibilities as late as possible in the planning horizon. In this way, the use of a given blend recipe is maximized. The penalty coefficients must decrease as fast as possible and after each L1-period boundary a significant change must take place.

When the solution of this phase has inventory infeasibilities (i.e., component supply or blender constraints are such that the recipes computed at the first level are not feasible within a L1-period), the algorithm will subdivide such L1-periods and reoptimize the blend recipes.

Given the assumption that the flow rates of blend components are given by the solution of the refinery planning level, inventory cost is not included at this level as the refinery will carry the total inventories as either blend components or finished gasoline grades

$$\min Z_{L2}^{feas} = \text{BlendCost}_{L2} + \sum_{m \in M} \left\{ \begin{array}{l} \sum_{i \in I} \text{Penalty}_{bc,L2}(m) \cdot (S_{bc,L2}^+(i, m) + S_{bc,L2}^-(i, m)) \\ + \sum_{p \in P} \text{Penalty}_{pool,L2}(m) \cdot (S_{pool,L2}^+(p, m) + S_{pool,L2}^-(p, m)) \\ + \sum_{j \in J} \text{Penalty}_{pr,L2}(m) \cdot (S_{pr,L2}^+(j, m) + S_{pr,L2}^-(j, m)) \end{array} \right\} \quad (16)$$

Equation 17 computes the blend cost

$$\text{BlendCost}_{L2} = \sum_{m \in M} \left(\sum_{(bl,p) \in BP} \sum_{i \in I} \text{Cost}_{bc}(i) \cdot V_{comp,L2}(i, p, bl, m) \right) \quad (17)$$

Approximate scheduling at the second level minimizes the number of product transitions in the blenders and in the product storage tanks, as well as the number of blend runs (see Eq. 18). Binary variable $x_{L2}(p, bl, m)$ defines a blend run; that is, it determines if product p is going to be produced in blender bl during period m if its value is 1. Blend runs are penalized because a solution at the second level can suggest to blend the same product in the same blender for several adjacent L2-periods, thus, not incurring in a penalty for product changeover in the blender; however, they are not likely to have the same blending rate, which is a constraint at the third level to define a blend run. Therefore, it is better to have the minimum number of blend runs, and then reduce the expected number of product changeovers (i.e., $\text{Penalty}_{BR,L2}(bl) \gg \text{Penalty}_{BS,L2}$). If the inventory slacks are zero at the blend planning solution, we know that the blend recipes and inventory targets from the first-level solution can yield a feasible blend plan; then, if those are fixed at the second level, the blend cost does not need to be included in Eq. 18.

Equations 19–66 only appear in Part I of this article

$$\min Z_{L2}^{opt} = \sum_{m \in M} \left\{ \begin{array}{l} \sum_{(bl,p) \in BP} \text{Penalty}_{BR,L2}(bl) \cdot x_{L2}(p, bl, m) \\ + \sum_{j \in J} \text{Penalty}_{TS,L2}(j) \cdot ue_{L2}(j, m) \end{array} \right\} + \sum_{g \in G} \left(\sum_{bl \in Bl} \text{Penalty}_{BS,L2} \cdot xe_{L2}(bl, g) \right) \quad (18)$$

Third level—Detailed scheduling

The decisions of the second level are now disaggregated at the third level. The goals of the third level are to determine:

1. The blending rates for each blend run;
2. The delivery rates from all tanks to each order along the scheduling horizon;
3. The production sequence in each blender; and
4. Start and end times of all tasks.

We use the blend recipes from the first level, and the inventory levels, the swing tanks allocation, the production of each blender, and the delivery plan from the second level to reduce the search space and model size at the third level.

The scheduling horizon is divided in various subintervals, denoted as L -intervals (see Figure 3), and a MILP model is solved for each one of them. The boundaries of the L -intervals must be synchronized with the boundaries of some $L2$ -periods to enable the inventory levels computed at the second level to be fixed at the start and end boundaries of the L -intervals. The L -intervals are solved first in a forward sequence, then the length of these L -intervals is increased and they are solved in a reverse sequence. During the forward pass, L -intervals do not need to overlap as the only goal is to check if the second-level solution is feasible or not. During the reverse pass, L -intervals may overlap to reduce the number of blend runs while solving relatively small size models.

The reverse pass decreases the number of blend runs by merging the same-product blends that are adjacent to the boundaries of the L -intervals used during the forward pass. Due to our model formulation that uses the start of the blend runs to count them, as well as the fulfillment of some constraints (e.g., minimum running time and minimum production volume of a blend run), the feasibility of the L -interval being solved with respect to the rest of the scheduling horizon is easier to attain using a reverse pass than a second forward pass.

The third level (at each forward and reverse pass) is solved in two phases: a feasibility phase followed by an optimization phase.

Objective Function of the Third-Level Feasibility Phase. The objective function is given by Eq. 67; it minimizes the blend cost, the inventory infeasibilities, and the delivery infeasibilities. As the third level uses the inventory levels computed by the second level, the inventory cost does not need to be included at this level. The blend cost is computed by Eq. 68. Inventory slack variables will be zero at the solution of the third level if a feasible operation can be obtained using the blend recipes computed at the first level and the blend plan from the second level. If this is not the case, the inventory slacks will show which specific products, by how much, and in which $L3$ -periods they cannot be produced in the amounts required to meet the demand. Once again, to have the nonzero slacks on the product tanks, inventory levels from the second level are fixed instead of the production targets, and:

1. Penalty coefficients for the products' inventory slack variables are smaller in comparison with the penalty for the components' inventory slacks (i.e., $\text{Penalty}_{\text{bc},L3}(n) \gg \text{Penalty}_{\text{pr},L3}(n) \forall n$), and they decrease with time (i.e., $\text{Penalty}_{\text{bc},L3}(n) \gg \text{Penalty}_{\text{bc},L3}(n+1) \forall n$). Component slacks appear when component tanks overflow.

2. The penalty coefficients for the product inventory slacks decrease with time (i.e., $\text{Penalty}_{\text{pr},L3}(n) > \text{Penalty}_{\text{pr},L3}(n+1) \forall n$) to move the inventory infeasibilities as late as possible in the planning horizon.

3. Penalty coefficients for the delivery slacks are higher than those of the component and product inventory slacks (i.e., $\text{Penalty}_{\text{Dpr},L3} \gg \text{Penalty}_{\text{bc},L3}(n) \gg \text{Penalty}_{\text{pr},L3}(n) \forall n$).

If the solution of the MILP has inventory infeasibilities, it indicates that component supply or blender constraints are

such that the recipes computed at the first level and/or the blend plan from the second level are not feasible within a $L2$ -period. The algorithm will subdivide the corresponding $L1$ -period and reoptimize the blend recipes

$$\begin{aligned} \min Z_{L3}^{\text{feas}} = & \text{BlendCost}_{L3} + \sum_{o \in O} \text{PenaltyD}_{\text{pr},L3} \\ & \cdot \left(S_{\text{order},L3}^+(o) + S_{\text{order},L3}(o) \right) \\ & + \sum_{n \in N} \left\{ \begin{aligned} & \sum_{i \in I} \text{Penalty}_{\text{bc},L3}(n) \cdot \left(S_{\text{bc},L3}^+(i, n) + S_{\text{bc},L3}(i, n) \right) \\ & + \sum_{j \in J} \text{Penalty}_{\text{pr},L3}(n) \cdot \left(S_{\text{pr},L3}^+(j, n) + S_{\text{pr},L3}(j, n) \right) \end{aligned} \right\} \end{aligned} \quad (67)$$

$$\begin{aligned} \text{BlendCost}_{L3} = & \sum_{n \in N} \sum_{\text{bl} \in \text{Bl}} \sum_{p \in P} \sum_{i \in I} \text{Cost}_{\text{bc}}(i) \\ & \cdot V_{\text{comp},L3}(i, p, \text{bl}, n) \end{aligned} \quad (68)$$

Trying to solve the entire scheduling horizon at once will result in a third-level model with a large number of equations and discrete variables which may be computationally inefficient to solve. However, the scheduling horizon can be divided in smaller time intervals denoted as L -intervals and Eqs. 67 and 68 are replaced by Eqs. 69 and 70, respectively

$$\begin{aligned} \min Z_{L3}^{\text{feas}}(l) = & \text{BlendCost}_{L3}(l) + \sum_{o \in O} \text{PenaltyD}_{\text{pr},L3} \\ & \cdot \left(S_{\text{order},L3}^+(o, l) + S_{\text{order},L3}(o, l) \right) \\ & + \sum_{n \in \text{NL}} \left\{ \begin{aligned} & \sum_{i \in I} \text{Penalty}_{\text{bc},L3}(n) \cdot \left(S_{\text{bc},L3}^+(i, n) + S_{\text{bc},L3}(i, n) \right) \\ & + \sum_{j \in J} \text{Penalty}_{\text{pr},L3}(n) \cdot \left(S_{\text{pr},L3}^+(j, n) + S_{\text{pr},L3}(j, n) \right) \end{aligned} \right\} \end{aligned} \quad (69)$$

$$\begin{aligned} \text{BlendCost}_{L3}(l) = & \sum_{n \in \text{NL}} \sum_{\text{bl} \in \text{Bl}} \sum_{p \in P} \sum_{i \in I} \text{Cost}_{\text{bc}}(i) \\ & \cdot V_{\text{comp},L3}(i, p, \text{bl}, n) \end{aligned} \quad (70)$$

Notice that, if no inventory infeasibilities appear, we have

$$\min Z_{L3}^{\text{feas}} = \sum_l \min Z_{L3}^{\text{feas}}(l)$$

And $\text{BlendCost}_{L3} = \sum_l \text{BlendCost}_{L3}(l)$.

Objective Function of the Third-Level Optimization Phase. After a feasible schedule is generated (i.e., all slack variables have a value of zero at the solution of the feasibility phase of the third level), the third level minimizes the number of blend runs, penalizes variations in the delivery rates, punish late deliveries, and reduces the number of tanks receiving product from the same blend run (reduction along the duration of the blend run as a blender only feeds one tank at a time). Equation 71 is the objective function for the optimization phase where the first term penalizes the number of blend runs, the second term penalizes irregular delivery rates, the third term penalizes late deliveries (i.e., $\text{Penalty}_{\text{Dpm},L3}(n) > \text{Penalty}_{\text{Dpm},L3}(n-1) \forall n$), the fourth term represents the penalty associated with long blend runs, and the last term is the penalty for sending product from one blender to different storage tanks during the same blend run but at different times (the blenders can only send product to only one tank at a time). In this work, the higher penalty corresponds to the number of blend runs. Equation 71 does not minimize the blend cost because the blend recipes and the

inventory levels at the boundaries of the L -interval are fixed (i.e., blend cost will be the same as that computed during the feasibility phase)

$$\min_{L_3} Z_{L_3}^{\text{opt}}(l) = \sum_{n \in \text{NL}} \left\{ \begin{array}{l} \sum_{\text{bl} \in \text{BI}} \text{PenaltyBsw}_{L_3} \cdot s_{w_{L_3}}(\text{bl}, n) \\ + \sum_{(j,p) \in \text{JON}} \text{PenaltyDsw}_{L_3} \cdot \text{Dsw}_{L_3}(j, o, n) \\ + \sum_{(j,p) \in \text{JON}} \text{PenaltyDpm}_{L_3}(n) \cdot \text{Del}_{\text{pr}, L_3}(j, o, n) \\ + \sum_{\text{bl} \in \text{BI}} \text{PenaltyBL}_{L_3} \cdot t_{\text{blend}, L_3}(\text{bl}, n) \\ + \sum_{j \in J} \text{PenaltyTsw}_{L_3} \cdot v_{e_{L_3}}(j, n) \end{array} \right\} \quad (71)$$

Material Balance on Blend Component Tanks. The volumetric balance on the blend components ensures that the initial inventory plus the supply are equal to the final inventory plus the amount transferred to the blenders, during a L_3 -period. At this level, the L_3 -periods are small enough that the supply rate of blend components is constant within such periods. Equation 72a is used during the feasibility phase to include the slack variables. If the blend recipes lead to a feasible solution, then the slack variables have a value of zero at the feasibility phase solution and they can be omitted during the optimization phase (Eq. 72b)

$$F_{\text{bc}}(i, \alpha) \cdot t_{L_3}(n) + V_{\text{bc}, L_3}(i, n-1) - V_{\text{bc}, L_3}(i, n) - \sum_{(\text{bl}, p) \in \text{BP}} V_{\text{comp}, L_3}(i, p, \text{bl}, n) + S_{\text{bc}, L_3}^+(i, n) - S_{\text{bc}, L_3}^-(i, n) = 0 \quad (72a)$$

$$\forall i, n \in \text{NL}, a \in \text{NA}$$

$$F_{\text{bc}}(i, \alpha) \cdot t_{L_3}(n) + V_{\text{bc}, L_3}(i, n-1) - V_{\text{bc}, L_3}(i, n) - \sum_{(\text{bl}, p) \in \text{BP}} V_{\text{comp}, L_3}(i, p, \text{bl}, n) = 0 \quad (72b)$$

$$\forall i, n \in \text{NL}, a \in \text{NA}$$

Fixed Blend Recipe. Equation 73 fixes the blend recipe $r(i, p, k)$ from the first level in its corresponding L_3 -periods. Note that $r(i, p, k)$ is a parameter and not a variable at the third level. $F_{\text{blend}, L_3}(p, \text{bl}, n)$ is the production rate of product p in blender bl during period n

$$V_{\text{comp}, L_3}(i, p, \text{bl}, n) = r(i, p, k) \cdot t_{L_3}(n) \cdot F_{\text{blend}, L_3}(p, \text{bl}, n) \quad (73)$$

$$\forall i, (p, \text{bl}) \in \text{BP}, (n, m) \in \text{NM}, n \in \text{NL}$$

Blender Constraints. Equation 74 establishes that a blender may only blend a product according to the second level blend plan. Equation 75 constrains the blender to process only one product during a L_3 -period. $x_{L_3}(p, \text{bl}, n)$ is a 0–1 continuous variable which value is 1 if product p is blended in bl during period n , and 0 otherwise. Note that $x_{L_2}(p, \text{bl}, m)$ is a parameter and not a variable at the third level. $w_{\text{blend}, L_3}(\text{bl}, n)$ is a 0–1 continuous variable which can only take the value equal to 1 if the blender is idle, or equal to 0 if it is running. All 0–1 continuous variables are set to be less than or equal to 1 (equations omitted here)

$$x_{L_3}(p, \text{bl}, n) \leq \sum_{m \in I} x_{L_2}(p, \text{bl}, m) \quad \forall (\text{bl}, p) \in \text{BP}, (n, m) \in \text{NM}, n \in \text{NL} \quad (74)$$

$$w_{\text{blend}, L_3}(\text{bl}, n) + \sum_{p \in \text{BP}} x_{L_3}(p, \text{bl}, n) = 1 \quad \forall \text{bl}, n \in \text{NL} \quad (75)$$

Equations 76 and 77 observe that the production rate must be equal to or less than the maximum blending rate, and equal to or greater than the minimum blending rate, respectively

$$F_{\text{blend}, L_3}(p, \text{bl}, n) \leq F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot x_{L_3}(p, \text{bl}, n) \quad \forall (\text{bl}, p) \in \text{BP}, n \in \text{NL} \quad (76)$$

$$F_{\text{blend}, L_3}(p, \text{bl}, n) \geq F_{\text{blend}}^{\text{min}}(\text{bl}) \cdot x_{L_3}(p, \text{bl}, n) \quad \forall (\text{bl}, p) \in \text{BP}, n \in \text{NL} \quad (77)$$

Volume Transferred from Blenders to Product Tanks. The set JPN is constructed according to the solution of the second-level model; this enables the product allocation of swing tanks determined at the second level to be fixed at the third level. Equations 78–81 force a blender to feed at most only one product tank during a L_3 -period. Binary variable $v_{L_3}(j, \text{bl}, n)$ specifies that blender bl is feeding product tank j during period n if its value is equal to 1

$$V_{\text{trans}, L_3}(j, \text{bl}, n) \leq F_{\text{blend}, L_3}(p, \text{bl}, n) \cdot t_{L_3}(n) + F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot t_{L_3}(n) \cdot (1 - v_{L_3}(j, \text{bl}, n)) \quad (78)$$

$$\forall (\text{bl}, p) \in \text{BP}, (\text{bl}, j) \in \text{BJ}, (j, p) \in \text{JPN}, n \in \text{NL}$$

$$V_{\text{trans}, L_3}(j, \text{bl}, n) \geq F_{\text{blend}, L_3}(p, \text{bl}, n) \cdot t_{L_3}(n) - F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot t_{L_3}(n) \cdot (1 - v_{L_3}(j, \text{bl}, n)) \quad (79)$$

$$\forall (\text{bl}, p) \in \text{BP}, (\text{bl}, j) \in \text{BJ}, (j, p) \in \text{JPN}, n \in \text{NL}$$

$$V_{\text{trans}, L_3}(j, \text{bl}, n) \leq F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot t_{L_3}(n) \cdot v_{L_3}(j, \text{bl}, n) \quad \forall (\text{bl}, j) \in \text{BJ}, n \in \text{NL} \quad (80)$$

$$w_{\text{blender}, L_3}(\text{bl}, n) + \sum_{j \in \text{BJ}} v_{L_3}(j, \text{bl}, n) = 1 \quad \forall \text{bl}, n \in \text{NL} \quad (81)$$

Equation 81 forces continuous variable $w_{\text{blender}, L_3}(\text{bl}, n)$ to be only 0 or 1. Equation 82 observes that the blender bl is feeding product p into a tank j that contains such product. Let us note that the allocation of swing tanks computed at the second level is fixed at the third level. This equation forces continuous variable $x_{L_3}(\text{bl}, n)$ to be only 0 or 1

$$\sum_{j \in \text{JP}} v_{L_3}(j, \text{bl}, n) = x_{L_3}(p, \text{bl}, n) \quad \forall \text{bl}, n \in \text{NL} \quad (82)$$

Equations 83a and 83b define if a change in the destination tank for the product in the blender has occurred at period n (when the blender is already running). However, let us note that this change is constrained by Eqs. 81 and 82 to be made only to another tank assigned to hold the same product at period n . 0–1 continuous variable $x_{e_{L_3}}(\text{bl}, n)$ represents the start or end of a blend run if its value is 1; therefore, $v_{e_{L_3}}(\text{bl}, n)$ is a 0–1 continuous variable due to being penalized in Eq. 71

$$v_{e_{L_3}}(\text{bl}, n) \geq v_{L_3}(j, \text{bl}, n) - v_{L_3}(j, \text{bl}, n-1) - x_{e_{L_3}}(\text{bl}, n) \quad \forall (\text{bl}, j) \in \text{BJ}, n \in \text{NL} \quad (83a)$$

$$v_{e_{L_3}}(\text{bl}, n) \geq v_{L_3}(j, \text{bl}, n-1) - v_{L_3}(j, \text{bl}, n) - x_{e_{L_3}}(\text{bl}, n) \quad \forall (\text{bl}, j) \in \text{BJ}, n \in \text{NL} \quad (83b)$$

It is important to notice that the only binary variable at the third level is $v_{L3}(j, bl, n)$, which indicates that the number of discrete variables only increases with the system structure (i.e., number of blenders and product tanks) and the discretization of the scheduling horizon (i.e., the number of L3-periods).

Material Balance on Product Tanks. The volumetric balance on the product tanks specifies that the initial inventory plus the volume supplied by the blenders are equal to the final inventory plus the amount delivered, during a L3-period. Equation 84a is used during the feasibility phase to include the slack variables, and Eq. 84b is utilized during the optimization phase

$$\begin{aligned} \sum_{bl \in BI} V_{trans,L3}(j, bl, n) + V_{pr,L3}(j, n-1) - V_{pr,L3}(j, n) - t_{L3}(n) \\ \cdot \sum_{o \in ION} D_{pr,L3}(j, o, n) + S_{pr,L3}^+(j, n) - S_{pr,L3}^+(j, n) = 0 \\ \forall j, n \in NL \end{aligned} \quad (84a)$$

$$\begin{aligned} \sum_{bl \in BI} V_{trans,L3}(j, bl, n) + V_{pr,L3}(j, n-1) - V_{pr,L3}(j, n) \\ - t_{L3}(n) \cdot \sum_{o \in ION} D_{pr,L3}(j, o, n) = 0 \forall j, n \in NL \end{aligned} \quad (84b)$$

Inventory Limits. Inventory constraints are only forced on individual storage tanks by Eqs. 85 and 86

$$V_{bc}^{\min}(i) \leq V_{bc,L3}(i, n) \leq V_{bc}^{\max}(i) \quad \forall i, n \in NL \quad (85)$$

$$V_{pr}^{\min}(j) \leq V_{pr,L3}(j, n) \leq V_{pr}^{\max}(j) \quad \forall j, n \in NL \quad (86)$$

Initial Inventory. Equations 87 and 88 set the initial state of the blend component and product tanks, respectively. Note that these equations are required only for the first L -interval

$$V_{bc,L3}(i, n=0) = V_{bc}^{\text{start}}(i) \quad \forall i \quad (87)$$

$$V_{pr,L3}(j, n=0) = V_{pr}^{\text{start}}(j) \quad \forall j \quad (88)$$

Blend Runs. At this level, 0–1 continuous variable $x_{L3}(bl, n)$ represents a state transition in blender bl at the beginning of period n ; in other words, a transition from being running to being idle, or vice versa. 0–1 continuous variable $sw_{L3}(bl, n)$ represents the start of a blend run in blender bl at the beginning of period n . Equation 89 defines which product the blender is processing if it is running at the beginning of the horizon, and Eq. 90 determines if the blender is idle at time zero. Equations 91 and 92 identify a state transition in the blender; they force continuous variable $x_{L3}(bl, n)$ to be 1 when a product starts or finishes a blend run. Equations 93a and 93b force $x_{L3}(bl, n)$ to be 0 when blender is running or idle during two consecutive L3-periods, respectively. Equations 94 and 95 define if a blend run has started in the blenders

$$x_{L3}(p, bl, n=0) = x^{\text{start}}(p, bl) \quad \forall (bl, p) \in BP \quad (89)$$

$$w_{blend,L3}(bl, n=0) = w_{blend}^{\text{start}}(bl) \quad \forall bl \quad (90)$$

$$\begin{aligned} x_{L3}(bl, n) \geq x_{L3}(p, bl, n) - x_{L3}(p, bl, n-1) \quad \forall (bl, p) \in BP, n \\ \in NL \end{aligned} \quad (91)$$

$$\begin{aligned} x_{L3}(bl, n) \geq x_{L3}(p, bl, n-1) - x_{L3}(p, bl, n) \quad \forall (bl, p) \in BP, n \\ \in NL \end{aligned} \quad (92)$$

$$\begin{aligned} x_{L3}(bl, n) \leq w_{blend,L3}(bl, n) + w_{blend,L3}(bl, n-1) \quad \forall bl, n \in NL \\ (93a) \end{aligned}$$

$$\begin{aligned} x_{L3}(bl, n) \leq 2 - w_{blend,L3}(bl, n) - w_{blend,L3}(bl, n-1) \quad \forall bl, n \in NL \\ (93b) \end{aligned}$$

$$\begin{aligned} sw_{L3}(bl, n) \geq x_{L3}(bl, n) + w_{blend,L3}(bl, n-1) - 1 \quad \forall bl, n \in NL \\ (94) \end{aligned}$$

$$\begin{aligned} sw_{L3}(bl, n) \leq \frac{x_{L3}(bl, n) + w_{blend,L3}(bl, n-1)}{2} \quad \forall bl, n \in NL \\ (95) \end{aligned}$$

Constant Blending Rate. Equation 96 ensures that the blending rate is constant during a blend run. Equation 97 specifies the blending rate of the blenders at the beginning of the scheduling horizon

$$\begin{aligned} -F_{blend}^{\max}(bl) \cdot x_{L3}(bl, n) \leq F_{blend,L3}(p, bl, n) \\ -F_{blend,L3}(p, bl, n-1) \leq F_{blend}^{\max}(bl) \cdot x_{L3}(bl, n) \\ \forall (bl, p) \in BP, n \in NL \end{aligned} \quad (96)$$

$$F_{blend,L3}(p, bl, n=0) = F_{blend}^{\text{start}}(p, bl) \quad \forall (bl, p) \in BP \quad (97)$$

Constraints on the Minimum Running Times of the Blenders. To know when a blender has surpassed its minimum allowed running time when processing product p , the cumulative running time of a blender at the end of period n is computed [i.e., $t_{blend,L3}(bl, n)$]. Equation 98 sets the running time at the beginning of the scheduling horizon. Equations 99 and 100 compute the cumulative running time at period n as the cumulative running time in period $n-1$ plus the duration of period n , if the blender is not idle. Equation 101 restarts to zero the cumulative running time when the blender goes idle. Parameter H is the length of the scheduling horizon

$$t_{blend,L3}(bl, n=0) = t_{blend}^{\text{start}}(bl) \quad \forall bl \quad (98)$$

$$\begin{aligned} t_{blend,L3}(bl, n) \leq t_{blend,L3}(bl, n-1) + t_{L3}(n) \\ + H \cdot w_{blend,L3}(bl, n) \quad \forall bl, n \in NL \end{aligned} \quad (99)$$

$$\begin{aligned} t_{blend,L3}(bl, n) \geq t_{blend,L3}(bl, n-1) + t_{L3}(n) - H \\ \cdot w_{blend,L3}(bl, n) \quad \forall bl, n \in NL \end{aligned} \quad (100)$$

$$\begin{aligned} t_{blend,L3}(bl, n) \leq H \cdot (1 - w_{blend,L3}(bl, n)) \quad \forall bl, n \in NL \\ (101) \end{aligned}$$

Equation 102 observes that state transitions in the blender can only occur after the minimum running time has been achieved or surpassed

$$\begin{aligned} x_{L3}(bl, n) \leq \frac{t_{blend,L3}(bl, n-1)}{t_{blend}^{\min}(p, bl)} + H \\ \cdot (1 - x_{L3}(p, bl, n-1)) \quad \forall (bl, p) \in BP, n \in NL \end{aligned} \quad (102)$$

Equation 102 does not avoid solutions with blend runs completed at the end of the horizon (or L -interval) with less than the minimum running time; therefore, Eq. 103 is necessary to ensure that any blend run within the scheduling horizon has a run time at least equal to the minimum

$$\begin{aligned} t_{blend,L3}(bl, n) \geq t_{blend}^{\min}(p, bl) \cdot x_{L3}(p, bl, n) \quad \forall (bl, p) \in BP, n \\ \in NLE \end{aligned} \quad (103)$$

Constraints on the Minimum Idle Times of the Blenders. We consider product-dependent changeover times in the blenders. The cumulative idle time of blender bl at the end of period n is denoted as $it_{blend,L3}(bl, n)$. Equation 104 sets

the idle time at the beginning of the scheduling horizon. Equations 105 and 106 compute the cumulative idle time at period n as the cumulative idle time at period $n - 1$ plus the duration of period n , if the blender is not running. Equation 107 restarts to zero the idle time when the blender starts a blend run

$$it_{\text{blend,L3}}(\text{bl}, n=0) = it_{\text{blend}}^{\text{start}}(\text{bl}) \quad \forall \text{bl} \quad (104)$$

$$it_{\text{blend,L3}}(\text{bl}, n) \leq it_{\text{blend,L3}}(\text{bl}, n-1) + t_{\text{L3}}(n) + H \cdot (1 - w_{\text{blend,L3}}(\text{bl}, n)) \quad \forall \text{bl}, n \in \text{NL} \quad (105)$$

$$it_{\text{blend,L3}}(\text{bl}, n) \geq it_{\text{blend,L3}}(\text{bl}, n-1) + t_{\text{L3}}(n) - H \cdot (1 - w_{\text{blend,L3}}(\text{bl}, n)) \quad \forall \text{bl}, n \in \text{NL} \quad (106)$$

$$it_{\text{blend,L3}}(\text{bl}, n) \leq H \cdot w_{\text{blend,L3}}(\text{bl}, n) \quad \forall \text{bl}, n \in \text{NL} \quad (107)$$

Equation 108 ensures that a blender cannot start to process product p in period n unless the cumulative idle time at period $n - 1$ is greater than the minimum required

$$it_{\text{blend,L3}}(\text{bl}, n-1) \geq it_{\text{blend}}^{\text{min}}(p, \text{bl}) \cdot x_{\text{L3}}(p, \text{bl}, n) - H \cdot (1 - w_{\text{blend,L3}}(\text{bl}, n-1)) \quad \forall (\text{bl}, p) \in \text{BP}, n \in \text{NL} \quad (108)$$

Constraints on the Minimum Production Volume of the Blend Runs. We consider that a blend run should produce at least a minimum amount of product. The cumulative volume produced by blender bl as the start of the blend run up to the end of period n (if the blend run has not been completed) is denoted as $vc_{\text{blend,L3}}(\text{bl}, n)$. Equation 109 sets the cumulative volume at the beginning of the scheduling horizon. Equations 110 and 111 compute the cumulative volume produced up to period n as the cumulative volume at period $n - 1$ plus the volume blended in period n , if the blender is not idle. Equation 112 restarts to zero the cumulative volume when the blender ends a blend run and while it remains idle

$$vc_{\text{blend,L3}}(\text{bl}, n=0) = vc_{\text{blend}}^{\text{start}}(\text{bl}) \quad \forall \text{bl} \quad (109)$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \leq vc_{\text{blend,L3}}(\text{bl}, n-1) + \sum_{p \in \text{BP}} V_{\text{blend,L3}}(p, \text{bl}, n) + F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot H \cdot w_{\text{blend,L3}}(\text{bl}, n) \quad \forall \text{bl}, n \in \text{NL} \quad (110)$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \geq vc_{\text{blend,L3}}(\text{bl}, n-1) + \sum_{p \in \text{BP}} V_{\text{blend,L3}}(p, \text{bl}, n) - F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot H \cdot w_{\text{blend,L3}}(\text{bl}, n) \quad \forall \text{bl}, n \in \text{NL} \quad (111)$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \leq F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot H \cdot (1 - w_{\text{blend,L3}}(\text{bl}, n)) \quad \forall \text{bl}, n \in \text{NL} \quad (112)$$

Equations 113 and 114 constraint a blender to end a blend run until the minimum volume has been produced

$$x_{\text{L3}}(\text{bl}, n) \leq \frac{vc_{\text{blend,L3}}(\text{bl}, n-1)}{\text{VMIN}_{\text{blend}}^{\text{min}}(p, \text{bl})} + F_{\text{blend}}^{\text{max}}(\text{bl}) \cdot H \cdot (1 - x_{\text{L3}}(p, \text{bl}, n-1)) \quad \forall (\text{bl}, p) \in \text{BP}, n \in \text{NL} \quad (113)$$

$$vc_{\text{blend,L3}}(\text{bl}, n) \geq \text{VMIN}_{\text{blend}}^{\text{min}}(p, \text{bl}) \cdot x_{\text{L3}}(p, \text{bl}, n) \quad \forall (\text{bl}, p) \in \text{BP}, n \in \text{NLE} \quad (114)$$

Equations 109–114 are not used if these minimum blend thresholds are not considered; in that case, the minimum volume that may be produced by a blend run is given as the minimum blending rate multiplied by the minimum running time. From Eqs. 99 to 113, parameter H can be substituted by the smallest possible number to tighten the model.

Order Delivery. Equation 115 computes the volume to deliver of order o during L -interval l , denoted as $\text{Demand}_{\text{or-der,L3}}(o, l)$, according to the second-level solution. As stated before, in this work only one L2-period is contained by one L -interval during the feasibility phase. Note that $of_{\text{L2}}(o, m)$ is a parameter and not a variable at the third level

$$\text{Demand}_{\text{order,L3}}(o, l) = \text{Demand}(o) \cdot \sum_{m \in \text{ML}} of_{\text{L2}}(o, m) \quad \forall o \quad (115)$$

Equations 116a, 116b, 117a, and 117b represent the material balance around the lifting/shipping ports. Together, Eqs. 116a and 117a force the delivery to occur within the corresponding window during the feasibility phase (and Eqs. 116b and 117b during the optimization phase). Equations 116a and 116b constraint the amount shipped within the delivery window [the delivery window is given by set $\text{NLO}(n, l, o)$] to be equal to the demand, while Eqs. 117a and 117b observe that the amount shipped within the whole L -interval is equal to the demand. Therefore, no delivery occurs outside the delivery window

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NLO}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \right] = \text{Demand}_{\text{order,L3}}(o, l) + S_{\text{order,L3}}^+(o, l) - S_{\text{order,L3}}(o, l) \quad \forall o \quad (116a)$$

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NLO}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \right] = \text{Demand}_{\text{order,L3}}(o, l) \quad \forall o \quad (116b)$$

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NL}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \right] = \text{Demand}_{\text{order,L3}}(o, l) + S_{\text{order,L3}}^+(o, l) - S_{\text{order,L3}}(o, l) \quad \forall o \quad (117a)$$

$$t_{\text{L3}}(n) \cdot \left[\sum_{n \in \text{NL}} \sum_{(j,o) \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \right] = \text{Demand}_{\text{order,L3}}(o, l) \quad \forall o \quad (117b)$$

Delivery rate of order o must be equal to or less than its specified maximum (Eq. 118), and the total delivery rate of tank j must be equal to or less than its maximum delivery capacity (Eq. 119)

$$\sum_{j \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \leq D_{\text{order}}^{\text{max}}(o) \quad \forall o, n \in \text{NLO} \quad (118)$$

$$\sum_{o \in \text{JON}} D_{\text{pr,L3}}(j, o, n) \leq D_{\text{pr}}^{\text{max}}(j) \quad \forall j, n \in \text{NLO} \quad (119)$$

To keep the delivery rate as constant as possible, Eqs. 120 and 121 compute the difference between delivery rates, which is penalized in Eq. 71

$$D_{\text{sw,L3}}(j, o, n) \geq D_{\text{pr,L3}}(j, o, n) - D_{\text{pr,L3}}(j, o, n-1) \quad \forall (j, o) \in \text{JON}, n \in \text{NLO} \quad (120)$$

$$D_{\text{sw,L3}}(j, o, n) \geq D_{\text{pr,L3}}(j, o, n-1) - D_{\text{pr,L3}}(j, o, n) \quad \forall (j, o) \in \text{JON}, n \in \text{NLO} \quad (121)$$

Target Inventories. During the forward rolling window pass, Eqs. 122 and 123 set the inventory targets for the blend

component and the product tanks at the end boundary of the L -interval l according to the second-level solution. Starting inventories are set according to the solution from the previous L -interval ($l - 1$).

During the reverse rolling window pass, the inventory targets for the blend component and the product tanks at the start boundary of the L -interval l are set by the values computed during the forward pass. Inventories at the end boundary of the L -interval are set according to the solution from the L -interval ($l + 1$)

$$V_{bc,L3}(i, n) = V_{bc,L2}(i, m) \quad \forall i, n \in \text{NLE}, m \in \text{MLE} \quad (122)$$

$$V_{pr,L3}(j, n) = V_{pr,L2}(j, m) \quad \forall j, n \in \text{NLE}, m \in \text{MLE} \quad (123)$$

MPIP Scheduling Algorithm

The algorithm presented here is based on the gasoline blend planning and scheduling problem; however, it can be applied to any system with similar characteristics.

Step 1: Modify the original order delivery windows if necessary. This is required to transform the problem from a continuous-time to a discrete-time domain.

Step 2: Construct the CTD and the CATP curves. Determine the pinch point(s) location.

Step 3: Set iteration counters $\text{iter}_s = 1$ and $\text{iter}_p = 1$. Divide the scheduling horizon (at the first level) in the number of L1-periods indicated by the pinch points, unit cost, and quality breakpoints of blend components.

Step 4: Solve the first-level model to compute the optimal blend recipes (Eq. 1–15).

- In the first “planning” iteration ($\text{iter}_p = 1$), the volumes to produce of each product in each L1-period are the minimum amounts required to meet the aggregated demand in each L1-period.

- In following “planning” iterations ($\text{iter}_p > 1$), the volumes to produce are defined according to the solution of the second level (see Step 7) or the third level (see Step 12).

- If the quantity to be produced [i.e., $V_{\text{blend,L1}}(p, k)$] violates the maximum blend capacity or the minimum blend size threshold constraints, volumes are adjusted by moving the least amount possible of volume to previous L1-periods (i.e., preblending).

- If any inventory slack variable has a nonzero value at the solution, the problem is infeasible as the availability or quality of blend components is not enough to meet the product quality specifications or to deliver the products within the delivery windows. Stop.

Step 5: Solve the second level blend planning model (Eqs. 16, 17, and 19–55).

Step 6: If all the inventory slack variables from Step 5 are zero, a feasible blend plan, based on optimal recipes computed at the first level, has been found; go to Step 9. Otherwise, continue to Step 7: Subdivide the L1-period at the end boundary of the L2-period with the first infeasibility.

- The volumes to be blended in each new L1-period are given by the solution of Step 5 plus the positive slacks minus the negative slacks. If $V_{\text{blend,L1}}(p, k)$, for any p or k , violates the maximum blend capacity or the minimum blend size threshold constraints, volumes are adjusted by moving the least amount possible of volume to previous L1-periods.

Step 8: Set $\text{iter}_p = \text{iter}_p + 1$. Go back to Step 4.

Step 9: Solve the second level approximate scheduling model (Eqs. 17–61).

- The solution provides an initial guess (upper bound) for the number of blend runs and a lower bound for the number of product transitions in the blenders.

- The solution provides the product-tank allocation that yields the minimum product transitions in the storage tanks.

Step 10: Define the L -intervals for the third level forward rolling window pass.

- The length of these L -intervals can be different but their boundaries must match L2-period boundaries.

- L -intervals are not required to overlap.

Step 11: Solve the third-level feasibility phase model (Eqs. 69, 70, and 72–123) for each L -interval from Step 10.

- Use the forward rolling window approach, that is, start with $l = 1$ and move on to the end of the scheduling horizon. In this way, the initial conditions of each L -interval are computed considering the conditions at the beginning of the horizon.

Step 12: If the solution from Step 11 has all slack variables, for all L -intervals, with values equal to zero, continue to Step 14. Otherwise:

- Subdivide the scheduling horizon at the first level at the start boundary of the L2-period containing the largest infeasibility. Small infeasibilities may disappear with new recipes.

- In the next iteration, if nonzero slacks continue to appear in the same L2-period, subdivide the scheduling horizon at the first level at the end boundary of that L2-period. If such period is the last L2-period, or if nonzero slacks continue appearing in the same period in following iterations; subdivide the horizon at the first level in such a way that the blend run with nonzero slacks is delimited (i.e., a blend recipe is computed for that particular blend run). A good guess to delimit the blend run is given by $t_{\text{blend,L2}}(p, \text{bl}, m) + \text{it}_{\text{blend}}^{\text{min}}(p, \text{bl})$.

- Necessity of moving volumes to previous time periods as in Step 7 is less common at this level.

- Inventory targets computed at the first level are not fixed at the corresponding boundaries at the second level if the inventory targets at other second level boundaries already define how much product is required to be blended with the same blend recipe; for example, when a single blend run is delimited at the first level.

Step 13: Set $\text{iter}_s = \text{iter}_s + 1$, and go back to Step 4.

Step 14: Solve the third level optimization phase model (Eqs. 70–123) for each L -interval from Step 10.

- Use the forward rolling window approach.

- If inventory infeasibilities appear or the model for an L -interval is infeasible, this is due to a different boundary condition (i.e., the initial blender conditions are different from those used at Step 11). Resolve previous L -interval and fix the initial conditions computed in Step 11.

Step 15: Determine new L -intervals for the reverse rolling window pass:

- If the same product is being blended in adjacent L2-periods, in the same blender and using the same blend recipe, those L2-periods can be grouped into one L -interval.

- L -intervals can be overlapped if necessary.

- Increasing the length of the L -intervals increases the model size as well as the execution times required to solve it.

Step 16: Solve the third level feasibility phase model (Eqs. 69, 70, and 72–121) for each L -interval from Step 15, including the following constraint

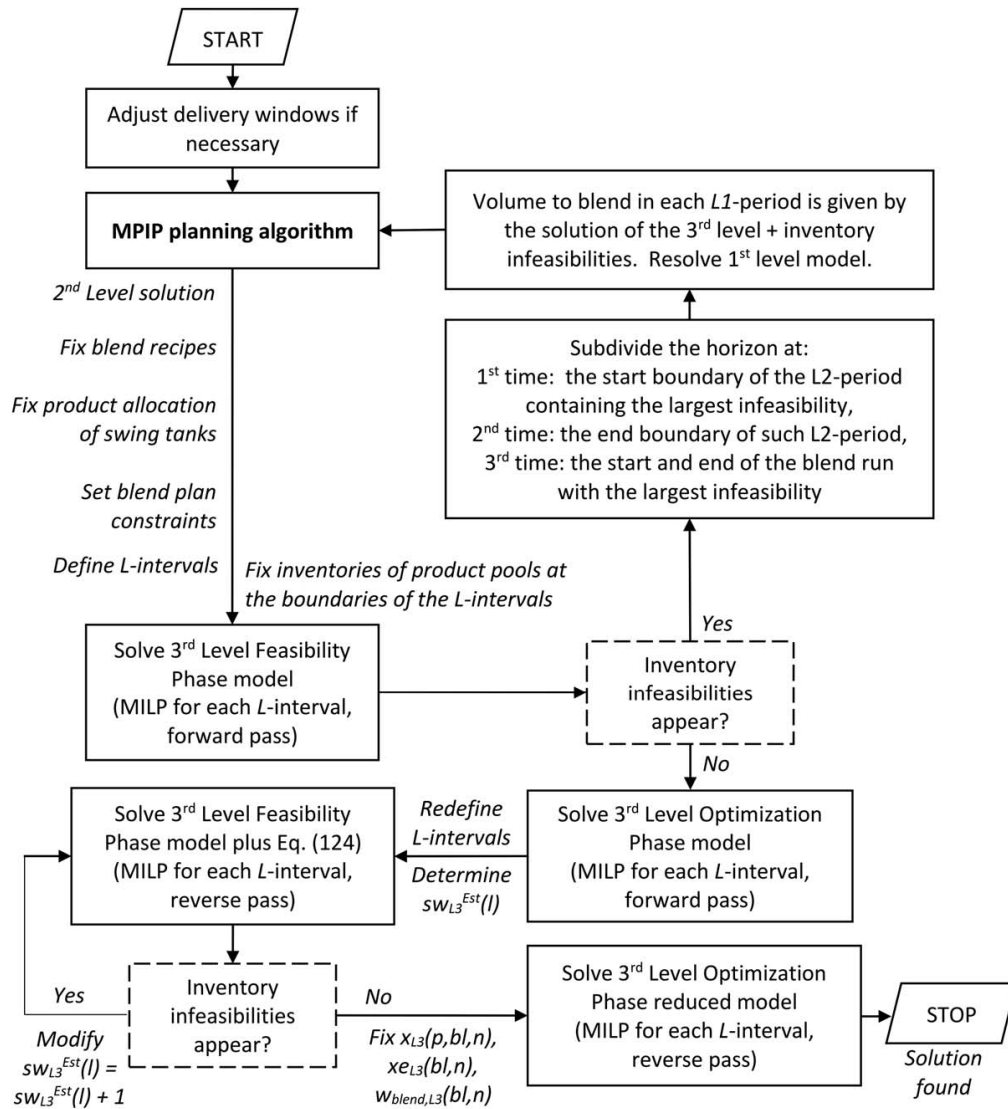


Figure 4. Flowchart for the MPIP scheduling algorithm.

$$\sum_{bl \in BI} sw_{L3}(bl, n) = sw_{L3}^{Est}(l) \quad \forall n \in NL \quad (124)$$

- sw_{L3}^{Est} is the estimated lower bound for the total number of blend runs according to solution from Step 14 (i.e., one blend run per product being blended in the L -interval l , minus the blend runs that have started in L -interval $l-1$)

- Use a reverse rolling window approach, that is, start with L -interval $l=L$ and continue until the beginning of the horizon. In this way, changes at the boundary conditions will not affect the feasibility of previous L -intervals.

- If the solution for an L -interval contains inventory slacks greater than zero, set $sw_{L3}^{Est}(l) = sw_{L3}^{Est}(l) + 1$ and resolve the L -interval. Repeat until the solution does not contain inventory infeasibilities (feasibility is guaranteed by Step 14).

- This step aims to join the most possible blend runs computed at Step 14.

- Inventory levels are only fixed at the boundaries of the L -interval being solved. In other words, if the L -intervals overlap (i.e., end boundary of L -interval $l-1$ is within L -interval l), then the new inventory levels computed at l should be used for $l-1$.

- Variables $x_{L3}(p,bl,n)$ and $w_{blend,L3}(bl,n)$ are fixed in the last $L3$ -period n of L -interval l if in $L3$ -period $n+1$ the blender is starting or continuing a blend run. In addition, parameters $t_{blend}^{min}(p,bl)$ and $VMIN_{blend}(p,bl)$ are adjusted for Eqs. 103 and 114, respectively, to account for blend runs that are completed in L -interval $l+1$.

Step 17: Fix variables $F_{blend,L3}(p,bl,n)$, $x_{L3}(p,bl,n)$, $x_{eL3}(bl,n)$, and $w_{blend,L3}(bl,n)$, and solve the third-level

Table 1. Test Set #1, Case Study 13—Demand Profile #5

Delivery Window		Product U87		Product U91		Product U93	
Start Time (h)	End Time (h)	Order	Amount ($\times 10^3$ bbl)	Order	Amount ($\times 10^3$ bbl)	Order	Amount ($\times 10^3$ bbl)
0	24	O1	60	O14	50	O26	30
24	48	O2	50	O15	80	O27	30
48	72	O3	50	O16	70	O28	45
72	96	O4	70	O17	50	–	0
96	120	O5	90	O18	50	O29	30
120	144	O6	80	O19	30	O30	40
144	168	O7	130	O20	30	O31	30
168	192	O8	50	O21	30	O32	30
192	216	–	0	O22	30	O33	30
216	240	O9	30	O23	30	O34	30
240	264	O10	50	–	0	O35	30
264	288	O11	50	O24	40	–	0
288	312	O12	50	O25	30	O36	30
312	336	O13	80	–	0	O37	40

optimization phase reduced model (Eqs. 70, 71, 80–86, 88, and 115–121) for each L -interval from Step 15. Stop.

- This step reduces the variations in the final delivery rates and the variations in the transfer sequence from the blenders to the storage tanks.

Steps 2–9 are the same steps of the MPIP planning algorithm for gasoline blend planning (see Part I of this article); therefore, the flowchart of the MPIP scheduling algorithm can be visualized as Figure 4. Hence, we refer to the iterations at Step 8 as planning iterations ($iter_p$), and those at Step 13 as scheduling iterations ($iter_s$).

Numerical Results and Discussion

We present two sets of case studies: Test Set #1 contains the examples discussed in Part I of this article (i.e., case studies with our own data), whereas Test Set #2 are examples taken from the literature⁴ which incorporate all of the characteristics given in the Problem Statement section of this article. Test Set #1 is used to produce detailed schedules from the blend plans computed at the second level of our case studies presented in Part I of this article, and Test Set #2 is used to compare our MPIP scheduling algorithm with a full-space continuous-time model. Test Set #1 contains problems where the operation range of the blenders is small, the demand requirements are high, and there is one nonlinear blending property; whereas in Test Set #2 there is a large difference between the maximum and minimum blending rates, demand requirements are low, and all blending properties are assumed linear. For all examples in each Test Set, the length of the L_3 -periods is 1 h.

Test Set #1: Integration of planning and detailed scheduling

The gasoline blending system and all data required for these case studies appear in Part I of this article.

The penalty coefficients at the third level are:

- For all blenders, the penalty for starting a blend run is $PenaltyBsw_{L_3} = 1 \times 10^4$ \$.
- The penalty for variations in the delivery rate is $PenaltyDsw_{L_3} = 100$ \$.
- The penalty profile for late delivery of orders [i.e., $PenaltyDpm_{L_3}(n)$] increases from 0\$ at the beginning of an L -interval up to 120\$ at the end of the L -interval.
- The penalty for changing the destination tank for the product from the blender is $PenaltyTsw_{L_3} = 500$ \$.

- The penalty for a blender being running for one L_3 -period is $PenaltyBL_{L_3} = 1$ \$.

We consider the highest penalty for a blend run to obtain the production sequence with the minimum possible number of blend runs; then, the next higher penalties are for variations in the delivery rates and late deliveries, and the smallest penalty corresponds to long blend runs. The demand orders in our case studies involve a single product and their delivery time windows are assumed to be one day. As we are using the L_2 -periods as 1 day, there is no need to adjust the time delivery windows.

All case studies have been computed on a DELL Power-Edge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 GB RAM) running Windows Server 2008 R2 OS. GAMS IDE 23.7.3 was used to solve each one of the case studies. The first-level NLP model was solved using IPOPT, and the second- and third-level models were solved using CPLEX 12.3.

Illustrative Example #1: Test Set #1, Case Study 13—Two Nonidentical Blenders, Irregular Supply Profile of Blend Components. Case study 13 was used to illustrate the steps of the MPIP planning algorithm in Part I of this article; now, we will continue with the steps of the MPIP scheduling algorithm. This case study has demand profile #5 (shown in Table 1), 2 blenders (A and B), and the supply flow rate of components is irregular along the planning horizon. We continue the calculations from the solution of the second level approximate scheduling solution presented in Part I of this article. The next step is to solve the third-level feasibility model. Each L_3 -period is 1-hour long.

First Scheduling Iteration. Let us define 14 L -intervals (Step 10), each one for each L_2 -period (i.e., each L -interval has 24 L_3 -periods). The inventory levels corresponding to the L_2 -period boundaries are fixed at the respective boundaries of the L -intervals. In this example, solution of the third-level feasibility phase (Step 11) presents two inventory slacks with nonzero values in the last L -interval on the light naphtha (LNP) component tank, $S_{bc,L_3}^+(LNP,336) = 0.0999 \times 10^3$ bbl, and $S_{bc,L_3}^-(LNP,330) = 0.0999 \times 10^3$ bbl. This means that LNP tank overflows at period $n = 330$. The positive slack at period $n = 336$ only appears because the target inventory at the end of L -interval $l = 14$ is not met as the overflow volume is not considered available anymore for future periods. Therefore, Step 12 of the MPIP scheduling algorithm indicates that we need to subdivide the last L_1 -period $k = 4$ at the point in time corresponding to the start boundary of L_2 -period $m = 14$. This example illustrates that

Table 2. Test Set 1, Case Study 13—Blend Plan

Period ID	L2-Periods		Production Volumes ($\times 10^3$ bbl)				
	Start Time (h)	End Time (h)	Blender A			Blender B	
			U87	U91	U93	U91	U93
$m = 1$	0	24	–	–	–	–	59
$m = 2$	24	48	100	–	–	–	30
$m = 3$	48	72	–	70	–	–	–
$m = 4$	72	96	70	–	–	64	–
$m = 5$	96	120	103.5	–	–	36	30
$m = 6$	120	144	103.5	–	–	30	36
$m = 7$	144	168	93	–	–	30	30
$m = 8$	168	192	50	30	–	–	30
$m = 9$	192	216	–	31.5	–	–	30
$m = 10$	216	240	31.5	33.3	–	–	30
$m = 11$	240	264	48.5	–	–	–	30
$m = 12$	264	288	54.3	–	–	65.2	–
$m = 13$	288	312	45.7	–	–	–	30
$m = 14$	312	336	80	–	–	–	40

typically at the third level the inventory infeasibilities are relatively small.

Second Scheduling Iteration. The new blend recipes computed at Step 4 have a cost equal to $\text{BlendCost}_{L1} = 37,784.52 \times 10^3$ \$, and the approximate schedule computed by the second level (Step 9) is presented in Table 2. Test Set. This approximate schedule has a total cost equal to $Z_{L2} = 38,502.52 \times 10^3$ \$. No product switchover is required in the storage tanks. The third-level feasibility phase (Step 11) did not find any inventory infeasibilities. The third-level optimization phase (Step 14) was solved initially with 14 *L*-intervals (one for each L2-period), and its solution is shown in Figure 5a. Then, two contiguous L2-periods where the same product is being blended in the same blender were merged into a new *L*-interval (Step 15). L2-periods with different blend recipes were not merged because it was considered that a single blend run should have only one blend recipe. The *L*-intervals from Step 15 were solved (Step 16) using the previous solution as starting point. Figure 5b shows the production schedule computed at Step 16. The final delivery schedule and the inventory profiles of blend compo-

nents and product pools (which are computed at Step 17 and that correspond to the production schedule shown in Figure 5b) are presented in Figures 6 and 7, respectively.

Test Set #1 Results. Table 3 shows the difference between the second- and third-level solutions (Step 9 and Step 17 of the MPIP scheduling algorithm, respectively). The blend cost is the same at both levels due to fixed blend recipes and inventory levels. The number of blend runs is smaller at the third level as some of the blend runs at the second level can be combined into single blend runs. The minimum number of product transitions in the blenders is achieved in some cases, and if it is not, the difference is very small. No product transitions are required in the storage tanks. Table 3 also presents the total number of scheduling iterations required and the cumulative execution times required at each level (i.e., the total time at each level considering all scheduling iterations). For our case studies, only three scheduling iterations were required at most.

The forward pass at the third level is carried out using 14 *L*-intervals (one for each L2-period), each one of them with 24 L3-periods. The sum of the cumulative execution times at

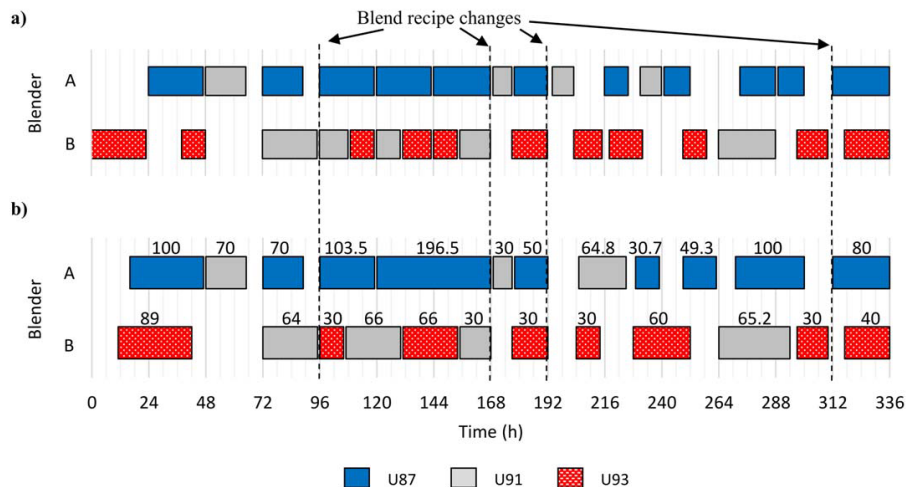


Figure 5. Test Set #1, Case study 13—Production schedule computed using (a) *L*-intervals with 24 L3-periods and (b) using *L*-intervals with 48 L3-periods.

Units in kbbl. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

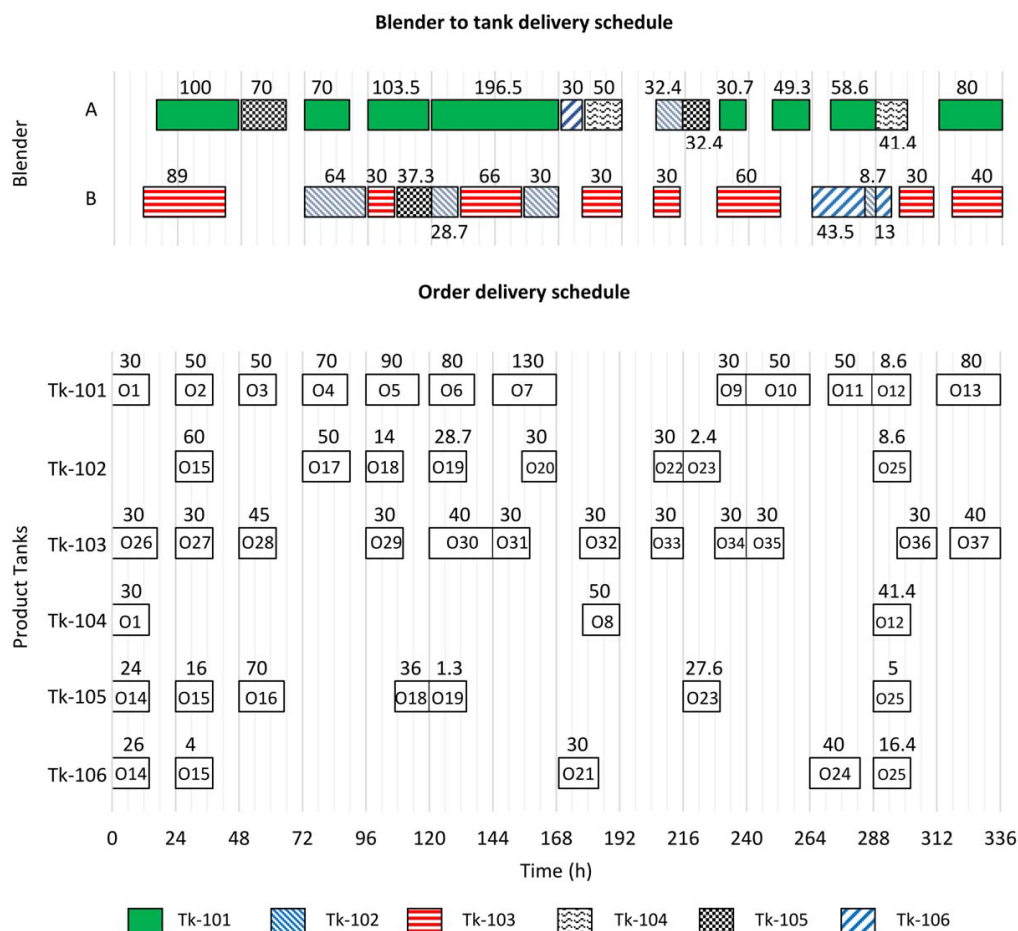


Figure 6. Test Set #1, Case study 13—Delivery schedule.

Units in kbb. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

the first level, the second level, and the third level forward pass is the time required to construct an initial solution for the detailed schedule.

The reverse pass at the third level is carried out using L -intervals with 48 L3-periods. The number of these L -intervals varies in each case study as it depends on the number of L2-periods that are adequate to merge (see Step 15). The execution times required to solve all of these 48-hour L -intervals are shown in the last column from Table 3.

Regarding the model size, the number of discrete variables is greater at the second level (which is solved for the entire horizon) than at the third level with L -intervals of 14 or 48 h; however, the third-level model with 48-hour L -intervals requires more computational effort to be solved. The number of discrete variables at the third level increases only with the addition of blenders or product storage tanks to the system.

Test Set #2: Comparison of the MPIP scheduling algorithm with full-space continuous-time model using unit slots

Examples 3, 4, 7, 8, 9, 12, and 14 from Li and Karimi⁴ were solved using the MPIP scheduling algorithm for com-

parison purposes. All examples were computed on a DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 GB RAM) running Windows Server 2008 R2 OS. GAMS IDE 23.7.3 was used to solve each one of the case studies. This computer is approximately 25% slower and has 4GB RAM less than the machine used by Li and Karimi⁴.

Time delivery windows for orders O8, O10, O13, O19, and O33 were reduced from [118, 190], [150.5, 185.5], [0, 56], [0, 50], and [0, 76], to [120, 190], [150, 185], [0, 48], [0, 48], and [0, 72], respectively. Scheduling horizon at the second level is divided in nine L2-periods (period $m = 1$ to $m = 7$ are 24-hours long, period $m = 8$ has 22 h, while $m = 9$ is a 2-hour period) in example 3, 4, 7, 8, 9, and 12; and the horizon in example 14 is divided in eight L2-periods, one per day. The cost coefficients of the blend components and the penalties associated with the number of blend runs and product transitions in the tanks are the same as those presented by Li and Karimi.⁴ The penalty coefficients for the third level are the same as those used in Test Set #1.

Li and Karimi⁴ use blend indices which blend linearly on a volumetric basis instead of the actual quality properties; therefore, the first level model is a linear program and all the three levels were solved using CPLEX 12.3.

Table 3. Test Set I—Comparison between the Solutions Computed at the Second and Third Level

Case Study	Demand Profile	# Pinch Points	# Blenders	Comp. Supply	Blend Cost ($\times 10^6$ \$)	Second Level Solution			Third Level Solution			Cumulative CPU Time (s)		
						# Blend runs	# Switches ^a	# Blend Runs	# Switches ^a	# Blend Runs	# Switches ^a	Second Level ^b	Third Level Forward Pass	Third Level Reverse Pass
1	1	0	1	C	37,542.5	18	14	16	15	1	0.6	300.0	75.7	905
2	2	1	1	C	38,309.8	24	15	20	17	2	2.1	121.1	95.7	1923
3	3	1	1	C	37,991.1	22	16	21	18	3	1.9	261.4	108.9	1926
4	4	3	1	C	37,991.1	24	10	22	12	1	0.6	300.0	108.0	2696
5	4	2	2	C	37,681.1	26	8	22	9	1	0.9	93.0	153.2	3343
6	5	2	2	C	37,324.3	28	8	24	12	1	2.2	37.3	320.9	652
7	6	3	3	C	37,377.5	33	10	27	10	1	0.9	300.0	104.9	2573
8	1	0	1	I	37,943.4	18	14	17	16	3	1.9	505.2	165.5	981
9	2	1	1	I	38,754.2	26	15	21	19	2	1.2	22.4	106.4	3477
10	3	1	1	I	38,405.2	25	16	21	18	3	2.2	83.6	179.8	2430
11	3	1	2	I	38,405.2	26	11	20	13	1	0.7	197.0	186.1	5106
12	4	2	2	I	38,073.4	26	10	23	10	1	0.9	15.8	114.2	3666
13	5	2	2	I	37,784.5	31	10	25	16	2	2.4	25.7	102.3	1923
14	6	3	3	I	37,796.4	33	10	26	10	1	1.1	300.0	138.6	2,455

^aProduct switches in the blenders.

^bMaximum allocated time in each iteration = 300 s.

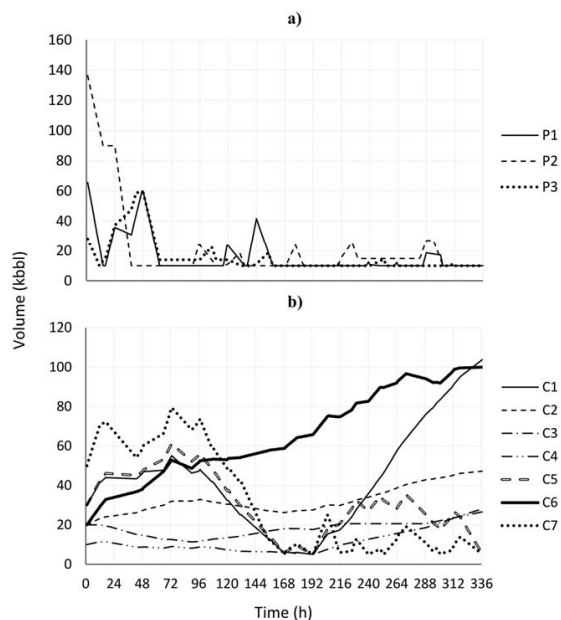


Figure 7. Test Set #1, Case study 13—Inventory profiles of (a) product pools and (b) blend components.

Illustrative Example #2: Example 12—Two Blenders, Constant Supply Profile of Blend Components. In this example, the gasoline blending system has 2 blenders, 9 blend components with dedicated tanks, 5 products, and 11 swing tanks; there are 35 orders to deliver within a scheduling horizon of 8 days (192 h). Nine product qualities are required to be within specification and blend indices are utilized to compute these (i.e., linear quality constraints are used). The supply flow rate of components is constant along the scheduling horizon. The first step is to adjust the start and end times of the delivery windows. The cumulative curves for this example (after the adjustment of the delivery windows) are shown in Figure 8, where it can be seen that there is only one inventory pinch point at time $t = 190$ h. As there are no orders to deliver after the pinch point, there is no production in the last 2 h of the horizon.

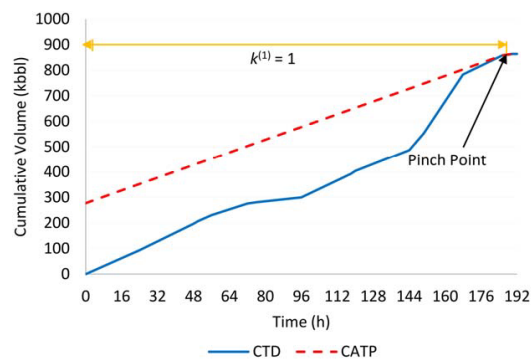


Figure 8. Test Set #2, Example 12—Cumulative curves, inventory pinch points, and L1-periods.

[Color figure can be viewed in the online issue, which is available at [wileyonlinelibrary.com](http://www.wileyonlinelibrary.com).]

Table 4. Test Set #2, Example 12—Blend Recipes for Period $k = 1$ (i.e., time interval [0, 190 h])

Component	Product				
	P1	P2	P3	P4	P5
C1	0.0127	–	0.1000	0.2400	–
C2	0.3873	0.3198	0.2900	0.1000	0.3076
C4	0.4000	0.4500	0.4300	0.3260	0.4000
C5	–	–	–	0.1899	–
C6	0.2000	0.2200	0.1800	0.1441	0.2000
C8	–	0.0102	–	–	0.0924

First Scheduling Iteration. The aggregate demand for L1-period $k = 1$ is 185 kbbl of P1, 190 kbbl of P2, 195 kbbl of P3, 178 kbbl of P4, and 116 kbbl of P5. Production targets for L1-period $k = 1$ are 86.49 kbbl of P1, 89.84 kbbl of P2, 165 kbbl of P3, 129.04 kbbl of P4, and 116 kbbl of P5. As aggregated demand for L1-period $k = 2$ is zero, production targets for that L1-period are zero as well. Production targets are computed as the minimum amount to fulfill aggregated demand and target inventories (see Eq. 65 from Part I of this article). Solving the first-level model (Step 4), which is a LP in this example, provides the blend recipes shown in Table 4. Only one set of blend recipes is required for the entire horizon as there is no production in period $k = 2$. The blend cost is $\text{BlendCost}_{L1} = 14,692.13 \times 10^3$ \$.

The second-level model has nine L2-periods (i.e., $m = 1$ to $m = 7$ are the first 7 days, $m = 8$ contains the first 22 h of day 8, and $m = 9$ contains the last 2 h of day 8) and is solved using the blend recipes from Table 4. The blend planning solution does not contain inventory infeasibilities (i.e., all slack variables have a value equal to zero), and the solution from the approximate scheduling (Step 9) is shown in Table 5. Blend cost at the second level is $\text{BlendCost}_{L2} = 14,692.13 \times 10^3$ \$, the same as the one from the first level. The second level determines that tank PT-102 must hold product P5 for the entire horizon and is the only product transition required in the storage tanks (tank PT-102 is initially empty but assigned to hold product P3).

Note that product P3 is being processed in different blenders (see Table 5), but it may be blended in the same unit; there is no penalty for this situation as both blenders are allowed to produce such product and in both blenders it represents a new blend run and a product transition (i.e., it incurs in the same penalty). Because there is only one product being blended in each L2-period (i.e., only one task being executed within the L2-periods), feasibility of the blend recipes and the blend plan is guaranteed at the third level; however, we still require to determine if the orders can be delivered on time.

The duration of the L3-periods is 1 h; thus, the complete scheduling horizon has 192 L3-periods. The L -intervals at Step 10 are defined as each single day; that is, there are eight L -intervals, each one with 24 L3-periods. The third-

level solution at Step 11 shows that all slack variables have a value equal to zero. The production and delivery schedules computed at Step 14 are shown in Figure 9. The blend cost at the third level is $\text{BlendCost}_{L3} = 14,692.13 \times 10^3$ \$, which is identical to the second-level blend cost. Step 15–17 of the MPIP algorithm can be omitted because there is no product being produced in the same blender in adjacent L2-periods.

It could be argued that, if both blend runs of product P3 are being processed in blender A, a single L -interval spanning [48 h, 168 h] should be able to merge those blend runs into a single one. However, the second level already has determined that this is not possible (at least with the blend recipes being used) as the second level could not find a solution where P3 can be blended in adjacent L2-periods, even when there is blend capacity available.

Test Set #2 Results. Table 6 summarizes the results for the Test Set #2. Although we implemented the full-space continuous-time model⁴ in GAMS and solved all the examples shown in Table 6 in our machine, we chose to present the reported final solution from Li and Karimi⁴ because their execution times are shorter than those required by our machine; however, we show the results that we obtained in our machine for the first integer solution. For examples 3, 4, 7, and 8 from Li and Karimi,⁴ the MPIP scheduling algorithm computed solutions with the same blend cost and the same total number of product transitions (i.e., the same total cost). For Test Set #2, we use the definition for product transitions by Li and Karimi⁴ as the number of blend runs plus the product transitions in the swing tanks. The same or a smaller number of product transitions was found for examples 9, 12, and 14. Please note that in examples 9, 12, and 14 from Li and Karimi,⁴ there seems to be an inconsistency between the data describing the examples and their reported solution because their reported value of the objective function is smaller than the lower bound which is computed as the optimal solution of the aggregate blend problem. As the solution of the aggregate problem is based on the relaxed problem that only minimizes the cost of materials subject to their availability, and inventory and production capacity constraints (i.e., the aggregate problem does not consider the cost of switchovers and their associated constraints), this solution is a lower bound on the optimum. Any results with

Table 5. Test Set #2, Example 12—Blend Plan at the Second Level

ID	L2-period		Product	Blender	
	Start Time (h)	End Time (h)		A	B
$m = 1$	0	24	P2	–	89.84
$m = 2$	24	48	P4	–	129.04
$m = 3$	48	72	P3	–	78.94
$m = 5$	96	120	P5	116.00	–
$m = 7$	144	168	P3	86.06	–
$m = 8$	168	190	P1	86.49	–

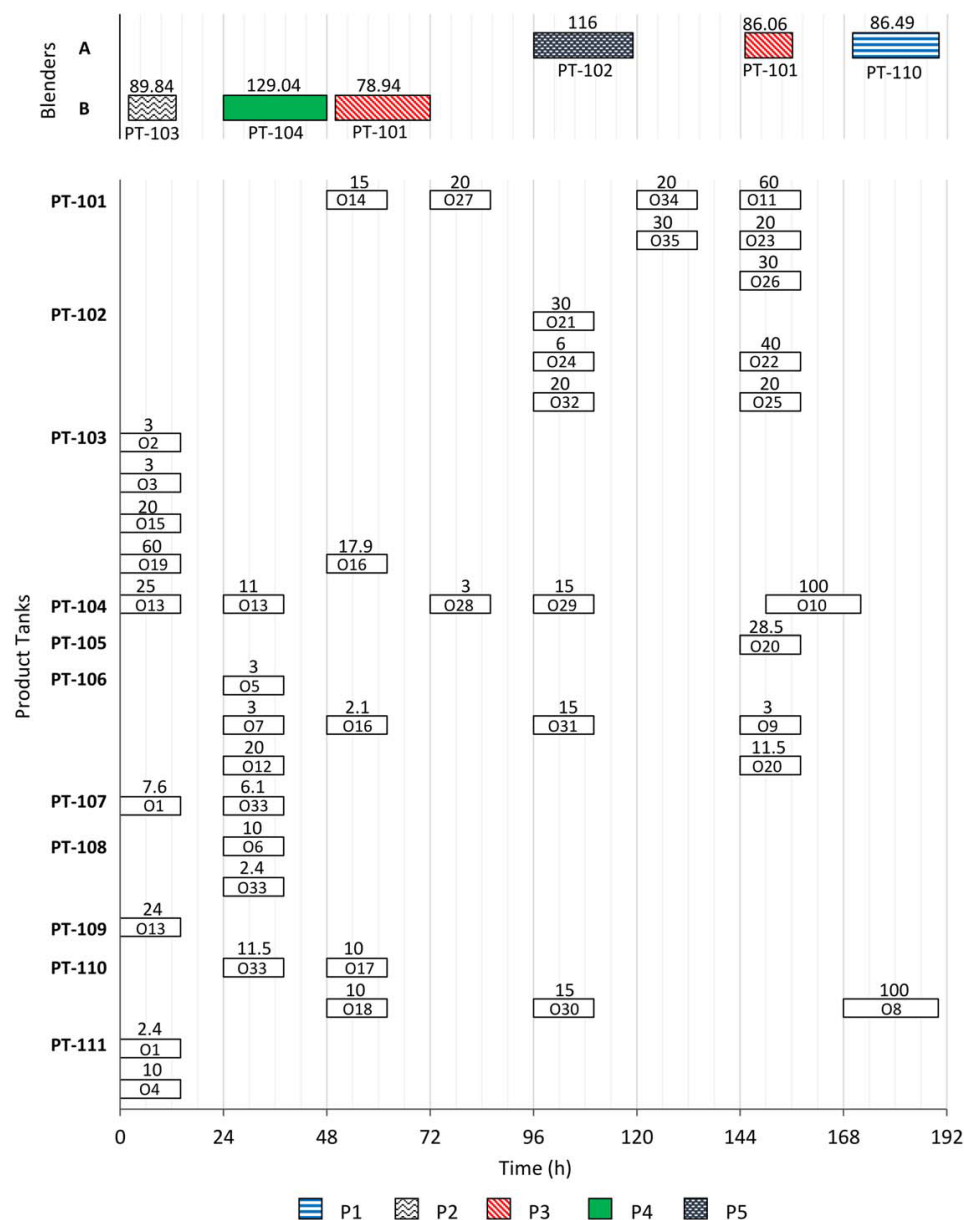


Figure 9. Test Set #2, Example 12—Production and delivery schedule at the third level.

Units in kbbbl. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

even lower value of the objective function correspond to infeasible solutions. Following this reasoning, it appears that Li and Karimi⁴ solutions for example 9, 12, and 14 are infeasible; therefore, it is not meaningful to compare the quality of the reported solutions for these examples with those obtained by the MPIP algorithm.

For large problems, the execution times required by the MPIP scheduling algorithm are considerably smaller than the times required by the continuous-time model. As expected, full-space continuous-time model is faster for small size problems when compared to MPIP algorithm; however, for large size problems the continuous-time model requires a

significant amount of CPU time to find an initial integer solution.

It is important to note that only one scheduling iteration of the MPIP algorithm is required to solve all examples from Test Set #2; moreover, it is not necessary to overlap L -intervals and solve Steps 15–17 to obtain the optimal solution due to few blend runs required to meet the demand. Table 7 shows the model size at each level. It can be seen that the number of discrete variables per L -interval with 24 L 3-periods is greater than the number required by the continuous-time full-space model in Example 3, but it is smaller for all the remaining examples.

Table 6. Test Set #2—Comparison between the MPIP scheduling Algorithm and a Full-Space Continuous-Time Model

Example ID	# Pmch Points	# Blenders	# Orders	Comp. Supply	Full-Space Continuous-Time Model ^a (MILP: CPLEX)			MPIP scheduling Algorithm (LP, MILP: CPLEX)					
					Reported Final Solution ^a		First Integer Solution ^b		Final Solution				
					Obj. Func. ($\times 10^3$ \$)	# Trans. ^c	CPU Time (s)	Obj. Func. ($\times 10^3$ \$)	CPU Time (s)	Obj. Func. ($\times 10^3$ \$)	Blend Cost ($\times 10^3$ \$)	# Trans. ^c	CPU Time (s)
3	1	1	12	C	3,159.1	1	2.6	3,392.3	0.67	3,159.1	3,139.1	1	14.7
4	1	1	15	C	4,556.7	1	4.2	4,820.2	0.33	4,556.7	4,536.7	1	15.3
7	1	1	20	C	8,100.4	3	10.802	11,557.1	108.8	8,100.4	8,040.4	3	16.2
8	1	2	20	C	8,080.4	2	10.819	8,815.8	17.6	8,080.4	8,040.4	2	19.1
9	1	2	23	C	10,573.7	4	10,814	14,134.0	120.2	10,777.2	10,702.7	4	35
12	1	2	35	C	14,764.9	5	46,800	16,191.1	42,619	14,786.6	14,692.1	5	44.5
14	0	3	45	C	17,737.4	9	118,800	23,265.8	117,928	20,381.0	20,286.5	5	372.2
3	1	1	12	I	3,159.1	1	1.2	3,368.6	0.78	3,159.1	3,139.1	1	14.2
4	1	1	15	I	4,556.7	1	2.0	5,988.2	0.94	4,556.7	4,536.7	1	16.2
7	1	1	20	I	8,100.4	3	10,814	9,671.7	67.8	8,100.4	8,040.4	3	17
8	1	2	20	I	8,082.9	2	14,170	8,657.6	614.6	8,080.4	8,040.4	2	17.3
9	1	2	23	I	10,573.7	4	10,817	11,483.3	1,138	10,778.8	10,704.3	4	33.2
12	1	2	35	I	14,809.4	7	46,800	15,363.5	12,671	15,241.7	15,147.2	5	128.7
14	0	3	45	I	17,737.4	9	118,800	23,856.2	116,354	21,121.4	21,046.9	4	333.2

^aReported values by Li and Karimi⁴ where problems were solved using CPLEX 12.1 on a Dell Precision PWS690 computer, Intel Xeon CPU, 3GHz, and 16GB RAM.

^bValues computed by implementing Li and Karimi⁴ model in our machine.

^cTotal number of product transitions in the blenders and storage tanks.

^dValues computed considering the objective function and associated coefficients from Li and Karimi⁴. C = Constant, I = Irregular.

Table 7. Test Set #2—Model Size at Each Level

Example ID	First Level			Second-Level Optimization Phase			Third-Level Optimization Phase			
	# Equations	# Continuous Variables	# Discrete Variables	# Equations	# Continuous Variables	# Discrete Variables	For Each L-Interval with 24 L3-Periods			
							# Equations	# Continuous Variables	# Discrete Variables	
3	423	237	3087	3632	3087	752	6485	10,711	264	246
4	487	237	3117	3695	3117	752	5659	12,399	264	285
7	487	237	3167	3800	3167	752	6872	15,583	264	672
8	487	237	4184	5348	4184	1174	10,212	17,298	528	595
9	588	280	5045	6570	5045	1460	11,421	19,390	528	635
12	588	280	7008	8078	5828	1794	17,232	26,398	528	2237
14	316	148	8078	8078	5828	1794	17,232	34,867	792	3664
3	423	237	3087	3632	3087	752	6485	10,711	264	246
4	487	237	3117	3695	3117	752	5659	12,399	264	285
7	487	237	3167	3800	3167	752	6872	15,583	264	672
8	487	237	4184	5348	4184	1174	10,212	17,298	528	595
9	588	280	5045	6570	5045	1460	11,421	19,390	528	635
12	588	280	7008	8078	5828	1794	17,232	26,398	528	2237
14	316	148	8078	8078	5828	1794	17,232	34,867	792	3664
3	423	237	3087	3632	3087	752	6485	10,711	264	246
4	487	237	3117	3695	3117	752	5659	12,399	264	285
7	487	237	3167	3800	3167	752	6872	15,583	264	672
8	487	237	4184	5348	4184	1174	10,212	17,298	528	595
9	588	280	5045	6570	5045	1460	11,421	19,390	528	635
12	588	280	7008	8078	5828	1794	17,232	26,398	528	2237
14	316	148	8078	8078	5828	1794	17,232	34,867	792	3989

Conclusions

This work introduces an inventory pinch-based, multi-scale algorithm (MPIP scheduling algorithm) to solve integrated gasoline blend planning and scheduling problems. Integrated planning and scheduling is decomposed via three levels, each of them representing a different view of the problem: the first level determines the best blend recipes by considering only the aggregated availability of resources and demand requirements across the planning and scheduling horizon. The second level first computes an optimal blend plan (how much to produce and when) using finer grid time periods; if there is no feasible blend plan based on the blend recipes computed at the first level, an interval at the first level is subdivided at the point of earliest infeasibility and the blend recipes are reoptimized. Once the existence of an optimal blend plan is confirmed, an approximate schedule is optimized. Volumetric constraints and resource allocation from the approximate schedule are constraints on the detailed schedule which is computed at the third level.

The MPIP scheduling algorithm has been tested on two sets of case studies. Test Set #1 includes problems with limited storage and blending capacity, and high product demands. These case studies are tightly constrained and hence are expected to be computationally demanding. Test Set #2 includes examples taken from the literature^{2,4} which are not so tightly constrained (e.g., the blending rates have a wider range between minimum and maximum, there is a large initial inventory of products) and a simple demand profile. For Test Set #2, the MPIP scheduling algorithm computed schedules which have the same or better blend cost and same or better number of product transitions as those provided by a full-space continuous-time model, and in significantly shorter execution times when solving large examples. In several large examples from Test Set #2, the results reported in the literature are lower than the aggregate lower bound, indicating an inconsistency between the problem descriptions and the results reported in the literature.

The scheduling model (i.e., third-level model) presented here only requires one binary variable: the decision to feed product storage tank j from blender bl in period n . Therefore, the model size only increases with the system structure (i.e., with the number of product tanks or blenders) and the number of time periods in which the horizon is discretized. Using only one binary variable at the third level, delivery rates are not forced to be greater than some minimum value. However, low delivery rates are easy to convert into higher rates as the feasibility of the solution is not affected, although nonintermittent deliveries are not guaranteed.

If the scheduling horizon is 2–4 weeks long, the model at the third level becomes a very large MILP model. As second level imposes constraints via approximate schedule, we have chosen to implement forward and reverse rolling window method as a quick heuristic strategy to obtain a good solution. Results presented in this work demonstrate that MPIP scheduling algorithm solves large scale gasoline blending problems to the same or better solution points and much faster than previously published methods.

Future work will be to solve the refinery planning and scheduling problem using our decomposition framework. Decomposition approaches (e.g., Lagrangian decomposition) will also be examined and integrated into our algorithm, especially at the third level.

Acknowledgment

Support by Ontario Research Foundation and McMaster Advanced Control Consortium is gratefully acknowledged.

Notation

Subscripts

bc = refers to a variable or parameter related to the blend component tanks
blend = refers to a variable or parameter related to the blenders
comp = refers to a variable or parameter related to the transfer of volume between component tanks and blenders
L1 = refers to a variable or parameter of the first level
L2 = refers to a variable or parameter of the second level
L3 = refers to a variable or parameter of the third level
order = refers to a variable or parameter related to the orders
pool = refers to a variable or parameter related to the product pools
pr = refers to a variable or parameter related to the individual product tanks
trans = refers to a variable or parameter related to the transfer of volume between blenders and product tanks or pools

Superscripts

max = refers to a maximum value that a variable may have
min = refers to a minimum value that a variable may have if different from zero
start = refers to the initial value at the beginning of the planning horizon that a variable may have
target = refers to a target value for a variable

Parameters

$Cost_{bc}(i)$ = cost of blend component i
 $D_{order}^{max}(o)$ = maximum delivery rate of order o
 $D_{pr}^{max}(j)$ = maximum delivery rate of tank j
Demand(o) = demand of order o for the complete scheduling horizon
Demand_{order,L3}(o,l) = demand of order o in L -interval l
 $F_{bc}(i,\alpha)$ = supply flow rate of blend component i for supply profile α
 $F_{blend}^{max}(bl)$ = maximum blending rate of blender bl
 $F_{blend}^{min}(bl)$ = minimum blending rate of blender bl
 H = length of the planning horizon
 $it_{blend}^{min}(p,bl)$ = minimum idle time required by blender bl before processing product p
Penalty_{bc,L1} = penalty for the inventory slack variables of blend component tanks at the first level
Penalty_{bc,L2}(m) = penalty for the inventory slack variables of component i in L2-period m
Penalty_{bc,L3}(n) = penalty for the inventory slack variables of blend component tanks in L3-period n
Penalty_{pool,L1} = penalty for the inventory slack variables of product pools at the first level
Penalty_{pool,L2}(m) = penalty for the inventory slack variables of product pool p in L2-period m
Penalty_{pr,L2}(m) = penalty for the inventory slack variables of product tank j in L2-period m
Penalty_{pr,L3}(n) = penalty for the inventory slack variables of product tanks in L3-period n
Penalty_{BL}_{L3} = penalty for processing a product in a blender during a L3-period n
Penalty_{BR}_{L2}(bl) = penalty for a blend run processed in blender bl during a L2-period m
Penalty_{BS}_{L2} = penalty for a product transition in a blender at the second level
Penalty_{Bsw}_{L3} = penalty for starting a blend run at the third level
Penalty_D_{pr,L3} = penalty for the delivery slack variables at the third level
Penalty_{Dpm}_{L3}(o,n) = penalty for late delivery of order o at the third level
Penalty_{Dsw}_{L3} = penalty for irregular delivery rates at the third level
Penalty_{TS}(j) = penalty for a product transition in a swing tank at the second level

PenaltyTsw_{L3} = penalty for changing the destination tank for the product in the blender
 $sw_{L3}^{\text{Ext}}(l)$ = parameter that indicates the minimum possible number of blend runs during L -interval l
 $t_{\text{blend}}^{\text{min}}(p, \text{bl})$ = minimum running time required by blender bl when processing product p
 $t_{L3}(n)$ = duration of L3-period n
 $u^{\text{start}}(j, p)$ = product p stored in tank j at the beginning of the planning horizon
 $V_{\text{bc}}^{\text{max}}(i)$ = maximum holdup of tank with blend component i
 $V_{\text{bc}}^{\text{min}}(i)$ = minimum holdup of tank with blend component i
 $V_{\text{bc}}^{\text{start}}(j, p)$ = volume of blend component i stored at the beginning of the planning horizon
 $V_{\text{pr}}^{\text{max}}(j)$ = maximum holdup of tank j
 $V_{\text{pr}}^{\text{min}}(j)$ = minimum holdup of tank j
 $V_{\text{pr}}^{\text{start}}(j)$ = volume stored in tank j at the beginning of the planning horizon
 $\text{VMIN}_{\text{blend}}(p, \text{bl})$ = minimum volume allowed to blend of product p in blender bl during each L2-period

Integer variables

$t_{\text{blend}, L2}(p, \text{bl}, m)$ = estimated time to process product p in blender bl in L2-period m
 $u_{L2}(j, p, m)$ = binary variable that indicates if tank j is storing product p in L2-period m
 $ue_{L2}(j, m)$ = binary variable that indicates if there is a product transition in tank j at the beginning of L2-period m
 $v_{L3}(j, p, \text{bl}, n)$ = binary variable that indicates if tank j is receiving product p from blender bl in L3-period n
 $x_{L2}(p, \text{bl}, m)$ = binary variable that indicates if product p is processed in blender bl in L2-period m

Continuous variables

BlendCost_{L1} = total blend cost at the first level
 BlendCost_{L2} = total blend cost at the second level
 BlendCost_{L3} = total blend cost at the third level
 $D_{\text{pr}, L3}(j, o, n)$ = delivery rate of tank j for order o within L3-period n
 $F_{\text{blend}, L3}(p, \text{bl}, n)$ = blending rate of blender bl to produce product p during L3-period n
 $i_{\text{blend}, L3}(\text{bl}, n)$ = cumulative idle time of blender bl in L3-period n
 $of_{L2}(o, m)$ = fraction of order o to be delivered during L2-period m (it becomes a parameter at the third level model)
 $\text{Dsw}_{L3}(j, o, n)$ = difference between delivery rates to be penalized in objective function of the third level optimization phase
 $r(i, p, k)$ = volume of blend component i into product p in L1-period k (it becomes a parameter at the second and third level model)
 $S_{\text{bc}, L1}^+(i, k)$ = positive inventory slack variable of blend component i at L1-period k
 $S_{\text{bc}, L1}^-(i, k)$ = negative inventory slack variable of blend component i at L1-period k
 $S_{\text{bc}, L2}^+(i, m)$ = positive inventory slack variable of blend component i at L2-period m
 $S_{\text{bc}, L2}^-(i, m)$ = negative inventory slack variable of blend component i at L2-period m
 $S_{\text{bc}, L3}^+(i, n)$ = positive inventory slack variable of blend component i at L3-period n
 $S_{\text{bc}, L3}^-(i, n)$ = negative inventory slack variable of blend component i at L3-period n
 $S_{\text{order}, L3}^+(o, l)$ = positive delivery slack variable of order o at L -interval l
 $S_{\text{order}, L3}^-(o, l)$ = negative delivery slack variable of order o at L -interval l
 $S_{\text{pool}, L1}^+(p, k)$ = positive inventory slack variable of product pool p at L1-period k
 $S_{\text{pool}, L1}^-(p, k)$ = negative inventory slack variable of product pool p at L1-period k
 $S_{\text{pool}, L2}^+(p, m)$ = positive inventory slack variable of product pool p at L2-period m
 $S_{\text{pool}, L2}^-(p, m)$ = negative inventory slack variable of product pool p at L2-period m
 $S_{\text{pr}, L1}^+(i, k)$ = positive inventory slack variable of product tank j at L1-period k

$S_{\text{pr}, L1}^-(i, k)$ = negative inventory slack variable of product tank j at L1-period k
 $S_{\text{pr}, L2}^+(i, m)$ = positive inventory slack variable of product tank j at L2-period m
 $S_{\text{pr}, L2}^-(i, m)$ = negative inventory slack variable of product tank j at L2-period m
 $S_{\text{pr}, L3}^+(i, n)$ = positive inventory slack variable of product tank j at L3-period n
 $S_{\text{pr}, L3}^-(i, n)$ = negative inventory slack variable of product tank j at L3-period n
 $sw_{L3}(\text{bl}, n)$ = 0–1 continuous variable that indicates if a blend run has started in blender bl at the beginning of L3-period n
 $t_{\text{blend}, L3}(\text{bl}, n)$ = cumulative running time of blender bl in L3-period n
 $V_{\text{bc}, L2}(i, m)$ = volume stored in component tank i at the end of L2-period m
 $V_{\text{bc}, L3}(i, n)$ = volume stored in component tank i at the end of L3-period n
 $V_{\text{comp}, L1}(i, p, k)$ = volume of blend component i into product p in L1-period k
 $V_{\text{comp}, L2}(i, p, \text{bl}, m)$ = volume of blend component i into product p in blender bl in L2-period m
 $V_{\text{comp}, L3}(i, p, \text{bl}, n)$ = volume of blend component i into product p in blender bl in L3-period n
 $V_{\text{pr}, L2}(j, m)$ = volume stored in product tank j at the end of L2-period m
 $V_{\text{pr}, L3}(j, n)$ = volume stored in product tank j at the end of L3-period n
 $V_{\text{trans}, L3}(j, p, \text{bl}, n)$ = volume transferred of product p from blender bl to tank j in L3-period n
 $vc_{\text{blend}, L3}(\text{bl}, n)$ = cumulative volume produced by blender bl during a blend run up to period n
 $ve_{L3}(\text{bl}, n)$ = 0–1 continuous variable which indicates if the product from blender bl is going to be sent to a different storage tank than that of the previous L3-period
 $w_{\text{blend}, L3}(\text{bl}, n)$ = 0–1 continuous variable which indicates that blender bl is idle in L3-period n if it is equal to 1
 $x_{L3}(p, \text{bl}, n)$ = 0–1 continuous variable that indicates if product p is processed in blender bl in L3-period n
 $xe_{L2}(p, \text{bl}, m)$ = 0–1 continuous variable that indicates if a state transition has occurred in blender bl at the beginning of L2-period m
 $xe_{L3}(p, \text{bl}, n)$ = 0–1 continuous variable that indicates if a state transition has occurred in blender bl at the beginning of L3-period n
 Z_{L1} = objective function value at the first level
 Z_{L2} = objective function value at the second level, total cost
 Z_{L2}^{opt} = objective function value at the second level, approximate scheduling
 Z_{L2}^{feas} = objective function value at the second level, blend planning
 Z_{L3} = objective function value at the third level, total cost
 Z_{L3}^{feas} = objective function value at the third level, optimization phase
 Z_{L3}^{opt} = objective function value at the third level, feasibility phase

Literature Cited

- Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Ind Eng Chem Res.* 2003;42:825–835.
- Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. *AIChE J.* 2010;56:441–465.
- Mendez CA, Grossman IE, Harjunkoski I, Kabore P. A simultaneous optimization approach for off-line blending and scheduling of oil refinery operations. *Comput Chem Eng.* 2006;30:614–634.
- Li J, Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res.* 2011;50:9156–9174.
- Sousa RT, Liu S, Papageorgiou LG, Shah N. Global supply chain planning for pharmaceuticals. *Chem Eng Res Des.* 2011;89:2396–2409.

6. Pinto JM, Joly M, Moro LFL. Planning and scheduling models for refinery operations. *Comput Chem Eng.* 2000;24:2259–2276.
7. Kelly JD. Logistics: the missing link in blend scheduling optimization. *Hydrocarbon Process.* 2006;85:45–51.
8. Maravelias CT, Sung C. Integration of production planning and scheduling: overview, challenges and opportunities. *Comput Chem Eng.* 2009;33:1919–1930.
9. Maravelias CT. General framework and modeling approach classification for chemical production scheduling. *AIChE J.* 2012;58(6):1812–1828.
10. Mendez CA, Cerda J, Grossmann IE, Harjunkoski I, Fahl M. State-of-the-art review of optimization methods for short term scheduling of batch processes. *Comput Chem Eng.* 2006;30:913–946.
11. Floudas CA, Lin X. Continuous-time vs discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng.* 2004;28:2109–2129.
12. Sundaramoorthy A, Maravelias CT. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind Eng Chem Res.* 2011;50:5023–5040.
13. Joly M, Pinto JM. Mixed-integer programming techniques for the scheduling of fuel oil and asphalt production. *Inst Chem Eng.* 2003;81:427–447.
14. Bassett MH, Pekny JF, Reklaitis GV. Decomposition techniques for the solution of large-scale scheduling problems. *AIChE J.* 1996;42(12):3373–3387.
15. Elkamel A, Zentner M, Pekny F, Reklaitis GV. A decomposition heuristic for scheduling the general batch chemical plant. *Eng Optim.* 1997;28(4):299–330.
16. Munawar SA, Gudi RD. A multi-level, control-theoretic framework for integration of planning, scheduling and rescheduling. *Ind Eng Chem Res.* 2005;44:4001–4021.
17. Li Z, Ierapetritou MG. Integrated production planning and scheduling using a decomposition framework. *Chem Eng Sci.* 2009;64:3585–3597.
18. Mouret S, Grossmann IE, Pestaix P. A new Lagrangian decomposition approach applied to the integration of refinery planning and crude-oil scheduling. *Comput Chem Eng.* 2011;35:2750–2766.
19. Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. *Comput Chem Eng.* 2001;25:627–634.
20. Castillo PAC, Kelly JD, Mahalec V. Inventory pinch algorithm for gasoline blend planning. *AIChE J.* 2013;59:3748–3766.
21. Singhvi A, Shenoy UV. Aggregate planning in supply chains by pinch analysis. *Trans IChemE A.* 2002;80:597–605.

Manuscript received Oct. 24, 2013, and revision received Feb. 17, 2014.

Chapter 4: Inventory Pinch-Based Multi-Scale Model for Refinery Production Planning

This chapter has been published in the proceedings of the 24th European Symposium on Computer Aided Process Engineering (ESCAPE):

Castillo Castillo, P. A., & Mahalec, V. (2014). Inventory pinch based multi-scale model for refinery production planning. In J. J. Klemeš, P. S. Varbanov, & P. Y. Liew (Eds.), *Computer Aided Chemical Engineering* (Vol. 33, pp. 283-288). Budapest, Hungary: Elsevier. doi: 10.1016/B978-0-444-63456-6.50048-X

Permission from © Elsevier Ltd. All rights reserved.

The first three sections of Chapter 4 are an overview of Chapter 2. Section 4 presents an example of the MPIP algorithm from Chapter 2 applied to a refinery planning problem. Compared to the gasoline blending problem, the refinery planning problem considers different product pools (e.g., gasoline, diesel, kerosene). Therefore, the inventory pinch points are determined based on the cumulative product demand curves of each pool.

Inventory Pinch Based Multi-Scale Model for Refinery Production Planning

Pedro A. Castillo Castillo, Vladimir Mahalec*,

*Dept. of Chemical Engineering, McMaster University, 1280 Main Street West,
Hamilton, ON, L8S 4L8, Canada
mahalec@mcmaster.ca*

Abstract

In this work, we introduce a new two-level decomposition algorithm for production planning. The top level optimizes operating conditions via a nonlinear (NLP) model, while the lower level computes an optimal production plan by solving a mixed-integer linear (MILP) model which uses operating states computed at the top level. Time periods at the top level are delineated by inventory pinch points. The algorithm solves mixed-integer nonlinear (MINLP) models much faster than current MINLP solvers for our case studies associated with gasoline blend planning and refinery production planning.

Keywords: inventory pinch, production planning, two-level decomposition

1. Introduction

In order to increase profit margins, supply chain optimization is becoming a common tool in modern industry. Mathematical programming is one of the main techniques to carry out such optimization. The supply chain of an oil refinery is a complex network that has to face a dynamic market and strict regulations. In the last decades, several researchers have worked developing and/or improving models and algorithms to optimize such network. Menezes et al. (2013) developed a single-period MINLP planning model for crude-oil distillation units which computes solutions closer to those observed in practice since they consider swing-cuts to have different properties than the final-cuts. Alhajri et al. (2013) proposed a single-period MINLP model to optimize the production planning in a refinery while reducing the CO₂ emissions. Oddsdottir et al. (2013) presented an MINLP model for procurement planning for oil refineries that considers the blending operations occurring in the refinery unloading section. Assuming fixed blend recipes, Guajardo et al. (2013) developed an MILP model to integrate production and sales tactical planning in oil refineries. Simplified empirical nonlinear equations have been used in refinery planning models as well as hybrid models, i.e. models that use first-principle rigorous relations and empirical linear correlations together (e.g. Mahalec and Sanchez, 2012).

Multi-level decomposition techniques have been implemented to solve the integrated planning and scheduling problem. Li and Ierapetritou (2009) developed a bilevel decomposition algorithm to solve separately the production planning and scheduling levels and proposed an iterative procedure to solve it. Terrazas-Moreno and Grossmann (2011) presented a bilevel decomposition for integrated production planning and scheduling where the upper level is decomposed using a spatial Lagrangean relaxation in order to solve a planning subproblem for each plant site. Leiras et al. (2013) developed an iterative algorithm to integrate the mid-term and short-term planning of multirefinery

networks using a bilevel decomposition and the two-stage stochastic programming framework to handle uncertainty.

2. Inventory Pinch Based Planning Algorithm

This work introduces inventory pinch concept as a basis for refinery production planning using a two-level decomposition. An inventory pinch point on the cumulative total demand (CTD) curve is a point where cumulative average total production (CATP) curve intersects the CTD curve, so the CATP curve is above the CTD curve and if we extrapolate the CATP curve from this point onwards it will not cross the CTD curve (Castillo et al., 2013). The inventory pinch points delineate the periods where optimal operating states are likely to remain constant. This observation leads to the following approach to solving the planning problems:

- (i) Average production rates in intervals delimited by the pinch points are different from each other. In refinery planning the pinch points may be separated by 5 or 6 months or more.
- (ii) Since plant efficiency and raw material utilization depends on the throughput and the properties of the feeds, let's compute optimal operating conditions corresponding to each of these periods (for a given average throughput in each of the intervals).
- (iii) Compute more detailed production plan by using optimal operating conditions computed in (ii).

Above reasoning leads to our two-level decomposition algorithm, denoted as multi-period inventory pinch algorithm (MPIP), shown in Fig. 1. The full-space model (i.e. the original single model) is decomposed into two levels. The top level is modelled as a discrete-time, multi-period NLP model. It optimizes the operating conditions of the different type of processing units in the refinery and sets the production targets for the lower level. The periods of the top level are initially defined by the inventory pinch points. At the top level, storage tanks are aggregated into inventory pools of each product, and parallel units are aggregated into a single processing unit. The lower level is formulated as a discrete-time, multi-period MILP model. This level computes the detailed production plan (i.e. how much to produce in each unit and in each time period of the lower level). The lower level periods are defined by the planner. IPOPT solver is used at the top level, and CPLEX is used at the lower level.

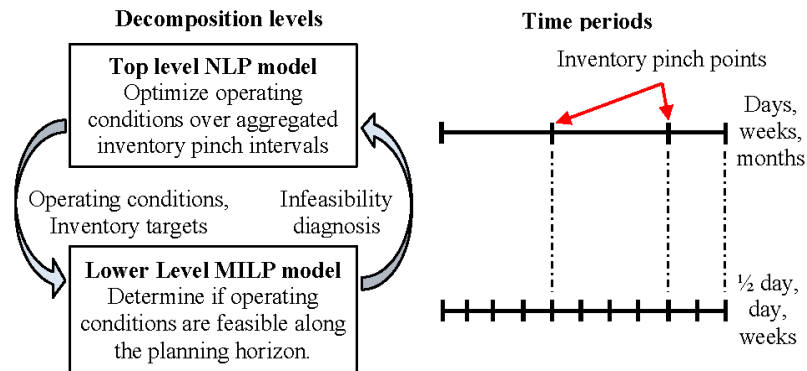


Figure 1. Multi-period Inventory Pinch (MPIP) decomposition.

3. Gasoline Blend Planning

Sample gasoline blending system is shown in Fig. 2. The storage tanks are defined as dedicated tanks (DT) if they can only hold one type of product during the entire planning horizon, or as swing tanks (ST) if they are allowed to hold different type of products along the horizon (but not at the same time). We have studied systems with 1 to 3 blenders operating in parallel. There are 9 quality properties; one of them, the Reid vapour pressure (RVP) blends nonlinearly according to Eq. (1) (used by Singh et al., 2000) where I is the set of blend components, J the set of products, and $x_{i,j}$ is the volume fraction of component i in product j . In addition, there is a minimum threshold constraint on each gasoline blend (this introduces integer variables). Length of time periods at the lower level is set to 1 day each and the planning horizon is 14 days. It is assumed that the length of the time periods at the lower level is equal to the minimum length of time that a swing tank would stay in any given service (if the minimum service time is shorter, then the periods at the lower level need also to be shorter, correspondingly). This enables us to compute (if required) allocation of swing tanks as a part of the production plan.

$$RVP_j = \left(\sum_{i \in I} x_{i,j} RVP_i^{1.25} \right)^{0.8} \quad \forall j \in J \quad (1)$$

Performance of inventory pinch based algorithm is compared to DICOPT, a current commercial MINLP solver. DICOPT is selected because, for our case studies, it requires shorter execution times than global MINLP solvers. Inventory pinch algorithm at the lower level has the same number of periods as the full-space MINLP model that is solved by DICOPT.

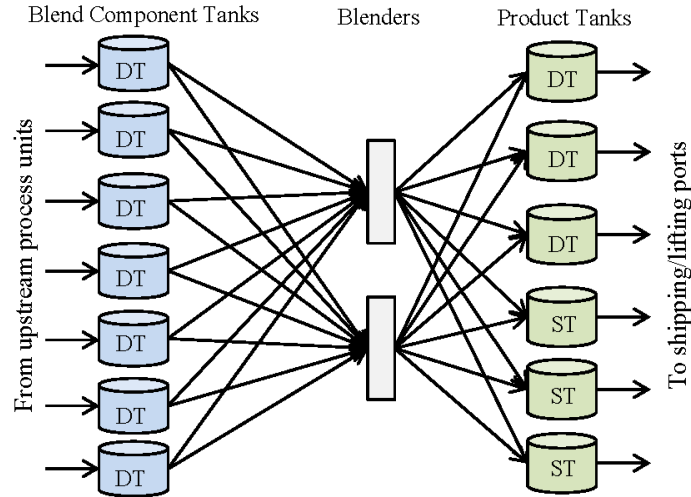


Figure 2. Sample gasoline blending system.

Table 1. Comparison of MPIP algorithm (IPOPT + CPLEX) and full-space MINLP (DICOPT).

Case study ID	# pinch points	# blenders	Blend Cost ($\times 10^3$ \$)	Full-space MINLP		MPIP Algorithm		
				# Recipes	CPU Time (s)	# Recipes	CPU Time (s)	# Its.
1	0	1	37,542.5	6	152.2	1	0.625	1
2	1	1	38,309.8	6	240.5	2	0.778	1
3	1	1	37,991.1	7	227.9	2	0.969	1
4	1	2	37,991.1	8	192.9	2	0.938	1
5	2	2	37,681.1	8	312.6	3	0.960	1
6	2	2	37,324.3	9	258.5	6	10.634	4
7	3	3	37,377.5	10	366.4	4	1.277	1

Results in Table 1 show that, for gasoline blend planning, MPIP algorithm is two orders of magnitude faster than DICOPT. Moreover, the number of different blend recipes is reduced significantly (a blend recipe is considered different if the absolute change of composition percentage of any component is greater than 1%). In addition, in most of our case studies the number of iterations (shown as “# Its.” in Table 1) required by the MPIP algorithm is one. Due to the form of the RVP nonlinear constraints used, it was possible to also create a linear version of these constraints, construct an MILP model and check if DICOPT and the MPIP algorithm reach the global optimum. Both DICOPT and MPIP algorithm compute the global optimum in these cases studies.

4. Refinery production planning

In many countries across the world, consumption of refining products varies as the seasons change. In the summer, there is a high demand for gasoline. In the winter, the demand for diesel fuel increases significantly, while the demand for gasoline decreases. A refinery meets such varying demands by shifting the operation of various process units from e.g. maximum gasoline to maximum diesel mode. Current practice in refinery production planning is to define several modes of operation for a process unit and define each mode of operation by a fixed set of yields and fixed product qualities. Since refinery units are nonlinear and since several units produce similar products, it is possible that the optimal operating states are different than those predetermined in advance and entered as parameters in the linear planning model. Hence, it is desirable to use nonlinear models for refinery planning.

Figure 3 shows configuration of the refinery used in our case studies. It is comprised of crude distillation unit (CDU), fluid catalytic cracking (FCC) unit, catalytic reformer (CR), and hydrotreaters (HT). The nomenclature of the streams is the following: sr = straight run, hc = hydrocracker, fcc = fluid catalytic cracker, ln = light naphtha, hn = heavy naphtha, ds = diesel, kero = kerosene, lgo = light oil, hgo = heavy oil, lco = light catalytic oil, hco = heavy catalytic oil, f = feed, lf = light feed, and hf = heavy feed. Only the blending operations in the gasoline and diesel pools are considered. Figure 4 shows cumulative curves for a sample refinery case study. There is a pinch point at period 8 for gasoline pool and a pinch point at period 4 for diesel pool. In one case study, MPIP

Inventory Pinch Based Multi-Scale Model for Refinery Production Planning

algorithm solves the planning problem in 3.03 sec, while corresponding MINLP problem is solved by DICOPT in 12.59 sec. Complete mathematical models and results will be presented in a full length paper.

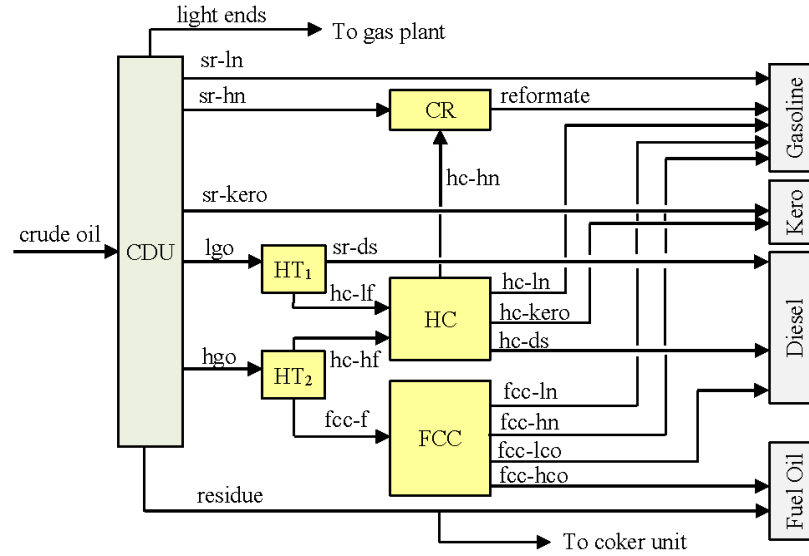


Figure 3. Sample refinery system.

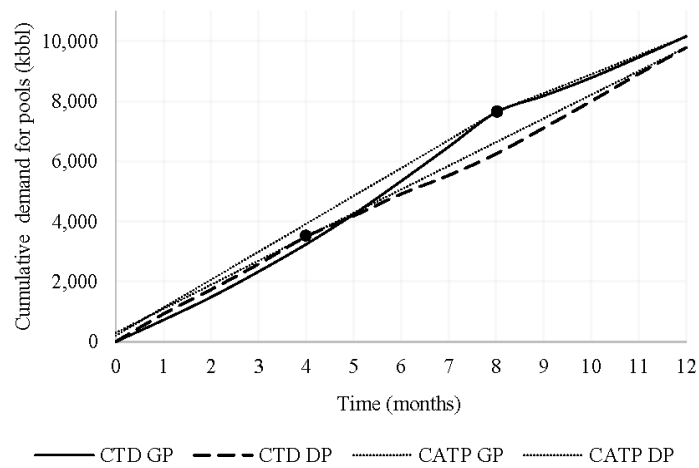


Figure 4. Cumulative curves and pinch points for gasoline pool (GP) and diesel pool (DP).

5. Conclusions

Inventory pinch provides a basis for two-level decomposition of the planning problems; this decomposition enables us to use nonlinear refinery models to solve nonlinear or mixed-integer nonlinear problems in a very efficient manner. Compared to the traditional, fixed-period planning models (e.g. 12 monthly periods for refinery planning, or 14 daily periods for gasoline blending), inventory pinch algorithm optimizes nonlinear models using significantly smaller number of periods. Moreover, decomposition introduced in the inventory pinch algorithm enables us to solve MINLP problems by first solving aggregate NLP problem and then solve an MILP problem. Application of the algorithm to the refinery operations planning case studies (gasoline blending, process units operations) shows that it leads to the same solutions as the multi-period MINLP models. Moreover, the solution times are one order of magnitude (or even more) faster than DICOPT MINLP solver.

References

- I. Alhajji, Y. Saif, A. Elkamel, A. Almansoori, 2013, Overall integration of the management of H₂ and CO₂ within refinery planning using rigorous process models, *Chemical Engineering Communications*, 200, 139-161.
- P.A.C. Castillo, J.D. Kelly, V. Mahalec, 2013, Inventory pinch algorithm for gasoline blend planning, *AIChE Journal*, 59, 3748-3766.
- K. Glismann, G. Gruhn, 2001, Short-term scheduling and recipe optimization of blending processes, *Computers and Chemical Engineering*, 25, 627-634.
- M. Guajardo, M. Kylinger, M. Ronnqvist, 2013, Specialty oils supply chain optimization: From a decoupled to an integrated planning approach, 229, 540-551.
- A. Leiras, G. Ribas, S. Hamacher, A. Elkamel, 2013, Tactical and operational planning of multirefinery networks under uncertainty: An iterative approach, *Industrial & Engineering Chemistry Research*, 52, 8507-8517.
- Z. Li, M.G. Ierapetritou, 2009, Integrated production planning and scheduling using a decomposition framework, *Chemical Engineering Science*, 64, 3585-3597.
- V. Mahalec, Y. Sanchez, 2012, Inferential monitoring and optimization of crude separation units via hybrid models, *Computers and Chemical Engineering*, 45, 15-26.
- B.C. Menezes, J.D. Kelly, I.E. Grossmann, 2013, Improved swing-cut modelling for planning and scheduling of oil-refinery distillation units, *Industrial & Engineering Chemistry Research*, 52, 18324-18333.
- T.A. Oddsdottir, M. Grunow, R. Akkerman, 2013, Procurement planning in oil refining industries considering blending operations, *Computers and Chemical Engineering*, 58, 1-13.
- A. Singh, J.F. Forbes, P.J. Vermeer, S.S. Woo, 2000, Model-based real-time optimization of automotive gasoline blending operations, *Journal of Process Control*, 10, 43-58.
- S. Terrazas-Moreno, I.E. Grossmann, 2011, A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants, *Chemical Engineering Science*, 66, 4307-4318.

Chapter 5: Improved Continuous-Time Model for Gasoline Blend Scheduling

This chapter has been published in the Computers and Chemical Engineering Journal.
Complete citation:

Castillo Castillo, P. A., & Mahalec, V. (2016). Improved continuous-time model for gasoline blend scheduling. *Computers & Chemical Engineering*, 84, 627–646. Elsevier Ltd., doi: 10.1016/j.compchemeng.2015.08.003

Permission from © Elsevier Ltd. All rights reserved.

In Chapter 5, the development of a continuous-time blend scheduling model is presented. As the problem size grows (e.g., more blenders, products, orders, and/or longer scheduling horizon), this model requires smaller number of binary variables than previous published model, while including more logistic constraints found in real practice. Although not all the examples were solved to proven optimality, the feasible solutions found were better than those previously reported in the literature.

This continuous-time blend scheduling model is used in Chapter 6 to improve the performance of the MPIP algorithm presented in Chapter 3.



Improved continuous-time model for gasoline blend scheduling



Pedro A. Castillo-Castillo, Vladimir Mahalec*

Department of Chemical Engineering, McMaster University, 1280 Main St. West, Hamilton, ON L8S 4L8, Canada

ARTICLE INFO

Article history:

Received 25 August 2014

Received in revised form 30 July 2015

Accepted 4 August 2015

Available online 13 August 2015

Keywords:

Gasoline blend scheduling

Continuous-time model

Reduced number of discrete variables

Nonlinear blending models

ABSTRACT

This work introduces a reduced-size continuous-time model for scheduling of gasoline blends. Previously published model has been modified by (i) introducing new model features (penalty for deliveries in order to reduce sending material from different product tanks to the same order, product and blender-dependent minimum setup times, maximum delivery rate from component tanks, threshold volume for each blend), (ii) by reducing the number of integer variables, and (iii) by adding lower bounds on the blend and switching costs, which significantly improve convergence. Nonlinearities are introduced by ethyl RT-70 equations for octane blending. Medium-size linear problems (two blenders, more than 20 orders, 5 products) are solved to optimality within one or two minutes. Previously unsolved large scale blending problems (more than 35 orders, 5 product, 2 or 3 blenders) have also been solved to less than 0.5% optimality gap.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Scheduling of a production plant involves making decisions to determine what and when the tasks must be executed, and in which processing units, in order to fulfill product demands along a given time horizon while meeting product specifications, storage and capacity constraints, and raw materials availability. Complex plants, such as oil-refineries, can have multiple processes, large number of raw materials with different quality properties, several intermediate and final products, and intricate pipeline and storage systems that make scheduling a difficult decision process. Determining the optimal production schedule can reduce operational costs, increase profit margins, and avoid deviations from environmental constraints (Harjunkoski et al., 2014). Mathematical programming is the most common approach to handle scheduling problems and several mixed-integer programming models and techniques have been developed and applied to solve scheduling problems in chemical production plants during the past three decades (Velez and Maravelias, 2015). However, due to the combinatorial nature of the problem, scheduling remains a research intensive and challenging area.

Scheduling models can be divided into those employed for batch processes and those utilized for continuous processes. Batch plant models do not require explicit material balances since material batches neither merge nor split and they can be tracked along the processing stages. On the other hand, material balances are required to be included in continuous plant models since material batches are allowed to merge and/or split. Another type of classification of scheduling models is based on the representation of time:

- Discrete-time models – The scheduling horizon is divided into time periods of known duration. Recently, Velez and Maravelias (2015) described a methodology to incorporate non-uniform time grids into a discrete-time model (i.e. the number of time periods is different for each process or unit).
- Continuous-time models – The scheduling horizon is partitioned into several time slots whose duration is not known in advance and it is calculated during the optimization procedure. One global time grid can be used (i.e. the variables representing the time slots are the same for all units), as well as unit-specific time grids (i.e. the variables representing the time slots are different for each unit). Fixed time periods can be incorporated as well, and time slots are assigned to them, i.e. time slots start and end within such periods (e.g. Mendez et al., 2006; Neiro et al., 2014). The boundaries of these fixed time periods can be delineated by intermediate product due dates or calendar days.

* Corresponding author. Tel.: +1 905 525 9140x26386.
E-mail address: mahalec@mcmaster.ca (V. Mahalec).

Nomenclature

Sets and indices

A = $\{\alpha\}$ different supply profiles of blend components
BL = $\{bl\}$ blenders
E = $\{e\}$ quality properties (e.g. research and motor octane number)
EV quality properties that blend linearly on a volumetric basis (subset of **E**)
EW quality properties that blend linearly on a weight basis (subset of **E**)
ENL quality properties that blend nonlinearly (subset of **E**)
I = $\{i\}$ blend components
J = $\{j\}$ product tanks
M = $\{m \mid 0, 1, \dots, M\}$ fixed time periods
MN = $\{(m, n)\}$ time slot n is assigned to time period m
NO = $\{n \mid 0, 1, \dots, N\}$ time slots assigned for the entire horizon
N1 = $\{n \mid 1, \dots, N\}$ subset of **NO**, does not include first time slot
N2 = $\{n \mid 0, 1, \dots, N-1\}$ subset of **NO**, does not include last time slot
N3 = $\{n \mid 1, \dots, N-1\}$ subset of **NO**, does not include first and last time slots
O = $\{o\}$ all demand orders
P = $\{p\}$ different products
Q = $\{\theta\}$ different quality profiles of blend components
AN = $\{(\alpha, n)\}$ time slot n has supply profile α
ANend = $\{(\alpha, n)\}$ time slot n is the last one under supply profile α
BP = $\{(p, bl)\}$ blender bl can produce product p
BJ = $\{(j, bl)\}$ blender bl can send material to product tank j
JO = $\{(j, o)\}$ product tank j can deliver order o
PO = $\{(p, o)\}$ order o consists of product p
QN = $\{(\theta, n)\}$ time slot n has quality profile θ
QNend = $\{(\theta, n)\}$ time slot n is the last one under quality profile θ
JON = $\{(j, o, n)\}$ product tank j can deliver order o during time slot n

Parameters

$c_1(i)$ unit cost of blend component i
 $c_2(bl)$ cost associated with one blend run in blender bl
 $c_3(j)$ cost associated with a product transition in swing tank j
 c_4 cost associated with a delivery run
 $c_5(o)$ demurrage cost for order o
 $c_6(n), c_7(n), c_8$ penalties for slack variables
 c_9 penalty to reduce blending rate variations in a blend run across multiple time slots
 $ct_{blend}^{\min}(p, bl)$ minimum setup time in blender bl for product p
 $ct_{blend}^{\min}(p, bl)$ minimum blend run length in blender bl for product p
 $CV_{blend}^{\min}(p, bl)$ minimum blend size in blender bl for product p
 $D_{order}^{\max}(o)$ maximum delivery rate of order o from any product tank
 $D_{pr}^{\max}(j)$ maximum delivery rate of product tank j
Demand(o) demand quantity of order o
 $F_{bc}(i, \alpha)$ flow rate of blend component i for supply profile α
 $F_{blend}^{\min}(bl), F_{blend}^{\max}(bl)$ minimum and maximum blending rates of blender bl
 $r_{comp}^{\max}(i, bl)$ maximum pumping rate of blend component tank i to blender bl
 $FT_{bc}^{end}(\alpha)$ time when supply profile α ends
 H length of the entire scheduling horizon
 $Q_{bc}(i, e, \theta)$ quality e of blend component i during quality profile θ
 $Q_{pr}^{\min}(p, e), Q_{pr}^{\max}(p, e)$ minimum and maximum specifications for quality property e and product p
 $QT_{bc}^{end}(\theta)$ time when quality profile θ ends
 $r^{\min}(i, p), r^{\max}(i, p)$ minimum and maximum composition of component i in product p
 sw_{exp} minimum number of blend runs expected in the blenders along the scheduling horizon
 $T^{start}(m), T^{end}(m)$ start and end time of period m
 $TO_{order}^{start}(o), TO_{order}^{end}(o)$ start and end time of delivery window for order o
 ue_{exp} minimum number of product changeovers expected in the swing tanks along the scheduling horizon
 $V_{bc}^{\min}(i), V_{bc}^{\max}(i)$ minimum and maximum inventory limits of blend component i
 $V_{pr}^{\min}(j), V_{pr}^{\max}(j)$ minimum and maximum inventory limits of product tank j
 $\rho(i)$ density of blend component i

Binary variables

$u(j, p, n)$ if equal to 1, product tank j is storing product p during slot n
 $v(j, bl, n)$ if equal to 1, blender bl is sending material to product tank j during slot n
 $z(j, o, n)$ if equal to 1, product tank j delivers material to order o during slot n

0–1 Continuous variables

$sw(bl, n)$ if equal to 1, a blend run is starting in blender bl at the end of slot n
 $w(bl, n)$ if equal to 1, blender bl is idle during slot n
 $x(p, bl, n)$ if equal to 1, blender bl is processing product p during slot n
 $xe(bl, n)$ if equal to 1, blender bl is changing its state at the end of slot n
 $y(i, bl, n)$ if equal to 1, component i is being fed to blender bl during slot n
 $ze(j, o, n)$ if equal to 1, product tank j change its state at the end of slot n (i.e. it starts or stops to deliver material)

Continuous variables

$ct_{blend}(bl, n)$ underestimator of the cumulative running time of blender bl at the end of slot n
 $cit_{blend}(bl, n)$ underestimator of the cumulative idle time of blender bl at the beginning of slot n
 $CV_{blend}(bl, n)$ underestimator of the cumulative volume blended up to slot n by blender bl
 $DV_{pr}(j, o, n)$ volume delivered from product tank j to order o during slot n
 $it_{blend}(bl, n)$ idle time of blender bl at time slot n
 $Q_{pr}(bl, e, n)$ quality e of the product from blender bl during slot n
 $S_{bc}^+(i, n), S_{bc}^-(i, n)$ positive and negative slack variables for the blend component inventories
 $S_{order}^+(o), S_{order}^-(o)$ positive and negative slack variables for each order fulfillment
 $S_{pr}^+(j, n), S_{pr}^-(j, n)$ positive and negative slack variables for the product inventories
 $T_b(bl, n)$ end time of a blender slot n
 $T_{bc}(i, n)$ end time of a component tank slot n
 $T_{pr}(j, n)$ end time of a product tank slot n
 $t_{blend}(bl, n)$ running time of blender bl at time slot n
 $td_{order}(j, o, n)$ time dedicated to deliver material from product tank j to order o during slot n
 $t_{dem}(o)$ time required to complete order o after the end of the corresponding delivery window
 $ts_{order}(j, o, n)$ time when product tank j starts to deliver material to order o during slot n
 $tt_{bc}(i, n)$ time when component tank i stops sending material to any blender during slot n
 $ue(j, n)$ if equal to 1, product tank j changes its service at the end of slot n
 $V_{bc}(i, n)$ inventory level of component tank i at the end of slot n
 $V_{blend}(bl, n)$ volume blended during slot n by blender bl
 $V_{comp,agg}(i)$ volume required of component i according to the aggregated model solution
 $V_{comp}(i, bl, n)$ volume of component i used in blender bl during slot n
 $V_{pr}(j, n)$ inventory level of product tank j at the end of slot n
 $V_{trans}(j, bl, n)$ volume transferred from blender bl to product tank j during slot n

In general, only one task can occur in a given processing unit during one time period or time slot. Each time representation has its advantages and disadvantages. Discrete-time models are much simpler to write than corresponding continuous-time models, but they usually require large number of time periods which increases their size (i.e. number of equations, continuous and discrete variables, etc.). Continuous-time models may require less number of discrete variables, but sometimes they could be less tight than corresponding discrete-time models (Joly and Pinto, 2003). Floudas and Lin (2004), Sundaramoorthy and Maravelias (2011), Maravelias (2012), and Harjunkoski et al. (2014) present more thorough reviews of discrete- and continuous-time scheduling models.

In this work, we present a continuous-time model for gasoline blend scheduling (a continuous process), which is an extensive modification of the previously published scheduling model by Li and Karimi (2011). This new version of the model adds additional operational constraints (minimization of deliveries of the same product from different tanks, limit on the maximum rate of delivery from component tanks, minimum volumetric size of a blend run, and product-dependent setup times), includes lower bounds on the objective function, contains equations to transform some binary variables into 0–1 continuous variables, and reduces the total number of binary variables in the model by making use of the demand information.

The paper is structured as follows: Section 2 is a brief overview of the prior work in this area, while Section 3 presents the problem statement. Section 4 describes the new version of the full-space blend scheduling model. Section 5 describes the examples used in this work. Section 6 presents computational results and it shows that the proposed lower bounds added to the full-space model significantly reduce solution times. As summarized in Section 7, the proposed new version of the full-space model can be solved much faster than the previously published model (Li and Karimi, 2011); however, large-scale problems still cannot be solved to proven optimality in reasonable time (e.g. <3 h).

2. Scheduling of gasoline blending operations

The gasoline blending process is a common example found in the literature on planning and scheduling optimization problems since gasoline is one of the main products of an oil refinery, both in quantity and profit margins (Mendez et al., 2006; Li and Karimi, 2011).

Moreover, the problem is easy to understand and its corresponding mathematical model can be formulated as an LP, NLP, MILP, or MINLP, depending of the assumptions, scope, and the required accuracy.

Jia and Ierapetritou (2003) constructed a continuous-time MILP model to schedule gasoline blending operations and its distribution. The model includes multipurpose product tanks (swing tanks), minimum run length requirements, and employs fixed recipes in order to maintain the linearity of the model.

Mendez et al. (2006) developed both a discrete- and continuous-time MILP model to schedule blending operations. They incorporate the off-line blending problem (i.e. recipe optimization) into the scheduling model, and they handle nonlinear blending rules through an iterative method. However, the distribution problem is not considered and not many logistic constraints are included (e.g. multipurpose tanks, non-identical blenders, minimum blend size, etc.).

Li et al. (2010) presented a continuous-time slot-based MILP model that uses process slots; i.e. a common global time grid for all units. This model includes blend recipe optimization, parallel non-identical blenders, multipurpose tanks, inventory constraints, blender capacity constraints, delivery scheduling for the demand orders, and other constraints found in industrial practice. Li and Karimi (2011) replaced process slots with unit slots (i.e. each unit has its specific time grid) and expanded the model by Li et al. (2010) to include blender setup times and simultaneous receipt/delivery by product tanks. Their execution times and final solutions were better than those from Li et al. (2010), but the large-scale case studies (e.g. 2–3 blenders, 5 products, 9 components, 9 quality properties, 11 product tanks, 35–45 orders, and a planning horizon of 8 days) still used all the allocated CPU time (46,800–118,800 s, depending on the problem) and the optimality gaps were not reported.

Kolodziej et al. (2013) formulated a discrete-time MINLP model for the pooling problem including inventory, flow, and quality constraints. The nonlinearities arise from the initial inventory of the blending tanks at each time period as a blend component. Since the quality properties of these initial inventories are not known after the first time period, the quality constraints involve bilinear nonconvex terms. In addition to one heuristic solution procedure, they proposed two different algorithms to solve this MINLP model to global optimality using a radix-based discretization technique which discretizes one variable in the bilinear terms found in the model in order to obtain MILP relaxations. Their largest case study has 8 tanks, 4 time periods (each 1 day long), and 2 quality properties. Their execution times range from a few seconds to 2840 s, depending on the problem instance.

3. Problem Statement

In this work, we address the gasoline blend scheduling problem stated as follows:

Given:

1. A short term scheduling horizon $[0, H]$.
2. A set of components, their properties, initial inventories, costs, and flow rates along the horizon (i.e. supply profile).
3. A set of products (i.e. gasoline grades) with prescribed minimum and maximum quality specifications, their initial inventories and corresponding initial quality.
4. A set of delivery orders for each product along the horizon (i.e. demand profile).
5. The maximum blending capacity of each blender.
6. A set of storage tanks and their minimum and maximum capacities.
7. Nonlinear or linear blending model.

The objective is to determine:

1. The blend recipes (i.e. the volume fractions of the blend components in each product) for each blend that will be made along the horizon.
2. The blenders that each component tank should feed over time, and the corresponding feed rates.
3. The products that each blender should produce over time, and their production rates.
4. The products that each product tank should receive over time, from which blender, and at what flow rates.
5. The orders that each product tank should deliver over time, their amounts and delivery rates.
6. The inventory profiles of component and product tanks.
7. Swing tanks product allocation along the horizon.

Minimizing:

1. Total cost which consists of the total cost of the blended materials plus the switching costs (i.e. number of blend runs, number of tanks delivering the same order, and product transitions in the swing tanks) and the demurrage costs.

Subject to:

1. If a blender is to produce a product, it must blend at least a minimum amount.
2. A blender can produce at most one product at any time. Once it begins blending, it must operate for some minimum time before it can switch to another product.
3. A blender requires a minimum setup time during a product changeover.
4. A blender can feed at most one product tank at any time (industrial practice).
5. Product tanks can only store one product at any time.

Assuming:

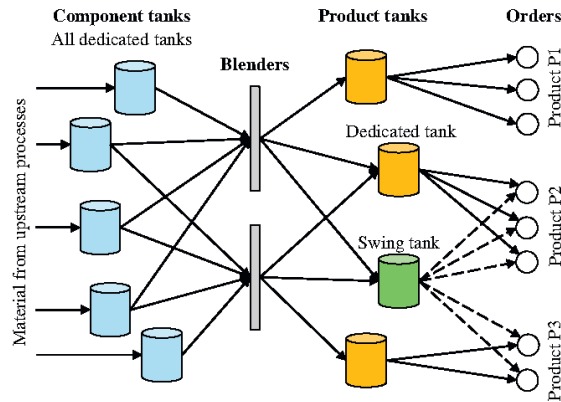


Fig. 1. Sample gasoline blending system.

1. Flow rate profile of each component from the upstream process is piecewise constant.
2. Component quality profile is piecewise constant.
3. Perfect mixing occurs in each blender.
4. There is only one tank for a given blend component.
5. There is no minimum flow restriction from component tanks to blenders.
6. Only product tanks defined as swing tanks can change its product service (i.e. change from storing one product to store another).
7. Changeover times between products are negligible for swing tanks.
8. For each blender, changeover times between product blending are product-dependent but sequence-independent.
9. Each order involves only one product (one original order involving different products can be broken into orders of each specific product).
10. Each order is completed during the scheduling horizon.

For illustration, Fig. 1 shows a sample blending system with five dedicated component tanks, two blenders, four product tanks (three dedicated tanks and one swing tank), and three different finished products.

4. Continuous-time full-space blend scheduling model

In this section, we present the continuous-time full-space blend scheduling model used in this work. This model is based on the one developed by Li and Karimi (2011), which was selected because it considers most of the key operational features of a blending system. We describe the differences between the two versions of this blend scheduling model. The notation used in this paper is not identical to the one presented by Li and Karimi (2011) since this model will be used in future work in combination with our previously published discrete-time models (Castillo and Mahalec, 2014a, 2014b). We denote as (LK-#) the equation number # presented by Li and Karimi (2011). We only present the equations for the case when there are different supply flow rates and quality profiles of blend components along the horizon (the multi-period model or MPM as denoted by Li and Karimi, 2011) since the case when they are constant can be handled by these equations as well.

One third of the equations are new, while two thirds are parts that are contained in the Li and Karimi model. The equations that are identical or equivalent to those presented by Li and Karimi (2011) are Eqs. (6)–(9), (11)–(18), (22)–(27), (29)–(35), (44)–(49), (54)–(60), (62)–(72), (77)–(82), (87)–(101), (103)–(108). The equations added or modified and presented in this work are Eqs. (1)–(5), (10), (19)–(21), (28), (36)–(43), (50)–(53), (61), (73)–(76), (83)–(86), (102), (109)–(113).

Three sets of modifications have been made to Li and Karimi (2011) model. Modification set #1 adds the following operational and model features:

- a) Minimization of deliveries of the same order from different product tanks (Eq. (3)).
- b) Minimization of number of blend runs by penalizing only the start of a blend run (Eqs. (3), (20), (21)).
- c) Addition of slack variables which ensure that a numerical solution is obtained, even if the problem is infeasible (Eqs. (1), (5), (61), (73), (74)).
- d) Minimum volume to be blended during a blend run (Eqs. (50)–(53)).
- e) Product- and blender-dependent minimum setup times (Eqs. (36)–(43)).
- f) Limit on the maximum delivery rate from the blend component tanks (Eqs. (75), (76), (83)–(86)).

Modification set #2 reduces the number of binary variables (Eqs. (19), (28), (102), and construction of refined sets for the delivery constraints).

Modification set #3 incorporates lower bounds on the blend cost and switching costs (Eqs. (109) and (110)).

Although these modifications significantly improve the solution times required by the full-space model, it can still take more than 3 h to solve to proven optimality medium and large-size problems.

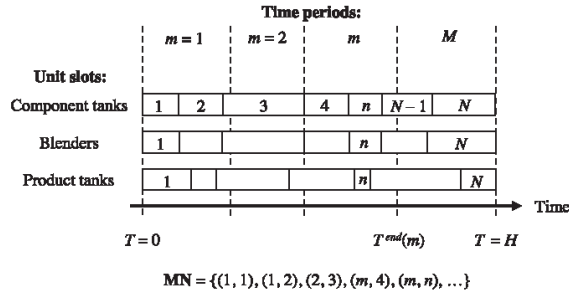


Fig. 2. Time slots and fixed time periods.

4.1. Time slots sets

From the original set of all time slots $\mathbf{N0} = \{n \mid n=0, 1, \dots, N-1, N\}$, three different subsets are constructed: set $\mathbf{N1}$ does not include the first slot (in this case, $n=0$), set $\mathbf{N2}$ does not include the last slot (in this case, $n=N$), and set $\mathbf{N3}$ does not include neither the first or last slot. Note that the time slots are “unit slots”, which means that each unit has its specific time grid. From here on, the terms time slot and unit slot will be used interchangeably.

4.2. Time periods and delivery slots sets

Binary variable $z(j,o,n)$ is equal to 1 if product tank j is sending material to order o during slot n , and 0 otherwise. Set $\mathbf{JO} = \{(j,o) \mid \text{tank } j \text{ may deliver order } o\}$ is composed of the pairs (j,o) found in the intersection of set $\mathbf{JP} = \{(j,p) \mid \text{tank } j \text{ can store product } p\}$ and set $\mathbf{PO} = \{(p,o) \mid \text{product } p \text{ constitutes order } o\}$. Writing the corresponding order delivery equations for all $(j,o) \in \mathbf{JO}$ and all $n \in \mathbf{N1}$ generates unnecessary instantiations of binary variable $z(j,o,n)$, which does not need to be introduced before the start time of the delivery window of the corresponding order, i.e. $TO_{order}^{start}(o)$; and in most cases, the orders are likely to be completed before or just a few hours after the end of the delivery window, i.e. $TO_{order}^{end}(o)$. Based on the demand information, and the allocation of time slots to fixed time periods along the scheduling horizon, it is possible to define set $\mathbf{ON} = \{(o,n) \mid \text{order } o \text{ may be delivered during time slot } n\}$ and set $\mathbf{JON} = \{(j,o,n) \mid \text{tank } j \text{ may deliver order } o \text{ during time slot } n\}$. Set \mathbf{JON} is constituted by the triplets (j,o,n) from the intersection of sets \mathbf{JO} and \mathbf{ON} . If set \mathbf{ON} contains less number of elements than $\{(o,n) \mid o \in \mathbf{O}, n \in \mathbf{N1}\}$, then the number of instantiations of variable $z(j,o,n)$ is reduced when using set \mathbf{JON} in the model instead of $\{(j,o,n) \mid (j,o) \in \mathbf{JO}, n \in \mathbf{N1}\}$.

In the MPM approach of Li and Karimi (2011), fixed time periods are defined by different supply flowrates or quality profiles of blend components; however, more time periods can be added, even when the supply flowrates and qualities of blend components are constant along the scheduling horizon. In this work, set $\mathbf{M} = \{1, \dots, m, \dots, M-1, M\}$ denotes the fixed time periods and set $\mathbf{MN} = \{(m, n) \mid \text{time slot } n \text{ is assigned to time period } m\}$ represents the allocation of time slots to fixed time periods. If $(m,n) \in \mathbf{MN}$, then slot n must end within period m (see Fig. 2). Boundaries of the time periods of set \mathbf{M} must include the times where changes are expected in the supply flowrates or qualities of the blend components. As mentioned before, more time periods can be added. However, this may increase the number of time slots required to solve the problem to optimality and therefore is only recommended if their addition decreases the total number of discrete variables in the model. For simplicity, in this work we only use the fixed time periods delineated by different supply flowrates of blend components.

Set \mathbf{ON} is constructed in the following way: If a portion of the time delivery window of order o lies within time period m , then order o may be delivered during the slots assigned to such time period. For the examples presented in this work, set \mathbf{JON} reduces 30–58% of the total number of discrete variables in the model.

4.3. Objective function

Objective function (LK-40) was replaced by Eqs. (1)–(5). The objective function (1) minimizes the blend cost (i.e. materials cost), the switching cost associated with each blend run, product changeovers in the swing tanks, and “delivery runs” (i.e. the number of time slots used to deliver an specific order from a given tank), the demurrage cost, and the penalty cost associated with the slack variables. Slack variables are included in the model in order to ensure there is always a numerical solution. Penalty coefficients $c_6(n)$, $c_7(n)$, and c_8 are much greater than the other cost coefficients to ensure that the slack variables have a 0 value in the final solution (if a physical feasible solution exists). Penalty coefficients $c_6(n)$ and $c_7(n)$ decrease with time in order to force the non-zero slack variables to appear as late as possible in the scheduling horizon. Delivery runs (i.e. slots with $z(j,o,n) = 1$) are penalized using the summation of the terms $c_4 \cdot z(j,o,n)$ in Eq. (3) in order to minimize deliveries to the same order from several tanks at the same time, and to minimize intermittent deliveries of the same order from the same tank. Number of blend runs are penalized with the summation of the terms $c_2(bl) \cdot sw(bl,n)$ in Eq. (3). Note that in Eq. (3), switching operations in the blenders and product tanks are penalized if they occur in time slot 0, in contrast with Eq. (LK-40) which does not penalizes switching operations at time slot 0 (i.e. at the beginning of the horizon).

$$\min(\text{Blend cost} + \text{Switching cost} + \text{Demurrage cost} + \text{Slacks cost}) \quad (1)$$

$$\text{Blend cost} = \sum_{n \in \mathbf{N1}} \sum_{bl} \sum_i c_1(i) \cdot V_{comp}(i, bl, n) \quad (2)$$

$$\text{Switching cost} = \sum_{n \in \mathbf{N2}} \left(\sum_{bl} c_2(bl) \cdot sw(bl, n) + \sum_j c_3(j) \cdot ue(j, n) \right) + \sum_{n \in \mathbf{N1}} \left(\sum_{(j,o):(j,o,n) \in \mathbf{JON}} c_4 \cdot z(j, o, n) \right) \quad (3)$$

$$\text{Demurrage cost} = \sum_{n \in \mathbf{N1}} \sum_{o \in \mathbf{O}} c_5(o) \cdot tdem(o) \quad (4)$$

$$\text{Slacks cost} = \sum_{n \in \mathbf{N1}} \left[\sum_i c_6(n) \cdot (S_{bc}^+(i, n) + S_{bc}^-(i, n)) + \sum_j c_7(n) \cdot (S_{pr}^+(j, n) + S_{pr}^-(j, n)) \right] + \sum_{o \in \mathbf{O}} c_8 \cdot (S_{order}^+(o) + S_{order}^-(o)) \quad (5)$$

4.4. Time slots definition

Variable $T_{unit}(unit_index, n)$ represents the end time of the unit slot n . These times are bounded by the beginning and end of the associated time period, i.e. $T^{start}(m) \leq T_{bc}(i, n), T_{pr}(j, n), T_b(bl, n) \leq T^{end}(m)$, for all m, i, j, bl , and $n: (m, n) \in \mathbf{MN}$. Eqs. (6)–(8) state that the end time of a unit slot must be equal to or greater than the end time of the previous unit slot. The initial unit slots are constrained to be equal to 0 by Eq. (9), and the last unit slots are forced to be equal to the length of the scheduling horizon by Eq. (10).

$$T_{bc}(i, n) \geq T_{bc}(i, n - 1) \quad \forall i, n \in \mathbf{N1} \quad (6)$$

$$T_{pr}(j, n) \geq T_{pr}(j, n - 1) \quad \forall j, n \in \mathbf{N1} \quad (7)$$

$$T_b(bl, n) \geq T_b(bl, n - 1) \quad \forall bl, n \in \mathbf{N1} \quad (8)$$

$$T_{bc}(i, n) = T_{pr}(j, n) = T_b(bl, n) = 0 \quad \forall i, j, bl, n = 0 \quad (9)$$

$$T_{bc}(i, n) = T_{pr}(j, n) = T_b(bl, n) = H \quad \forall i, j, bl, n = N \quad (10)$$

4.5. Binary and 0–1 continuous variables

Binary variable $v(j, bl, n)$ is 1 if product tank j is receiving material from blender bl in slot n , and 0 otherwise. 0–1 continuous variable $x(p, bl, n)$ is 1 if product p is being produced by blender bl in slot n , and 0 otherwise. 0–1 continuous variable $w(bl, n)$ represents the status of blender bl during slot n and it is equal to 0 (if the blender is running) or 1 (if the blender is idle). A blender can only process one product and can only feed one product tank in each slot, see Eqs. (11) and (12).

$$w(bl, n) + \sum_{p:(p, bl) \in \mathbf{BP}} x(p, bl, n) = 1 \quad \forall bl, n \in \mathbf{N1} \quad (11)$$

$$w(bl, n) + \sum_{j:(j, bl) \in \mathbf{BJ}} v(j, bl, n) = 1 \quad \forall bl, n \in \mathbf{N1} \quad (12)$$

Binary variable $u(j, p, n)$ is 1 if product tank j is holding product p during slot n , and 0 otherwise. Eq. (13) constrains a product tank to hold not more than one specific product type in each slot. If a blender produces product p in slot n , there must at least one product tank j holding that particular product type in that slot, and there must exist a physical connection between the blender and such product tank, as stated by Eqs. (14) and (15). Li et al. (2010) showed that Eqs. (14) and (15) make $x(p, bl, n)$ a 0–1 continuous variable.

$$\sum_{p:(j,p) \in \mathbf{JP}} u(j, p, n) = 1 \quad \forall j, n \in \mathbf{N1} \quad (13)$$

$$x(p, bl, n) \geq u(j, p, n) + v(j, bl, n) - 1 \quad \forall (p, bl) \in \mathbf{BP}, (j, p) \in \mathbf{JP}, (j, bl) \in \mathbf{BJ}, n \in \mathbf{N1} \quad (14)$$

$$x(p, bl, n) \leq u(j, p, n) - v(j, bl, n) + 1 \quad \forall (p, bl) \in \mathbf{BP}, (j, p) \in \mathbf{JP}, (j, bl) \in \mathbf{BJ}, n \in \mathbf{N1} \quad (15)$$

0–1 continuous variable $xe(bl, n)$ represents a state transition in blender bl at the end of slot n , i.e. a transition from being running to being idle, or vice versa, see Eqs. (16)–(19).

$$xe(bl, n) \geq x(p, bl, n) - x(p, bl, n + 1) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N2} \quad (16)$$

$$xe(bl, n) \geq x(p, bl, n + 1) - x(p, bl, n) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N2} \quad (17)$$

$$xe(bl, n) \leq 2 - x(p, bl, n) - x(p, bl, n + 1) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N2} \quad (18)$$

$$xe(bl, n) \leq 2 - w(bl, n) - w(bl, n + 1) \quad \forall bl, n \in \mathbf{N2} \quad (19)$$

Continuous variable $sw(bl, n)$ represents the start of a blend run in blender bl at the beginning of slot $n + 1$. $sw(bl, n)$ is constrained to have values between 0 and 1 by Eqs. (20) and (21), but since it is penalized in the objective function (1), it behaves like a 0–1 continuous variable. Li and Karimi (2011) did not use a variable similar to $sw(bl, n)$ because they penalize corresponding variable $xe(bl, n)$. However, when including non-zero setup times, it is possible that a blender requires an idle slot between blend runs; we do not want to penalize the transition from running to idle, and then from idle to running (a double penalty). By using variable $sw(bl, n)$, we avoid a double penalty for a single blend run.

$$sw(bl, n) \geq xe(bl, n) - w(bl, n + 1) \quad \forall bl, n \in \mathbf{N2} \quad (20)$$

$$sw(bl, n) \leq \frac{(xe(bl, n) + w(bl, n))}{2} \quad \forall bl, n \in \mathbf{N2} \quad (21)$$

Continuous variable $ue(j, n)$ represents a product transition in swing tank j at the end of slot n if it is equal to 1. This variable is defined by Eqs. (22) and (23). $ue(j, n)$ is bounded to be equal to or less than 1 (i.e. $ue(j, n) \leq 1, \forall j, n$).

$$ue(j, n) \geq u(j, p, n) - u(j, p, n + 1) \quad \forall (j, p) \in \mathbf{JP}, n \in \mathbf{N2} \quad (22)$$

$$ue(j, n) \geq u(j, p, n + 1) - u(j, p, n) \quad \forall (j, p) \in \mathbf{JP}, n \in \mathbf{N2} \quad (23)$$

Eq. (24) states that a product tank j cannot deliver order o if it does not hold the corresponding product p . Using Eq. (25), a product tank cannot receive and deliver product at the same time.

$$z(j, o, n) \leq u(j, p, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON}, p : (p, o) \in \mathbf{PO} \quad (24)$$

$$u(j, bl, n) + z(j, o, n) \leq 1 \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON}, bl : (j, bl) \in \mathbf{BJ} \quad (25)$$

0–1 continuous variable $y(i, bl, n)$ is equal to 1 if blend component i is being used in blender bl during slot n , and 0 otherwise. Eq. (26) ensures that if a blender is idle, it cannot receive material. If a blender is running, it must receive at least one blend component, as stated by Eq. (27). Including Eq. (28) in the model makes $y(i, bl, n)$ a 0–1 continuous variable. However, Eq. (28) synchronizes most of the component tank slots with the blender slots, and hence, the number of time slots required to solve the problem to optimality may increase. This has been observed particularly in small-scale problems (see Section 6). Nevertheless, computed solutions are still of the same quality and, for medium- and large-scale problems, number of binary variables does not increase and the execution times are shorter than when not including Eq. (28) and using $y(i, bl, n)$ as a binary variable. If a minimum pumping rate from the tanks to the blenders is imposed (constraint that it is not considered in this work), Eq. (28) should not be included in the model and $y(i, bl, n)$ must be defined as a binary variable.

$$w(bl, n) + y(i, bl, n) \leq 1 \quad \forall i, bl, n \in \mathbf{N1} \quad (26)$$

$$w(bl, n) + \sum_i y(i, bl, n) \geq 1 \quad \forall bl, n \in \mathbf{N1} \quad (27)$$

$$y(i, bl, n) = \sum_{p:(p,bl) \in BP} x(p, bl, n) \quad \forall i, bl, n \in \mathbf{N1} \quad (28)$$

Therefore, only three binary variables are required by this model: $u(j, p, n)$, $v(j, bl, n)$ and $z(j, o, n)$.

4.6. Blending-time constraints

A blender must process a product at least for a minimum amount of time given by parameter $ct_{blend}^{\min}(p, bl)$, see Eqs. (29)–(35). Continuous variables $t_{blend}(bl, n)$ and $ct_{blend}(bl, n)$ represent the blending time during slot n and the cumulative blending time (since the start of the current blend run) at the beginning of slot n , for blender bl , respectively. Eq. (29) constrains $t_{blend}(bl, n)$ to be equal to or less than the blender slot duration, while Eq. (30) forces the equality unless the blender is idle or is at the last slot of a blend run. Eq. (31) forces $t_{blend}(bl, n)$ to be equal to 0 if the blender is idle.

$$t_{blend}(bl, n) \leq T_b(bl, n) - T_b(bl, n - 1) \quad \forall bl, n \in \mathbf{N1} \quad (29)$$

$$t_{blend}(bl, n) \geq T_b(bl, n) - T_b(bl, n - 1) - H \cdot (w(bl, n) + xe(bl, n)) \quad \forall bl, n \in \mathbf{N1} \quad (30)$$

$$t_{blend}(bl, n) \leq H \cdot (1 - w(bl, n)) \quad \forall bl, n \in \mathbf{N1} \quad (31)$$

Eq. (32) restricts $ct_{blend}(bl, n)$ to be equal to or less than the cumulative time at the end of slot $n - 1$ plus the blending time during slot n . If a blender changes its state, the cumulative time is reset to 0 by Eq. (33). A blender that is running during slot n can only change its state in $n + 1$ if the cumulative blending time up to n is greater than the minimum specified, see Eq. (34). Eq. (35) is a version of Eq. (34) specified for the last slot of the horizon.

$$ct_{blend}(bl, n) \leq ct_{blend}(bl, n - 1) + t_{blend}(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (32)$$

$$ct_{blend}(bl, n) \leq H \cdot (1 - xe(bl, n)) \quad \forall bl, n \in \mathbf{N1} \quad (33)$$

$$ct_{blend}(bl, n - 1) + t_{blend}(bl, n) \geq \sum_{p:(p,bl) \in BP} ct_{blend}^{\min}(p, bl) \cdot x(p, bl, n) - RL_{blend}(bl) \cdot (1 - xe(bl, n)) \quad \forall bl, n \in \mathbf{N3} \quad (34)$$

$$ct_{blend}(bl, n - 1) + t_{blend}(bl, n) \geq \sum_{p:(p,bl) \in BP} ct_{blend}^{\min}(p, bl) \cdot x(p, bl, n) \quad \forall bl, n = N \quad (35)$$

where $RL_{blend}(bl) = \max_{p:(p,bl) \in BP} ct_{blend}^{\min}(p, bl) \quad \forall bl$.

Before a new blend run, a blender requires an idle time at least equal to the setup time given by parameter $cit_{blend}^{\min}(p, bl)$, see Eqs. (36)–(43). Variable $it_{blend}(bl, n)$ represents the idle time of blender bl during slot n , and variable $cit_{blend}(bl, n)$ is the cumulative idle time (since the end of the last blend run) of blender bl at the beginning of slot n . Eq. (36) constrains $it_{blend}(bl, n)$ to be less than the blender slot duration minus the blending time, and Eq. (37) forces $it_{blend}(bl, n)$ to be equal to 0 unless the blender is idle or is at the last slot of a blend run. Eq. (38) constraints $it_{blend}(bl, n)$ to be equal to 0 during the last time slot unless the blender is idle.

$$it_{blend}(bl, n) + t_{blend}(bl, n) \leq T_b(bl, n) - T_b(bl, n - 1) \quad \forall bl, n \in \mathbf{N1} \quad (36)$$

$$it_{blend}(bl, n) \leq H \cdot (xe(bl, n) + w(bl, n)) \quad \forall bl, n \in \mathbf{N1} \quad (37)$$

$$it_{blend}(bl, n) \leq H \cdot w(bl, n) \quad \forall bl, n = N \quad (38)$$

The cumulative idle time at slot n is equal to or less than the idle time from slot $n - 1$ plus the idle time during slot n , see Eq. (39). When a blender is running, the cumulative idle time $cit_{blend}(bl, n)$ is reset to 0 by Eq. (40). The cumulative idle time must be at least the current idle time at slot n , this is imposed by Eqs. (41) and (42). An idle blender can only change its state after the minimum idle time has been completed, see Eq. (43).

$$cit_{blend}(bl, n) \leq cit_{blend}(bl, n - 1) + it_{blend}(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (39)$$

$$cit_{blend}(bl, n) \leq H \cdot (xe(bl, n) + w(bl, n)) \quad \forall bl, n \in \mathbf{N1} \quad (40)$$

$$cit_{blend}(bl, n) \geq it_{blend}(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (41)$$

$$cit_{blend}(bl, n) \leq it_{blend}(bl, n) + H \cdot w(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (42)$$

$$cit_{blend}(bl, n) \geq cit_{blend}^{\min}(p, bl) \cdot (xe(bl, n) - w(bl, n + 1)) - IL_{blend}(bl) \cdot (1 - x(p, bl, n + 1)) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N2} \quad (43)$$

where $IL_{blend}(bl) = \max_{p:(p,bl) \in BP} cit_{blend}^{\min}(p, bl) \quad \forall bl$.

4.7. Blending-material balance equations

Volume processed by a blender is the same as the volume coming from the component tanks and the volume transferred to the product tanks, see Eqs. (44) and (45). Only one product tank can receive product within a time slot, as Eqs. (12), (46) enforce. The volume from each component to the blender is limited by the pumping rate of each component tank or the blender capacity, see Eq. (47).

$$V_{blend}(bl, n) = \sum_i V_{comp}(i, bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (44)$$

$$V_{blend}(bl, n) = \sum_{j:(j,bl) \in \mathbf{BJ}} V_{trans}(j, bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (45)$$

$$V_{trans}(j, bl, n) \leq F_{blend}^{\max}(bl) \cdot H \cdot v(j, bl, n) \quad \forall (j, bl) \in \mathbf{BJ}, n \in \mathbf{N1} \quad (46)$$

$$V_{comp}(i, bl, n) \leq \beta(i, bl) \cdot H \cdot y(i, bl, n) \quad \forall i, bl, n \in \mathbf{N1} \quad (47)$$

where $\beta(i, bl) = \min \{F_{comp}^{\max}(i, bl), F_{blend}^{\max}(bl)\} \quad \forall i, bl$.

4.8. Blending-volume threshold constraints

The volume produced by a blender must be between its minimum and maximum production rates, according to Eqs. (48) and (49).

$$V_{blend}(bl, n) \geq F_{blend}^{\min}(bl) \cdot t_{blend}(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (48)$$

$$V_{blend}(bl, n) \leq F_{blend}^{\max}(bl) \cdot t_{blend}(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (49)$$

A blend run must end after a minimum volume given by parameter $CV_{blend}^{\min}(p, bl)$ has been produced, see Eqs. (50)–(53). Variable $CV_{blend}(bl, n)$ is the cumulative blended volume (since the start of the current blend run) at the beginning of slot n in blender bl . $CV_{blend}(bl, n)$ is constrained by Eq. (50) to be equal to or less than the cumulative volume from slot $n - 1$ plus the volume blended during slot n . $CV_{blend}(bl, n)$ is reset to 0 by Eq. (51) if a blender changes its state. Eqs. (52) and (53) permit a blender to finish a blend run only after the minimum volume has been produced.

$$CV_{blend}(bl, n) \leq CV_{blend}(bl, n - 1) + V_{blend}(bl, n) \quad \forall bl, n \in \mathbf{N1} \quad (50)$$

$$CV_{blend}(bl, n) \leq F_{blend}^{\max}(bl) \cdot H \cdot (1 - xe(bl, n)) \quad \forall bl, n \in \mathbf{N1} \quad (51)$$

$$CV_{blend}(bl, n - 1) + V_{blend}(bl, n) \geq \sum_{p:(p,bl) \in BP} (CV_{blend}^{\min}(p, bl) \cdot x(p, bl, n)) - VL_{blend}(bl) \cdot (1 - xe(bl, n)) \quad \forall bl, n \in \mathbf{N3} \quad (52)$$

$$CV_{blend}(bl, n - 1) + V_{blend}(bl, n) \geq \sum_{p:(p,bl) \in BP} (CV_{blend}^{\min}(p, bl) \cdot x(p, bl, n)) \quad \forall bl, n = N \quad (53)$$

where $VL_{blend}(bl) = \max_{p:(p,bl) \in BP} CV_{blend}^{\min}(p, bl) \quad \forall bl$.

4.9. Blending-composition constraints

Eqs. (54) and (55) ensure that the amount of blend components per unit of product are within the minimum and maximum composition limits, respectively.

$$V_{comp}(i, bl, n) \geq r^{\min}(i, p) \cdot V_{blend}(bl, n) - \beta(i, bl) \cdot H \cdot (r^{\min}(i, p) - \kappa^{\min}(i)) \cdot (1 - x(p, bl, n)) \quad \forall i, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1} \quad (54)$$

$$V_{comp}(i, bl, n) \leq r^{\max}(i, p) \cdot V_{blend}(bl, n) + \beta(i, bl) \cdot H \cdot (\kappa^{\max}(i) - r^{\max}(i, p)) \cdot (1 - x(p, bl, n)) \quad \forall i, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1} \quad (55)$$

where $\kappa^{\min}(i) = \min_p r^{\min}(i, p) \quad \forall i$ and $\kappa^{\max}(i) = \max_p r^{\max}(i, p) \quad \forall i$.

4.10. Blending-quality balance equations

One of the assumptions in the Problem Statement (Section 3) is a piecewise constant component quality profile. Parameter $Q_{bc}(i, e, \theta)$ represents the value of quality property e for blend component i during quality profile θ . Set $\mathbf{QN} = \{(\theta, n)\}$ indicates which time slots correspond to each quality profile. The quality properties that blend linearly on a volumetric basis are constrained by Eqs. (56) and (57) to ensure that the final product quality is within the minimum and maximum specifications, respectively. Eqs. (58) and (59) are analogous for the quality properties that blend linearly on a weight basis.

$$\sum_i (V_{comp}(i, bl, n) \cdot Q_{bc}(i, e, \theta)) \geq Q_{pr}^{\min}(p, e) \cdot V_{blend}(bl, n) - F_{blend}^{\max}(bl) \cdot H \cdot (Q_{pr}^{\min}(p, e) - \xi^{\min}(e)) \cdot (1 - x(p, bl, n)) \quad (56)$$

$\forall e \in \mathbf{EV}, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN}$

$$\sum_i (V_{comp}(i, bl, n) \cdot Q_{bc}(i, e, \theta)) \leq Q_{pr}^{\max}(p, e) \cdot V_{blend}(bl, n) + F_{blend}^{\max}(bl) \cdot H \cdot (\xi^{\max}(e) - Q_{pr}^{\max}(p, e)) \cdot (1 - x(p, bl, n)) \quad (57)$$

$\forall e \in \mathbf{EV}, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN}$

where $\xi^{\min}(e) = \min_p Q_{pr}^{\min}(p, e) \quad \forall e$, and $\xi^{\max}(e) = \max_p Q_{pr}^{\max}(p, e) \quad \forall e$.

$$\sum_i (V_{comp}(i, bl, n) \cdot Q_{bc}(i, e, \theta) \cdot \rho(i, \theta)) \geq Q_{pr}^{\min}(p, e) \cdot \sum_i (V_{comp}(i, bl, n) \cdot \rho(i, \theta)) - F_{blend}^{\max}(bl) \cdot H \cdot (Q_{pr}^{\min}(p, e) - \xi^{\min}(e)) \cdot \rho^{\max} \cdot (1 - x(p, bl, n)) \quad (58)$$

$\forall e \in \mathbf{EW}, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN}$

$$\sum_i (V_{comp}(i, bl, n) \cdot Q_{bc}(i, e, \theta) \cdot \rho(i, \theta)) \leq Q_{pr}^{\max}(p, e) \cdot \sum_i (V_{comp}(i, bl, n) \cdot \rho(i, \theta)) + F_{blend}^{\max}(bl) \cdot H \cdot (\xi^{\max}(e) - Q_{pr}^{\max}(p, e)) \cdot \rho^{\max} \cdot (1 - x(p, bl, n)) \quad (59)$$

$\forall e \in \mathbf{EW}, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN}$

where $\rho^{\max} = \max_{i, \theta} \rho(i, \theta)$.

Eq. (60) constrains the last component tank slots with a given quality profile to coincide with the time when such quality profile ends. This equation assumes that the first quality profile starts at time equal to 0. Eq. (60) is required to avoid using material of a given quality that is not available anymore, or not available yet. See Fig. 3 for an example of Eq. (60).

$$T_{bc}(i, n) = QT_{bc}^{end}(\theta) \quad \forall i, \theta, n : (\theta, n) \in \mathbf{QNend} \quad (60)$$

4.11. Order delivery constraints

Li and Karimi (2011) used fixed delivery rates for each order (i.e. if a tank is sending product to meet order o , it must do it at the rate specified a priori for order o). In our model, if the delivery rate for each order is not known, we let the delivery rate vary between maximum and minimum rates, denoted as $D_{order}^{\max}(o)$ and $D_{order}^{\min}(o)$, respectively. A fixed delivery rate is obtained by making both of these parameters equal.

Eq. (61) constrains the amount of volume delivered from the product tanks to the shipping/lifting ports, plus positive and negative slack variables, is equal to the demand. By Eq. (62), the total delivery rate of tank j must be smaller than its maximum possible. The delivery rate of product tank j to satisfy order o must be smaller than the maximum delivery rate of such tank, and the maximum and minimum delivery rates specified for order o , according to Eqs. (63)–(65), respectively. If tank j is delivering order o during slot n , then binary variable $z(j, o, n)$ is equal to 1 by Eq. (66).

$$\left[\sum_{n \in \mathbf{N1}} \left(\sum_{j: (j, o, n) \in \mathbf{JON}} DV_{pr}(j, o, n) \right) \right] + S_{order}^+(o) - S_{order}^-(o) = Demand(o) \quad \forall o \in \mathbf{O} \quad (61)$$

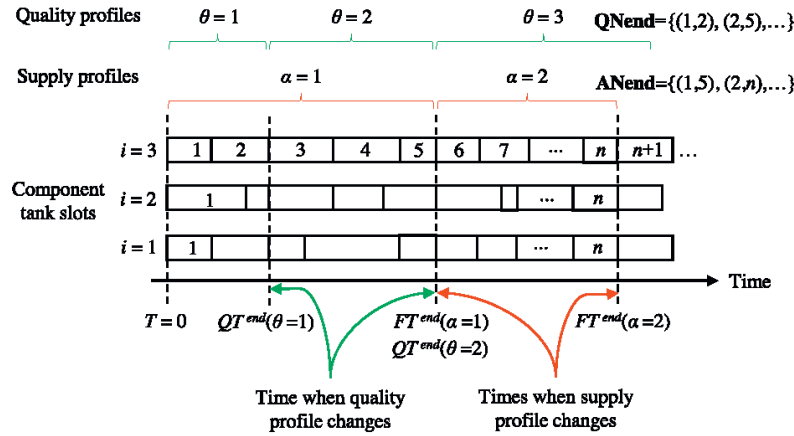


Fig. 3. Graphical example of Eqs. (60) and (81).

$$\sum_{o:(j,o,n) \in \text{JON}} DV_{pr}(j, o, n) \leq D_{pr}^{\max}(j) \cdot (T_{pr}(j, n) - T_{pr}(j, n-1)) \quad \forall j, n \in \mathbf{N1} \quad (62)$$

$$DV_{pr}(j, o, n) \leq D_{pr}^{\max}(j) \cdot td_{order}(j, o, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (63)$$

$$DV_{pr}(j, o, n) \leq D_{order}^{\max}(o) \cdot td_{order}(j, o, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (64)$$

$$DV_{pr}(j, o, n) \geq D_{order}^{\min}(o) \cdot td_{order}(j, o, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (65)$$

$$DV_{pr}(j, o, n) \leq Demand(o) \cdot z(j, o, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (66)$$

4.12. Order delivery-time constraints

Continuous variable $td_{order}(j, o, n)$ represents the time used during slot n to deliver order o from tank j , and it must be equal to or less than the length of such time slot, see Eq. (67). Note that Li and Karimi (2011) did not require this variable since they use a term of the form (delivered volume)/(fixed delivery rate) to calculate it.

The start time to deliver order o from product tank j during slot n , i.e. $ts_{order}(j, o, n)$, must be equal to or greater than the start time of such product tank slot (i.e. the end time of the previous slot), see Eq. (68). The end time to deliver order o within slot n must be equal to or less than the end time of that slot, as stated by Eq. (69). Eq. (70) constrains $ts_{order}(j, o, n)$ to be equal to or greater than the beginning of the corresponding time delivery window.

$$td_{order}(j, o, n) \leq T_{pr}(j, n) - T_{pr}(j, n-1) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (67)$$

$$ts_{order}(j, o, n) \geq T_{pr}(j, n-1) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (68)$$

$$ts_{order}(j, o, n) + td_{order}(j, o, n) \leq T_{pr}(j, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (69)$$

$$ts_{order}(j, o, n) \geq TO_{order}^{\text{start}}(o) \cdot z(j, o, n) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (70)$$

Eq. (71) computes the time outside the delivery window required to complete order o (which is used in Eq. (4) to calculate the demurrage cost). All orders must be completed within the scheduling horizon, as stated by Eq. (72).

$$tdem(o) \geq ts_{order}(j, o, n) + td_{order}(j, o, n) - TO_{order}^{\text{end}}(o) - H \cdot (1 - z(j, o, n)) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \text{JON} \quad (71)$$

$$tdem(o) \leq H - TO_{order}^{\text{end}}(o) \quad \forall o \in \mathbf{O} \quad (72)$$

4.13. Inventory balance

The volume in product tank j at the end of slot n , i.e. $V_{pr}(j, n)$, is equal to the volume at the end of slot $n-1$ plus the volume transferred from the blenders minus the volume sent to the shipping/lifting ports, plus the corresponding positive and negative slack variables, see Eq. (73).

One of the assumptions in Section 3 is a piecewise constant component supply profile. Variable $F_{bc}(i, \alpha)$ represents the flow rate of blend component i during supply profile α . Set $\mathbf{ANend} = \{(\alpha, n)\}$ indicates which time slots correspond to each supply profile. The volume in

component tank i at the end of slot n , i.e. $V_{bc}(i, n)$, is equal to the volume at the end of slot $n - 1$ plus the volume received during that interval minus the volume sent to the blenders, plus the corresponding positive and negative slack variables, see Eq. (74).

$$V_{pr}(j, n) = V_{pr}(j, n - 1) + \sum_{bl:(j,bl) \in BJ} V_{trans}(j, bl, n) - \sum_{o:(j,o,n) \in JON} DV_{pr}(j, o, n) + S_{pr}^+(j, n) - S_{pr}^-(j, n) \quad \forall j, n \in \mathbf{N1} \quad (73)$$

$$V_{bc}(i, n) = V_{bc}(i, n - 1) + \sum_{\alpha:(\alpha,n) \in AN} F_{bc}(i, \alpha) \cdot (T_{bc}(i, n) - T_{bc}(i, n - 1)) - \sum_{bl} V_{comp}(i, bl, n) + S_{bc}^+(i, n) - S_{bc}^-(i, n) \quad \forall i, n \in \mathbf{N1} \quad (74)$$

The total volume sent from tank i to all blenders is limited by the maximum pumping rate of such tank or the maximum total production rate (Eq. (75)). Continuous variable $tt_{bc}(i, n)$ indicates the time when a component tank stops sending materials to the blenders. Eq. (76) ensures that such variable is within the corresponding time slot.

$$\sum_{bl} V_{comp}(i, bl, n) \leq \gamma(i) \cdot (tt_{bc}(i, n) - T_{bc}(i, n - 1)) \quad \forall i, n \in \mathbf{N1} \quad (75)$$

$$T_{bc}(i, n - 1) \leq tt_{bc}(i, n) \leq T_{bc}(i, n) \quad \forall i, n \in \mathbf{N1} \quad (76)$$

where $\gamma(i) = \min \left\{ \sum_{bl} F_{comp}^{\max}(i, bl), \sum_{bl} F_{blend}^{\max}(bl) \right\} \forall i$.

Eqs. (77) and (78) force the inventory levels to be within the minimum and maximum limits at the end of the unit slot, and Eqs. (79) and (80) at the moment when the tank stops sending material to the blenders.

$$V_{pr}^{\min}(j) \leq V_{pr}(j, n) \leq V_{pr}^{\max}(j) \quad \forall j, n \in \mathbf{N1} \quad (77)$$

$$V_{bc}^{\min}(i) \leq V_{bc}(i, n) \leq V_{bc}^{\max}(i) \quad \forall i, n \in \mathbf{N1} \quad (78)$$

$$V_{bc}^{\min}(i) \leq V_{bc}(i, n - 1) + \sum_{\alpha:(\alpha,n) \in AN} F_{bc}(i, \alpha) \cdot (tt_{bc}(i, n) - T_{bc}(i, n - 1)) - \sum_{bl} V_{comp}(i, bl, n) \quad \forall i, n \in \mathbf{N1} \quad (79)$$

$$V_{bc}^{\max}(i) \geq V_{bc}(i, n - 1) + \sum_{\alpha:(\alpha,n) \in AN} F_{bc}(i, \alpha) \cdot (tt_{bc}(i, n) - T_{bc}(i, n - 1)) - \sum_{bl} V_{comp}(i, bl, n) \quad \forall i, n \in \mathbf{N1} \quad (80)$$

Eq. (81) constrains the last component tank slots with a given supply profile to coincide with the time when such supply profile ends. This equation assumes that the first supply profile starts at time equal to 0. Eq. (81) is required to avoid using material not available yet, as well as to prevent disregarding material. See Fig. 3 for an example of Eq. (81).

$$T_{bc}(i, n) = FT_{bc}^{end}(\alpha) \quad \forall i, \alpha, n : (\alpha, n) \in \mathbf{ANend} \quad (81)$$

A swing tank can only change its service if it is empty, as required by Eq. (82).

$$V_{pr}(j, n) \leq V_{pr}^{\max}(j) \cdot (1 - ue(j, n)) \quad \forall j, n \in \mathbf{N2} \quad (82)$$

4.14. Slot timings on units

Since the different units (component tanks, product tanks, and blenders) are sharing material, the synchronization of some particular time slots is required in order to guarantee the validity of the material balance equations and a physical feasible solution.

If a component tank is feeding a blender, the start time of the component tank and blender slots must be equal, see Eqs. (83) and (84). If a component tank is feeding a blender (i.e. $y(i, bl, n) = 1$), the times when the blender finishes the blend run and the time when the component tank stops sending material need to be the same, see Eqs. (85) and (86). Note that Eqs. (85) and (86) force the component tanks and blender slots to be equal across all slots composing a blend run, thus synchronizing all the component tank slots so the material arrives to the blender within the same time interval.

$$T_b(bl, n - 1) \geq T_{bc}(i, n - 1) - H \cdot (1 - y(i, bl, n)) \quad \forall i, bl, n \in \mathbf{N1} \quad (83)$$

$$T_b(bl, n - 1) \leq T_{bc}(i, n - 1) + H \cdot (1 - y(i, bl, n)) \quad \forall i, bl, n \in \mathbf{N1} \quad (84)$$

$$T_b(bl, n - 1) + t_{blend}(bl, n) \geq tt_{bc}(i, n - 1) - H \cdot (1 - y(i, bl, n)) \quad \forall i, bl, n \in \mathbf{N1} \quad (85)$$

$$T_b(bl, n - 1) + t_{blend}(bl, n) \leq tt_{bc}(i, n - 1) + H \cdot (1 - y(i, bl, n)) \quad \forall i, bl, n \in \mathbf{N1} \quad (86)$$

When considering that product tanks cannot receive and deliver material at the same time, it is enough to ensure that the start of the product tank slot precedes the start of the blender slot, and that the end of the product tank slot succeeds the end of the blend run, see Eqs. (87) and (88).

$$T_{pr}(j, n+1) \geq T_b(bl, n) + t_{blend}(bl, n+1) - H \cdot (1 - v(j, bl, n+1)) \quad \forall (j, bl) \in \mathbf{BJ}, n \in \mathbf{N2} \quad (87)$$

$$T_{pr}(j, n) \leq T_b(bl, n) + H \cdot (1 - v(j, bl, n+1)) \quad \forall (j, bl) \in \mathbf{BJ}, n \in \mathbf{N2} \quad (88)$$

4.15. Initial conditions

For the first time slot, variables $x(p, bl, n)$, $v(j, bl, n)$, $u(j, p, n)$, $w(bl, n)$, $V_{bc}(i, n)$, $V_{pr}(j, n)$, $ct_{blend}(bl, n)$, $cit_{blend}(bl, n)$, $it_{blend}(bl, n)$, and $CV_{blend}(bl, n)$, are set equal to their initial values (Eqs. (89)–(98)).

$$x(p, bl, n) = x^{ini}(p, bl) \quad \forall (p, bl) \in \mathbf{BP}, n = 0 \quad (89)$$

$$v(j, bl, n) = v^{ini}(j, bl) \quad \forall (j, bl) \in \mathbf{BJ}, n = 0 \quad (90)$$

$$u(j, p, n) = u^{ini}(j, p) \quad \forall (j, p) \in \mathbf{JP}, n = 0 \quad (91)$$

$$w(bl, n) = w^{ini}(bl) \quad \forall bl, n = 0 \quad (92)$$

$$V_{bc}(i, n) = V_{bc}^{ini}(i) \quad \forall i, n = 0 \quad (93)$$

$$V_{pr}(j, n) = V_{pr}^{ini}(j) \quad \forall j, n = 0 \quad (94)$$

$$ct_{blend}(bl, n) = ct_{blend}^{ini}(bl) \quad \forall bl, n = 0 \quad (95)$$

$$cit_{blend}(bl, n) = cit_{blend}^{ini}(bl) \quad \forall bl, n = 0 \quad (96)$$

$$it_{blend}(bl, n) = it_{blend}^{ini}(bl) \quad \forall bl, n = 0 \quad (97)$$

$$CV_{blend}(bl, n) = CV_{blend}^{ini}(bl) \quad \forall bl, n = 0 \quad (98)$$

4.16. Simultaneous receipt/delivery by product tanks

If the product tanks are allowed to receive and deliver material at the same time, Eq. (25) is omitted from the model and Eqs. (99)–(107) are added. 0–1 continuous variable $ze(j, o, n)$ is equal to 1 if tank j starts or stops delivering material to order o at the end of product tank slot n , see Eqs. (99)–(101).

$$ze(j, o, n) \geq z(j, o, n) - z(j, o, n+1) \quad \forall n \in \mathbf{N2}, (j, o) : (j, o, n) \in \mathbf{JON} \quad (99)$$

$$ze(j, o, n) \geq z(j, o, n+1) - z(j, o, n) \quad \forall n \in \mathbf{N2}, (j, o) : (j, o, n) \in \mathbf{JON} \quad (100)$$

$$ze(j, o, n) \leq 2 - z(j, o, n) - z(j, o, n+1) \quad \forall n \in \mathbf{N2}, (j, o) : (j, o, n) \in \mathbf{JON} \quad (101)$$

$$ze(j, o, n) \leq z(j, o, n) + z(j, o, n+1) \quad \forall n \in \mathbf{N2}, (j, o) : (j, o, n) \in \mathbf{JON} \quad (102)$$

Eqs. (67), (68), (103) and (104) force the start and end times of a product tank slot to be equal to the delivery times used in each slot.

$$ts_{order}(j, o, n) + td_{order}(j, o, n) \geq T_{pr}(j, n) - H \cdot (1 - z(j, o, n) + ze(j, o, n)) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON} \quad (103)$$

$$ts_{order}(j, o, n+1) \leq T_{pr}(j, n) + H \cdot (1 - z(j, o, n+1) + ze(j, o, n)) \quad \forall n \in \mathbf{N2}, (j, o) : (j, o, n) \in \mathbf{JON} \quad (104)$$

Eqs. (105)–(108) match the start and end times of a blender slot with those of a delivery run from the product tank receiving material from such blender.

$$ts_{order}(j, o, n) \geq T_b(bl, n-1) - H \cdot (2 - v(j, bl, n) - z(j, o, n)) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON}, bl : (j, bl) \in \mathbf{BJ} \quad (105)$$

$$ts_{order}(j, o, n) \leq T_b(bl, n-1) + H \cdot (2 - v(j, bl, n) - z(j, o, n)) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON}, bl : (j, bl) \in \mathbf{BJ} \quad (106)$$

$$ts_{order}(j, o, n) + td_{order}(j, o, n) \geq \quad (107)$$

$$T_b(bl, n-1) + t_{blend}(bl, n) - H \cdot (2 - v(j, bl, n) - z(j, o, n)) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON}, bl : (j, bl) \in \mathbf{BJ}$$

$$ts_{order}(j, o, n) + td_{order}(j, o, n) \leq \quad (108)$$

$$T_b(bl, n-1) + t_{blend}(bl, n) + H \cdot (2 - v(j, bl, n) - z(j, o, n)) \quad \forall n \in \mathbf{N1}, (j, o) : (j, o, n) \in \mathbf{JON}, bl : (j, bl) \in \mathbf{BJ}$$

4.17. Lower bounds for the materials and switching costs

An estimation of lower bounds can be determined for the materials and switching costs. A lower bound for the materials cost can be computed by formulating and solving an aggregate optimization model minimizing the materials cost subject to the components availability, the maximum blending capacity and inventory constraints, using a single period for the entire horizon or a multi-period model using the inventory pinch concept to delineate the time period boundaries (e.g. Castillo et al., 2013; Castillo and Mahalec, 2014a, 2014b). The volumes of components from the aggregate solution are used in Eq. (109).

$$\text{Blend cost} \geq \sum_i c_1(i) \cdot V_{comp,agg}(i) \quad (109)$$

A lower bound for the switching cost can be estimated by Eq. (110). sw_{exp} is the expected minimum number of blend runs, which is determined as follows: (i) for a given product, if the result of (initial inventory) – (total demand during the horizon) – (minimum inventory limit) is negative, that product must be blended during the scheduling horizon at least once, (ii) after determining which different products are required, subtract those that are being blended at the beginning of the horizon. ue_{exp} is the expected minimum number of product changeovers in the swing tanks, which can be estimated as the number of different products required to be blended that are not initially stored by any product tank. Finally, at least one delivery run is expected for each order. The notation $\|\mathbf{set}\|$ indicates the number of elements in the set.

$$\text{Switching cost} \geq \left(\min_{bl} c_2(bl) \right) \cdot sw_{exp} + \left(\min_j c_3(j) \right) \cdot ue_{exp} + c_4 \cdot \|\mathbf{O}\| \quad (110)$$

4.18. Nonlinear blending models

Eqs. (111)–(113) correspond to nonlinear blending models and quality specifications. Inclusion of Eq. (111) will result in a MINLP scheduling model. The nonlinear equations used in this work are presented in Section 4.2.

$$Q_{pr}(bl, e, n) = f(V_{comp}(i, bl, n), V_{blend}(bl, n), Q_{bc}(i, e, \theta)) \quad \forall bl, e \in \mathbf{ENL}, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111)$$

$$Q_{pr}(bl, e, n) \geq Q_{pr}^{\min}(p, e) \cdot x(p, bl, n) \quad \forall e \in \mathbf{ENL}, (p, bl) \in \mathbf{NP}, n \in \mathbf{N1} \quad (112)$$

$$Q_{pr}(bl, e, n) \leq Q_{pr}^{\max}(p, e) + \xi^{\max}(e) \cdot (1 - x(p, bl, n)) \quad \forall e \in \mathbf{ENL}, (p, bl) \in \mathbf{BP}, n \in \mathbf{N1} \quad (113)$$

4.19. Scheduling adjustment (reducing blending rate variations)

The presented continuous-time scheduling model does not assume fixed blending rates (although this can be easily incorporated into the model if desired by setting $F_{blend}^{\min}(bl) = F_{blend}^{\max}(bl)$) and it does not compute them directly (in order to avoid nonconvex nonlinear terms). Therefore, if a blend run spans several time slots, it is possible that the blending rates vary across the slots. In order to reduce such variations, average blending rates are computed based on the solution from the full-space model and, with the production and delivery sequence fixed, the full-space model is resolved minimizing the difference between the average and actual blending rates.

4.20. Versions of the full-space model

The following notation is used. “L” stands for linear blending rules, “N” for nonlinear blending equations, “SimRD” means that simultaneous receipt/delivery by product tanks is permitted, and “NoSimRD” indicates that simultaneous receipt/delivery by product tanks is not allowed. Different operational scenarios can be constructed as follows:

- (i) No simultaneous receipt/delivery by product tanks, based on linear models, model L-NoSimRD is defined by Eqs. (1)–(98), (109) and (110).
- (ii) Simultaneous receipt/delivery by product tanks, based on linear models, model L-SimRD is described by Eqs. (1)–(24), (26)–(110).
- (iii) No simultaneous receipt/delivery by product tanks, based on non-linear models, model N-NoSimRD is defined by Eqs. (1)–(98), (109)–(113).
- (iv) Simultaneous receipt/delivery by product tanks, based on non-linear models, model N-SimRD is defined by Eq. (1)–(24), (26)–(113).

4.21. Nonlinear blending equations

In this work, we only consider the research octane number (RON) and the motor octane number (MON) as the properties to blend nonlinearly following the ethyl RT-70 models (Singh et al., 2000; Healey et al., 1959). Therefore, set $\mathbf{ENL} = \{\text{‘RON’}, \text{‘MON’}\}$. In the ethyl RT-70 models, RON and MON are functions of the blend components sensitivity (i.e. $sens(i) = Q_{bc}(i, \text{‘RON’}) - Q_{bc}(i, \text{‘MON’})$), and olefins (‘OLP’) and aromatics (‘ARO’) content. Eq. (111) is substituted by Eqs. (111-a)–(111-f). The model parameter values used by Singh et al. (2000) are: $a_1 = 0.03224$, $a_2 = 0.00101$, $a_3 = 0$, $a_4 = 0.04450$, $a_5 = 0.00081$, and $a_6 = -0.0645$.

$$V_{comp}(i, bl, n) = r(i, bl, n) \cdot V_{blend}(bl, n) \quad \forall i, bl, n \in \mathbf{N1} \quad (111-a)$$

$$r_{avg}^{RON}(bl, n) = \sum_i r(i, bl, n) \cdot Q_{bc}(i, e, \theta) \quad \forall e = \text{‘RON’}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-b)$$

$$r_{avg}^{MON}(bl, n) = \sum_i r(i, bl, n) \cdot Q_{bc}(i, e, \theta) \quad \forall e = \text{‘MON’}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-c)$$

$$sens_{avg}(bl, n) = \sum_i r(i, bl, n) \cdot sens(i) \quad \forall i, bl, n \in \mathbf{N1} \quad (111-d)$$

$$sens_{avg}^{RON}(bl, n) = \sum_i r(i, bl, n) \cdot Q_{bc}(i, e, \theta) \cdot sens(i) \quad \forall e = \text{‘RON’}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-e)$$

$$sens_{avg}^{MON}(bl, n) = \sum_i r(i, bl, n) \cdot Q_{bc}(i, e, \theta) \cdot sens(i) \quad \forall e = \text{‘MON’}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-f)$$

Table 1
Summary of the examples from Test sets #1 and #2 (Li and Karimi, 2011).

Example ID	# Blenders	# Orders	# Products	# Product tanks	# Quality properties (weight basis)
3	1	12	4	11	5
4	1	15	4	11	9 (2)
7	1	20	4	11	9 (2)
8	2	20	4	11	9 (2)
9	2	23	5	11	9 (2)
12	2	35	5	11	9 (2)
14	3	45	5	11	9 (2)

$$Ol_{avg}(bl, n) = \sum_i r(i, bl, n) \cdot Q_{bc}(i, e, \theta) \quad \forall e = \text{'OLF'}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-g)$$

$$Ar_{avg}(bl, n) = \sum_i r(i, bl, n) \cdot Q_{bc}(i, e, \theta) \quad \forall e = \text{'ARO'}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-h)$$

$$Ol_{avg}^{sq}(bl, n) = \sum_i r(i, bl, n) \cdot [Q_{bc}(i, e, \theta)^2] \quad \forall e = \text{'OLF'}, bl, n \in \mathbf{N1}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-i)$$

$$Ar_{avg}^{sq}(bl, n) = \sum_i r(i, bl, n) \cdot [Q_{bc}(i, e, \theta)^2] \quad \forall e = \text{'ARO'}, bl, n \in \mathbf{N}, \theta : (\theta, n) \in \mathbf{QN} \quad (111-j)$$

$$Q_{pr}(bl, e, n) = r_{avg}^{RON}(bl, n) + a_1 (sens_{avg}^{RON}(bl, n) - r_{avg}^{RON}(bl, n) \cdot sens_{avg}(bl, n)) \\ + a_2 (Ol_{avg}^{sq}(bl, n) - [Ol_{avg}(bl, n)]^2) \\ + a_3 ([Ar_{avg}^{sq}(bl, n)]^2 - 2 [Ar_{avg}^{sq}(bl, n)] [Ar_{avg}(bl, n)]^2 + [Ar_{avg}(bl, n)]^4) \quad (111-k)$$

$\forall e = \text{'RON'}, bl, n \in \mathbf{N1}$

$$Q_{pr}(bl, e, n) = r_{avg}^{MON}(bl, n) + a_4 (sens_{avg}^{MON}(bl, n) - r_{avg}^{MON}(bl, n) \cdot sens_{avg}(bl, n)) \\ + a_5 (Ol_{avg}^{sq}(bl, n) - [Ol_{avg}(bl, n)]^2) \\ + \left(\frac{a_6}{10000}\right) ([Ar_{avg}^{sq}(bl, n)]^2 - 2 [Ar_{avg}^{sq}(bl, n)] [Ar_{avg}(bl, n)]^2 + [Ar_{avg}(bl, n)]^4) \quad (111-l)$$

$\forall e = \text{'MON'}, bl, n \in \mathbf{N1}$

5. Test problems

Two sets of test problems have been used in this study:

Test set #1 consists of a subset of problems from Li and Karimi (2011): Example number 3, 4, 7, 8, 9, 12, and 14. Scheduling horizon is 8 days, linear blending constraints are employed, and component tanks have different supply flowrates along the horizon. Quality properties

Table 2
Periods, time slots, and orders that can be delivered in each period (Test sets #1 and #2).

Ex	Period	Duration (h)	Slots	Orders that can be delivered
3, 4	1	100	1, 2	01–07, 012–015
	2	92	3, 4	08–011
7 ^a	1	80	1–3	01–07, 012–019
	2	70	4–6	08
	3	42	7	08–011, 020
8	1	80	1, 2	01–07, 012–019
	2	70	3, 4	08
	3	42	5, 6	08–011, 020
9	1	80	1, 2	01–07, 012–019
	2	70	3, 4	08, 021
	3	42	5, 6	08–011, 020, 022, 023
12	1	50	1–3	01–07, 012, 013, 015, 019, 033
	2	50	4–6	014–018, 027, 028, 033
	3	50	7–9	08, 021, 024, 029–032, 034, 035
	4	42	10–12	08–011, 020, 022, 023, 025, 026
14	1	50	1–3	01–07, 012, 013, 015, 019, 026
	2	50	4–7	014–018, 026
	3	50	8–10	08, 021, 024, 027–031, 045
	4	42	11–13	08–011, 020, 022, 023, 025, 032–044

^a For the NoSimRD scenario, example 7 requires 9 slots as follows: (period: slots) = {(1: 1–4), (2: 5–7), (3: 8–9)}.

Table 3
RON and MON values for examples from Test set #2.

Property	Blend Components								
	C1	C2	C3	C4	C5	C6	C7	C8	C9
RON	75	90.3	95.6	97.3	83	100	115	118	81
MON	66	80.8	80.5	91.7	74	100	109	100	72
	Product specifications [min, max]								
	P1		P2		P3		P4		P5
RON	[95,200]		[96,200]		[94,200]		[90,200]		[98,200]
MON	[0,200]		[0,200]		[0,200]		[0,200]		[0,200]

Table 4
Continuous-time full-space model size comparison for Test set #1.

Ex	Li and Karimi (2011) NoSimRD				Model L-NoSimRD-opt				Model L-SimRD-opt			
	# Slots	# Eqs	# Conts	# Bins	# Slots	# Eqs	# Conts	# Bins	# Slots	# Eqs	# Conts	# Bins
3	3	2987	799	246	4	3910	1251	306	4	5134	1432	306
4	3	3479	925	285	4	4512	1404	342	4	6060	1639	342
7	7	10,337	2381	986	9	10,515	3298	753	7	10,660	2940	571
8	6	11,491	2244	949	6	8881	2446	553	6	11,737	2769	553
9	6	12,635	2391	1015	6	9780	2569	594	6	12,876	2915	594
12	12	34,151	6572	3050	12	22,203	6205	1317	12	30,051	7008	1317
14	13	49,097	8508	3989	13	31,135	8012	1628	13	43,165	8945	1628

#Slots = number of unit slots, #Eqs = equations, #Conts = continuous variables, and #Bins = binary variables.

under specification are RON, Reid vapor pressure, sulfur content, specific gravity, aromatics content, olefin content, benzene content, oxygenates, and flammability limit. According to Li and Karimi (2011), they use the same addition bases and index correlations as Li et al. (2010); therefore, only sulfur content and oxygenates blend on a weight basis. However, Li and Karimi (2011) do not consider specific gravity in example 3, hence in this work sulfur content is assumed to blend linearly on a volumetric basis in example 3. Table 1 shows a description of the blending system for each example. Table 2 shows the allocation of time slots to each time period (in this work, time periods correspond to intervals with different supply flowrates of blend components) and the orders that can be delivered during such periods.

Test set #2 consists of example number 4, 8, 12, and 14 from Li and Karimi (2011); however, RON and MON properties are considered to blend nonlinearly following the ethyl RT-70 models. RON index correlation from Li et al. (2010) was used to compute the actual RON values and product specifications. Li and Karimi (2011) do not specify MON values, and these were assumed in this work. For simplicity, MON minimum product specifications were set equal to 0 in order to observe only the effect of the RON constraint in the optimum. RON and MON values and specifications are shown in Table 3.

6. Computational performance

The continuous-time blend scheduling models L-SimRD, L-NoSimRD, N-SimRD, and N-NoSimRD were implemented in GAMS IDE 24.3.2. CPLEX 12.6 was used for MILP models, and BARON 14.0.3 and ANTIGONE 1.1 were employed for MINLP models. CPLEX 12.6 and ANTIGONE 1.1 were selected for LP and NLP models, respectively, required to compute the lower bounds on the blend cost (Castillo and Mahalec, 2014a). All problems have been solved on DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40 GHz, and 12 GB RAM) running Windows Server 2008 R2 OS.

Table 5
Results for full-space model L-SimRD, not using Eqs. (109) and (110) lower bounds, not penalizing deliveries ($c_4 = 0$). Test set #1.

Ex	Obj. func. as defined by:		Gap (%)	RMIP ($\times 10^3$ \$)	CPU time (s) to reach:			Blend cost ($\times 10^3$ \$)	# TLK	# BR	# TST	# DR
	Eq. (LK-40)	Eq. (1) ($\times 10^3$ \$)			Stopping criteria	Final upper bound	Opt. gap <5%					
3	3159.1	3179.1	0	3085.1	53.8	1.0	0.5	3139.1	1	2	0	19
4	4556.7	4576.7	0	4481.8	106.5	13.1	11.0	4536.7	1	2	0	20
7	8100.3	8120.3	3.87	7737.0	10,800 ^a	364.0	362	8040.3	3	4	0	29
8	8080.3	8120.3	4.23	7737.0	10,800 ^a	292.0	128	8040.3	2	4	0	27
9	10,778.8	10,818.8	7.62	9962.2	10,800 ^a	442.0	NA	10,704.3	4	5	1	31
12	15,241.7	15,281.7	9.79	13,773.5	43,200 ^a	42,743	NA	15,147.2	5	6	1	46
14	21,125.9	21,185.9	9.86	19,088.5	43,200 ^a	41,720	NA	21,046.9	4	6	1	52

^a Reached maximum allocated time.

#TLK = number of transitions counted the same way as Li and Karimi (2011) model, #BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs, NA = Not available.

Table 6
Results for full-space model L-SimRD, using Eqs. (109) and (110) lower bounds, not penalizing deliveries ($c_4 = 0$). Test set #1.

Ex	Obj. func. as defined by:		Gap (%)	RMIP ($\times 10^3$ \$)	CPU time (s) to reach:			Blend Cost ($\times 10^3$ \$)	# TLK	# BR	# TST	# DR
	Eq. (LK-40) ($\times 10^3$ \$)	Eq. (1) ($\times 10^3$ \$)			Stopping criteria	Final upper bound	Opt. gap <5%					
3	3159.1	3179.1	0	3179.1	2.56	0.81	0.81	3139.1	1	2	0	17
4	4556.7	4576.7	0	4576.7	2.27	0.80	0.80	4536.7	1	2	0	20
7	8100.3	8120.3	0	8120.3	63.1	61.1	60.1	8040.3	3	4	0	31
8	8080.3	8120.3	0	8120.3	37.8	35.5	9.72	8040.3	2	4	0	28
9	10,778.8	10,818.8	0	10,818.8	83.4	80.7	40.1	10,704.3	4	5	1	32
12	15,241.7	15,281.7	0.13	15,261.7	43,200 ^a	189	189	15,147.2	5	6	1	49
14	21,101.4	21,161.4	0	21,161.4	12,106	12,106	497	21,046.9	3	5	1	62

^a Reached maximum allocated time.
#TLK = number of transitions counted the same way as Li and Karimi (2011) model, #BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

Table 7
Results for full-space model L-SimRD, using Eqs. (109) and (110) lower bounds, penalizing deliveries ($c_4 = 5$). Test set #1.

Ex	Obj. func. as defined by:		Gap (%)	RMIP ($\times 10^3$ \$)	CPU time (s) to reach:			Blend Cost ($\times 10^3$ \$)	# TLK	# BR	# TST	# DR
	Eq. (LK-40) ($\times 10^3$ \$)	Eq. (1) ($\times 10^3$ \$)			Stopping criteria	Final upper bound	Opt. gap <5%					
3	3159.1	3254.1	0	3239.1	44.4	8.8	0.8	3139.1	1	2	0	15
4	4556.7	4666.7	0	4651.7	530	3.8	1.7	4536.7	1	2	0	18
7	8102.4	8247.4	0.25	8220.3	10,800 ^a	63.3	22.7	8042.4	3	4	0	25
8	8080.6	8240.6	0.25	8220.3	10,800 ^a	660	9.9	8040.6	2	4	0	24
9	10,781.1	10,951.1	0.16	10,933.8	10,800 ^a	896	46.7	10,706.6	4	5	1	26
12	15,221.7	15,451.7	0.10	15,436.7	43,200 ^a	3320	589	15,147.2	4	5	1	38
14	21,101.4	21,386.4	0	21,386.4	40,192	40,192	187	21,046.9	3	5	1	45

^a Reached maximum allocated time.
#TLK = number of transitions counted the same way as Li and Karimi (2011) model, #BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

6.1. Test set #1

Table 4 shows the full-space model size for these examples, while Tables 5 and 6 show the results for scenarios when multiple delivery runs from the tanks are not minimized (penalized). Hence, these results are for the same kind of problems as those solved by Li and Karimi (2011). In order to facilitate comparison with Li and Karimi (2011), the results in Table 5 and Table 6 show the objective function computed by Eq. (1) and its equivalent value in terms of Eq. (LK-40) from Li and Karimi (2011). Note that Eq. (LK-40) does not take into account the transitions that occur during the first time slot (i.e. slot $n = 0$).

For smaller examples, the computed results are the same as those by Li and Karimi (2011). There is a discrepancy between the problem data as published and the solutions reported by Li and Karimi (2011) for larger examples. Our recent communication with Dr. Li (Li, 2014) indicates that the published problem description data for examples 9 and 14 have typographical errors. Hence, the solutions presented here for examples 9 and 14 are for data as published and the comparison with the published solution by Li and Karimi (2011) is not meaningful. The example 12 description data as published are correct (Li, 2014). Since the solution of Example 12 as published by Li and Karimi (2011) is lower than the lower bound for the blend cost (i.e. it is infeasible), it is not meaningful to compare our solution of example 12 with the originally published solution.

In order to evaluate the impact of additional lower bounds, i.e. Eqs. (109) and (110), the full-space continuous-time model, without penalizing multiple deliveries from tanks, has been solved without these bounds (see Table 5) and with these bounds (see Table 6). Without additional lower bounds, the optimality gap cannot be closed for larger problems even after 12 h. With additional lower bounds, examples 3–9 are solved very rapidly, example 14 is solved to optimality within approximately 3.5 h, while example 12 is solved to 0.13% gap within 12 h. The solution of the relaxed problem (i.e. RMIP) is reported as well. Eqs. (109) and (110) increase the RMIP solutions from 2 to almost 10%, depending on the example.

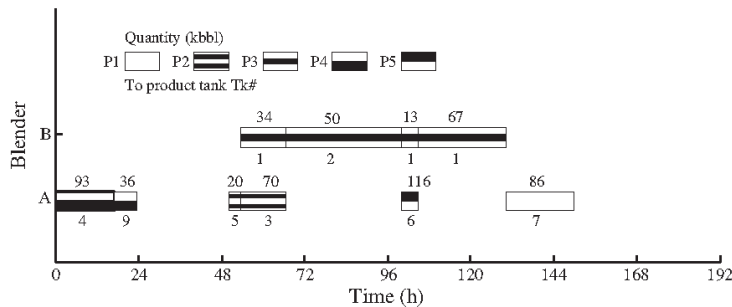


Fig. 4. Production schedule for example 12, Test set #1, SimRD scenario, penalizing delivery runs.

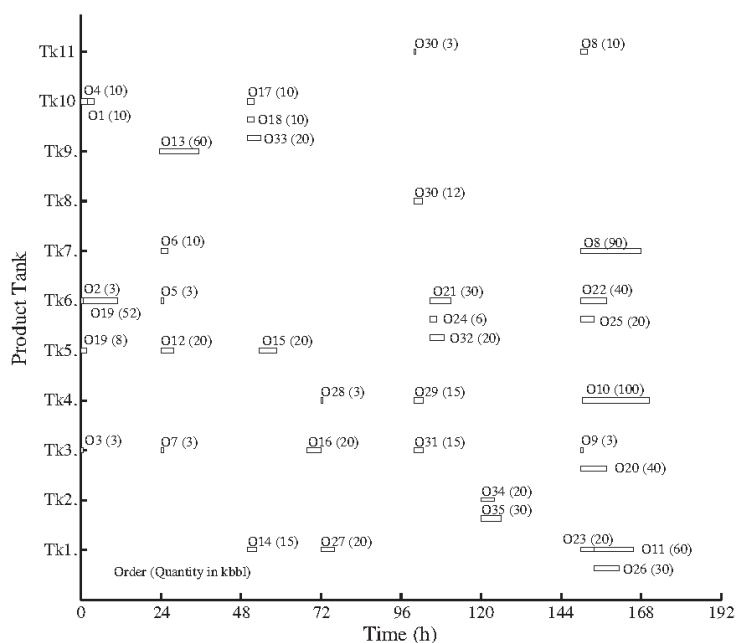


Fig. 5. Delivery schedule for example 12, Test set #1, SimRD scenario, penalizing delivery runs.

Table 8

Results for full-space model L-NoSimRD, using Eqs. (109) and (110) lower bounds, not penalizing deliveries ($c_4 = 0$). Test set #1.

Ex	Obj. func. as defined by:		Gap (%)	RMIP ($\times 10^3$ \$)	CPU time (s) to reach:			Blend Cost ($\times 10^3$ \$)	# TLK	# BR	# TST	# DR
	Eq. (LK-40) ($\times 10^3$ \$)	Eq. (1) ($\times 10^3$ \$)			Stopping criteria	Final upper bound	Opt. gap < 5%					
3	3159.1	3179.1	0	3179.1	1.90	0.36	0.36	3139.1	1	2	0	17
4	4556.7	4576.7	0	4576.7	2.02	0.33	0.33	4536.7	1	2	0	20
7 ^a	8100.3	8120.3	0	8120.3	36.9	32.8	32.8	8040.3	3	4	0	29
8	8080.3	8120.3	0	8120.3	45.8	43.9	13.7	8040.3	2	4	0	28
9	10,778.8	10,818.8	0	10,818.8	210	210	141	10,704.3	4	5	1	30
12	15,221.7	15,261.7	0	15,261.7	1342	1342	47.7	15,147.2	4	5	1	45
14	21,101.4	21,161.4	0	21,161.4	2877	2877	202	21,046.9	3	5	1	57

^a Reached maximum allocated time.

#TLK = number of transitions counted the same way as Li and Karimi (2011) model, #BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

Solutions in Tables 5 and 6 have large numbers of delivery runs associated with the product tanks. Since it is desirable to have a delivery schedule as simple as possible, the number of delivery runs should be minimized. The results for such scenario are included in Table 7. The new version of the continuous-time full-space model finds solutions which are very close to the optimum or are optimal, but the execution times are too large for practical applications. Figs. 4 and 5 show the production and delivery schedule, respectively, computed for example 12 (penalizing delivery runs, simultaneous delivery and receipt by product tanks).

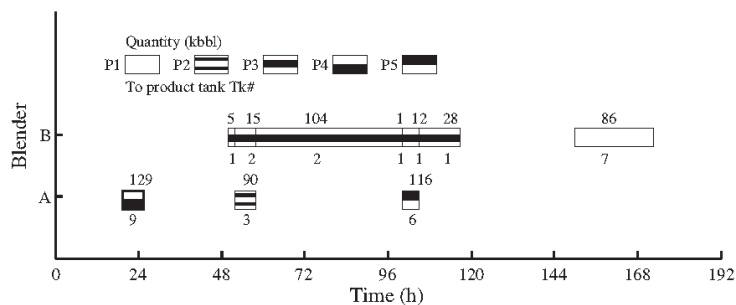


Fig. 6. Production schedule for example 12, Test set #1, NoSimRD scenario, not penalizing delivery runs.

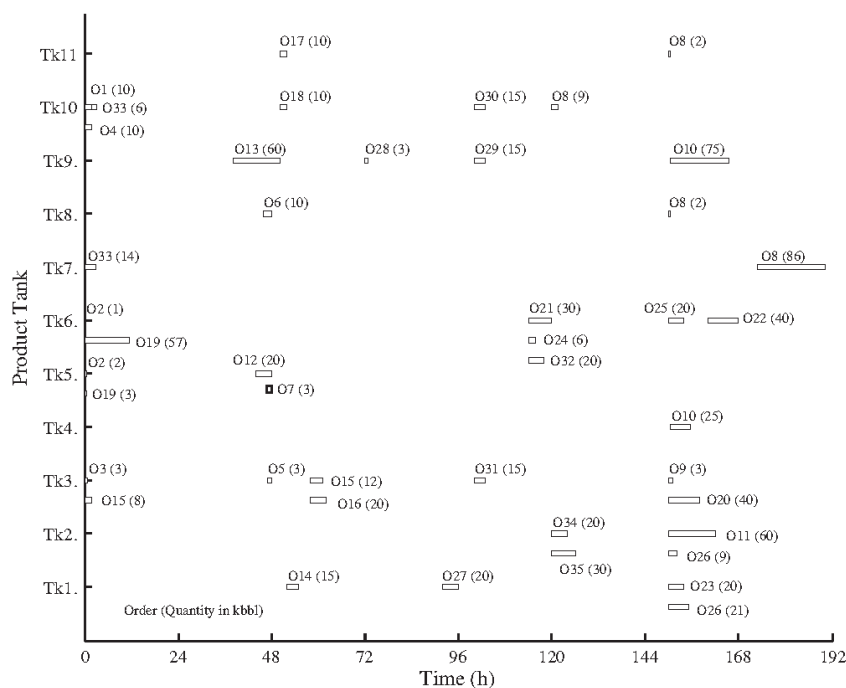


Fig. 7. Delivery schedule for example 12, Test set #1, NoSimRD scenario, not penalizing delivery runs.

Table 9
Model N-NoSimRD-opt size for Test set #2.

Ex	Model N-NoSimRD-opt				
	# Slots	# Eqs	# Confs	# Bins	# Nonlinear terms
4	4	4624	1484	342	104
8	6	9217	2686	553	312
12	12	22,923	6685	1317	624
14	13	32,305	8792	1628	1014

#Slots = number of unit slots, #Eqs = equations, #Confs = continuous variables, and #Bins = binary variables.

Operational scenario when simultaneous receipt and delivery is not permitted from any tank (model L-NoSimRD) limits the times when a blender can run. Table 8 shows the results for this operational scenario. Solutions are very similar to the SimRD scenario. Solution times required by model L-NoSimRD are similar to those from model L-SimRD for example 3, 4, 7 and 8, larger for example 9, and shorter for example 12 and 14. Model L-NoSimRD solves example 12 and 14 to optimality more easily than model L-SimRD. The reason for this can be attributed to the reduced number of feasible schedules compared with the SimRD scenario. Figs. 6 and 7 show the production and delivery schedule, respectively, computed for example 12 (not penalizing delivery runs, no simultaneous delivery and receipt by product tanks).

Table 10
Results for full-space model N-NoSimRD, using Eqs. (109) and (110) lower bounds, not penalizing deliveries ($c_4 = 0$). Test set #2.

Ex	MINLP solver	Obj. func.		RMIP ($\times 10^3$ \$)	CPU time (s) to reach:			Blend Cost ($\times 10^3$ \$)	# BR	# TST	# DR
		Value ($\times 10^3$ \$)	Gap (%)		Stopping criteria						
					Final upper bound	Opt. gap < 5%					
4	ANTIGONE	4633.0	0.01	4632.7	856	856	141	4593.0	2	0	21
	BARON	4633.0	0.01	4632.7	15	15	11	4593.0	2	0	21
8	ANTIGONE	8205.4	0.03	8203.1	10,800 ^a	9246	380	8125.4	4	0	28
	BARON	8207.4	0.05	8203.1	10,800 ^a	1388	79	8127.4	4	0	26
12	ANTIGONE	15,406.8	0.16	15,382.6	43,200 ^a	43,054	907	15,272.3	6	1	42
	BARON	15,453.1	0.45	15,382.6	43,200 ^a	7933	7933	15,318.6	6	1	45
14	ANTIGONE	21,283.1	0.19	21,243.1	43,200 ^a	12,839	1205	21,128.6	7	1	56
	BARON	21,497.7	1.20	21,243.1	43,200 ^a	7204	3634	21,383.2	5	1	56

^a Reached maximum allocated time.
#BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

6.2. Test set #2

Nonlinear problems have been solved for the scenario when no simultaneous receipt and delivery by product tanks is allowed, and when delivery runs are not penalized. Eqs. (109) and (110) were included in the model. The size of model N-NoSimRD-opt for Test set #2 examples is shown in Table 9. As expected, the number of binary variables remains the same, and the number of equations and continuous variables increased slightly due to the addition of Eqs. (111-a)–(111-l), (112), (113). The number of nonlinear terms is presented as well. Table 10 shows the results obtained by global MINLP solvers BARON and ANTIGONE. Only small-scale example 4 is solved to optimality. Example 8 is solved very close to optimality within the allocated time of 3 h. Large-scale examples 12 and 14 cannot be solved to proven optimality in 12 h. However, final optimality gaps are relatively small, and ANTIGONE computes solutions with gap values below 5% in less than 30 min. ANTIGONE seems to find better solutions than BARON for the same execution time.

7. Conclusions

In this paper we have presented a new version of the continuous-time blend scheduling model, which is a significantly modified version of the previously published model by Li and Karimi (2011). This new version improves convergence by addition of novel lower bounds and by reducing the number of binary variables via (a) equations which convert some binary variables into 0–1 continuous, and by (b) using demand information in order to avoid generating binary variables where orders cannot be delivered. The model also includes additional operational features such as penalty for deliveries of the same order from different product tanks, product and blender-dependent minimum setup times, maximum delivery rate from component tanks, and a threshold volume for each blend.

These modifications significantly improve convergence of the model, to the extent that previously published unsolved large linear blend scheduling examples have been solved to optimality or very close to it. The execution times are two to three orders of magnitude shorter than originally published times. Satisfactory convergence for medium size nonlinear blend scheduling problems (ethyl RT-70 equations for octane blending) has been obtained applying global MINLP solvers and the results are very close to optimality. Solutions with optimality gap smaller than 5% were found in less than 30 min for the two large-scale problems considered in this work. This has motivated our work on a new type of scheduling algorithm which is described in the companion paper (Castillo-Castillo and Mahalec, 2015).

Acknowledgments

Support by Ontario Research Foundation and McMaster Advanced Control Consortium is gratefully acknowledged.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at <http://dx.doi.org/10.1016/j.compchemeng.2015.08.003>.

References

- Castillo PA, Kelly JD, Mahalec V. Inventory pinch algorithm for gasoline blend planning. *AIChE J* 2013;59(10):3748–66.
- Castillo PA, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling-part I: gasoline blend planning. *AIChE J* 2014a;60(6):2158–78.
- Castillo PA, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling-part II: gasoline blend scheduling. *AIChE J* 2014b;60(7):2475–97.
- Castillo-Castillo PA, Mahalec V. Inventory pinch gasoline blend scheduling algorithm combining discrete- and continuous-time models. *Comput Chem Eng* 2016;84:611–26.
- Floudas CA, Lin X. Continuous-time vs discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng* 2004;28:2109–29.
- Harjunkoski I, Maravelias CT, Bongers P, Castro PM, Engell S, Grossmann IE, et al. Scope for industrial applications of production scheduling models and solution methods. *Comput Chem Eng* 2014;62:161–93.
- Healey WC, Maassen CW, Peterson RT. A new approach to blending octanes. In: Proc. 24th Meeting of American Petroleum Institute's Division of Refining. New York; 1959.
- Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Ind Eng Chem Res* 2003;42:825–35.
- Joly M, Pinto JM. Mixed-integer programming techniques for the scheduling problem of fuel oil and asphalt production. *Inst Chem Eng* 2003;81:427–47.
- Kołodziej SP, Grossmann IE, Furman KC, Sawayac NW. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. *Comput Chem Eng* 2013;53:122–42.
- Li J. Personal communication. 2014.
- Li J, Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res* 2011;50:9156–74.
- Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. *AIChE J* 2010;56:441–65.
- Maravelias CT. General framework and modeling approach classification for chemical production scheduling. *AIChE J* 2012;58:1812–28.
- Mendez CA, Grossmann IE, Harjunkoski I, Kabore P. A simultaneous optimization approach for off-line blending. *Comput Chem Eng* 2006;30:614–34.
- Neiro SMS, Murata VV, Pinto JM. Hybrid time formulation for diesel blending and distribution scheduling. *Ind Eng Chem Res* 2014;53:17124–34.
- Singh A, Forbes JF, Vermeer PJ, Woo SS. Model-based real-time optimization of automotive gasoline blending operations. *J Process Control* 2000;10:43–58.
- Sundaramoorthy A, Maravelias CT. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind Eng Chem Res* 2011;50:5023–40.
- Velez S, Maravelias CT. Theoretical framework for formulating MIP scheduling models with multiple and non-uniform discrete-time grids. *Comput Chem Eng* 2015;72:233–54.

Chapter 6: Inventory Pinch Gasoline Blend Scheduling Algorithm Combining Discrete- and Continuous-Time Models

This chapter has been published in the Computers and Chemical Engineering Journal.
Complete citation:

Castillo Castillo, P. A., & Mahalec, V. (2016). Inventory pinch gasoline blend scheduling algorithm combining discrete- and continuous-time models. *Computers & Chemical Engineering*, 84, 611–626. Elsevier Ltd., doi: 10.1016/j.compchemeng.2015.08.005

Permission from © Elsevier Ltd. All rights reserved.

In Chapter 6, the MPIP-C algorithm is introduced. It combines the solution strategy described in Chapter 3 with the continuous-time blend scheduling model from Chapter 5. The continuous-time blend scheduling model enables MPIP-C algorithm to solve the 3rd level for the entire time horizon (instead of subintervals as in Chapter 3). The execution times required by MPIP-C are almost one order of magnitude shorter than those required by MPIP algorithm. It is demonstrated as well that MPIP-C can handle nonlinear blending rules.

Chapter 6 marks a milestone within my Ph.D. project. The MPIP-C method is a heuristic approach that provides optimal or near-optimal solutions in a few seconds for linear and nonlinear blend scheduling problems, and with a reduced number of blend recipes. This fulfills one of the general objectives of this work.



Inventory pinch gasoline blend scheduling algorithm combining discrete- and continuous-time models

Pedro A. Castillo-Castillo, Vladimir Mahalec*

Department of Chemical Engineering, McMaster University, 1280 Main St. West, Hamilton, ON L8S 4L8, Canada

ARTICLE INFO

Article history:

Received 25 August 2014

Received in revised form 30 July 2015

Accepted 4 August 2015

Available online 18 August 2015

Keywords:

Inventory pinch

Nonlinear gasoline blending

Discrete-time models

Continuous-time models

ABSTRACT

This work introduces multi-period inventory pinch-based algorithm to solve continuous-time scheduling models (MPIP-C algorithm), a three level method which combines discrete-time approximate scheduling with continuous-time detailed scheduling and with inventory pinch-based optimization of operating states. When applied to gasoline blending, the top level computes optimal recipes for aggregated blends over periods initially delineated by inventory pinch points. Discrete-time middle level uses fixed blend recipes to compute an approximate schedule, i.e. what, when, and how much to produce; it also allocates swing storage and associated product shipments with specific storage. Continuous-time model at the third level computes when exactly to start/stop an operation (blend, tank transfer, shipment). MPIP-C algorithm solves linear or nonlinear problems 2–3 orders of magnitude faster than full-space models.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

Process industries' supply chains are comprised of facilities and activities which procure raw materials to the plants, and store and distribute finished products to the customers. Structure of the supply chains (location, capacity, and type of each plant and storage) is decided upon by optimizing returns over long time horizons, e.g. five or ten years. Once the structure of the supply chain is fixed, one needs to decide the best way to operate it. Since supply chains are large systems, and there are many decisions to be made, it is customary to optimize supply chain operations over different lengths of the time horizon and at different levels of model accuracy. When optimizing operation over long time horizons, e.g. over one year, it is important to determine how much of each product will be produced from one season to another, while it is not important to decide at what specific time some equipment will change from one operation mode to another. The latter is a detailed decision which can be made e.g. one or two weeks prior to that day.

Scheduling is an activity that determines the operating states and their sequence for each equipment, amount of feed to be processed by each instance of an operating state, amount and time of shipment of each product, allocation of multipurpose storage to a given service, and others. Scheduling models for oil refinery operations are usually mixed-integer linear programs (e.g.

Gothe-Lundgren et al., 2002; Jia and Ierapetritou, 2003, 2004; Li and Karimi, 2011) even if the underlying processes are nonlinear by nature, in order to decrease the computational burden. Some MINLP models have been published recently for the detailed scheduling of crude oil operations (Li et al., 2012) and the pooling problem (Kolodziej et al., 2013); they have been solved via the latest generation of specialized MINLP solvers. Currently, most scheduling models are usually formulated using a continuous-time representation (i.e. the horizon is divided into several time slots which duration is a variable to determine) since it requires a smaller number of discrete variables compared with the corresponding discrete-time model. One example of commercial scheduling software is Aspen Petroleum Scheduler (Aspen Technology).

Inclusion of sequencing terms in the model formulation makes scheduling a very challenging computational problem. Due to the inherent discrete decisions involved in the scheduling problem, it is at least NP-complete (Birewar and Grossmann, 1990; Pinto et al., 2000) and frequently NP-hard (Terrazas-Moreno and Grossmann, 2011); this means that there are no polynomial time bounded algorithms to solve this type of problems. Within this current paradigm, it is not possible to guarantee computation of optimal solutions in a reasonable amount of time by a general algorithm when scheduling problems grow beyond a certain model size (which depends on the problem type and instance). Most successful solution strategies are those that employ algorithms tailored to a specific class of scheduling problems and scale up to solve large problems within acceptable computational times. Necessity for such approaches has been presented as “no free lunch theorems for optimization” by Wolpert and Macready (1997).

* Corresponding author. Tel.: +1 905 525 9140x26386.
E-mail address: mahalec@mcmaster.ca (V. Mahalec).

Nomenclature

Sets and indices

BL = {*bl*} blenders
E = {*e*} quality properties (e.g. research and motor octane number)
I = {*i*} blend components
J = {*j*} product tanks
K = {*k*} L1-periods (time periods defined for the 1st level discrete-time model)
L = {*l*} L-intervals (non-overlapping subintervals of the scheduling horizon for the 3rd level continuous-time model)
M = {*m*} L2-periods (time periods defined for the 2nd level discrete-time model)
NO = {*n* | 0, 1, ..., *N*} time slots assigned for the entire horizon (3rd level continuous-time model)
N1 = {*n* | 1, ..., *N*} subset of **NO**, does not include first time slot
N2 = {*n* | 0, 1, ..., *N* - 1} subset of **NO**, does not include last time slot
N3 = {*n* | 1, ..., *N* - 1} subset of **NO**, does not include first and last time slots
NO_l, **N1_l**, **N2_l**, **N3_l** analogous sets to **NO**, **N1**, **N2**, and **N3**, respectively, defined for each L-interval
O = {*o*} all demand orders
O_l demand orders that can be delivered during L-interval *l*
P = {*p*} different products
BP = {(*p*, *bl*)} blender *bl* can produce product *p*
JO = {(*j*, *o*)} product tank *j* can deliver order *o*
KM = {(*k*, *m*)} L2-period *m* is within L1-period *k*
KN = {(*k*, *n*)} time slot *n* is within L1-period *k*
KNend = {(*k*, *n*)} time slot *n* is the last one of L1-period *k*
LM = {(*l*, *m*)} L2-period *m* is within L-interval *l*
LN = {(*l*, *n*)} time slot *n* is within L-interval *l*
LNend = {(*l*, *n*)} last time slot of L-interval *l*
MN = {(*m*, *n*)} time slot *n* is within L2-period *m*
PO = {(*p*, *o*)} order *o* consists of product *p*
JON = {(*j*, *o*, *n*)} product tank *j* can deliver order *o* during time slot *n*

Parameters

*c*₂(*bl*) cost associated with one blend run in blender *bl*
*c*₃(*j*) cost associated with a product transition in swing tank *j*
*c*₄ cost associated with a delivery run
 Demand(*o*) demand quantity of order *o*
*F*_{blend}^{min}(*bl*), *F*_{blend}^{max}(*bl*) minimum and maximum blending rates of blender *bl*
H length of the entire scheduling horizon
*of*_{L2}(*o*, *m*) if equal to 1, delivery window of order *o* spans, partially or completely, L2-period *m*
*Q*_{bcL1}(*i*, *e*, *k*) quality *e* of blend component *i* during L1-period *k*
*Q*_{pr}^{min}(*p*, *e*), *Q*_{pr}^{max}(*p*, *e*) minimum and maximum specifications for quality property *e* and product *p*
*T*_{L1}^{end}(*k*) end time of L1-period *k*
*T*_{L2}^{end}(*m*) end time of L2-period *m*
*T*_{L3}^{end}(*l*) end time of L-interval *l*
*V*_{pr,L3}^{target}(*j*, *n*) inventory target for product tank *j* at time slot *n*

Binary variables

*u*_{L2}(*j*, *p*, *m*) binary variable at the 2nd level, but a parameter at the 3rd level. If equal to 1, product tank *j* is storing product *p* during L2-period *m*
*u*_{L3}(*j*, *p*, *n*) if equal to 1, product tank *j* is storing product *p* during slot *n*
*x*_{L2}(*p*, *bl*, *m*) binary variable at the 2nd level, but a parameter at the 3rd level. If equal to 1, blender *bl* is processing product *p* during slot *n* during L2-period *m*
*z*_{L2}(*j*, *o*, *m*) binary variable at the 2nd level, but a parameter at the 3rd level. If equal to 1, product tank *j* delivers material to order *o* during L2-period *m*
*z*_{L3}(*j*, *o*, *n*) if equal to 1, product tank *j* delivers material to order *o* during slot *n*
 0–1 continuous variables
*x*_{L3}(*p*, *bl*, *n*) if equal to 1, blender *bl* is processing product *p* during slot *n*

Continuous variables

*Deliver*_{order,L2}(*j*, *o*, *m*) volume delivered from product tank *j* to order *o* during L2-period *m*
*Deliver*_{pr,L2}(*j*, *p*, *m*) volume delivered of product *p* from product tank *j* during L2-period *m*
*DV*_{pr}(*j*, *o*, *n*) volume delivered from product tank *j* to order *o* during slot *n*
*Q*_{pr,L1}(*p*, *e*, *k*) quality *e* of product *p* during L1-period *k*
*r*_{L1}(*i*, *p*, *k*) continuous variable at the 1st level, but a parameter at the 2nd and 3rd levels. Blend recipe for product *p* in L1-period *k*
*T*_{b,L3}(*bl*, *n*) end time of a blender slot *n*
*T*_{bc,L3}(*i*, *n*) end time of a component tank slot *n*
*T*_{pr,L3}(*j*, *n*) end time of a product tank slot *n*
*ue*_{L2}(*j*, *m*) if equal to 1, product tank *j* changes its service at the end of L2-period *m*
*ue*_{L3}(*j*, *n*) if equal to 1, product tank *j* changes its service at the end of slot *n*
*V*_{blend,L3}(*bl*, *n*) volume blended during slot *n* by blender *bl*
*V*_{comp,L1}(*i*, *p*, *k*) volume required of component *i* according to the aggregated model solution
*V*_{comp,L3}(*i*, *bl*, *n*) volume of component *i* used in blender *bl* during slot *n*
*V*_{pr,L3}(*j*, *n*) inventory level of product tank *j* at the end of slot *n*
*ze*_{L2}(*o*, *m*) number of different product tanks delivering material to order *o* during L2-period *m*

Decomposition strategies are usually employed to handle large-scale scheduling problems. Bassett et al. (1996) presented various heuristic methods for discrete-time formulations of scheduling problems, most of them using time-based decompositions. Wu and Ierapetritou (2003) reviewed several heuristic approaches for scheduling problems with continuous-time models, including methods using time- and resource-based decompositions. They also included a review of rigorous mathematical approaches such as Lagrangean relaxation and Lagrangean decomposition.

A major part of an oil refinery's profit comes from gasoline (Mendez et al., 2006; Li and Karimi, 2011). Therefore, determining the best possible way to mix the refinery's intermediate products (blend components) to produce the different gasoline grades is an important task. For that reason, inclusion of recipe optimization in gasoline blend scheduling models has become the norm in the last ten years. Both discrete- and continuous-time formulations have been developed to solve the gasoline blend scheduling problem. In

order to develop a mixed-integer linear model, quality properties are transformed into blend indices that blend linearly on a volumetric or weight basis (e.g. Li et al., 2010; Li and Karimi, 2011). A different approach is to solve a sequence of MILP models in order to converge to the proper quality values (e.g. Mendez et al., 2006). As well, stochastic methods have been proposed to solve nonlinear recipe optimization problems (e.g. Chen and Wang, 2010; Zhao and Wang, 2011); however, no operational features nor logistic constraints are considered in such cases.

Commercial MILP solvers may require several hours to obtain the optimal solution of detailed scheduling models for medium- and large-scale problems (e.g. Li and Karimi, 2011; Shah and Ierapetritou, 2011). Castillo-Castillo and Mahalec (2015) presented a detailed continuous-time scheduling model with reduced number of discrete variables, and although such model was able to solve some large-scale examples to optimality in less time than previously published models, execution times still can reach more than 12 h.

Glismann and Gruhn (2001) proposed a two-level approach to solve the blend scheduling problem: at the top level, a discrete-time NLP model computes blend recipes and production targets, and at the lower level, a discrete-time MILP model solves the short term scheduling problem using the recipes and targets from the top level. An iterative procedure is required to handle possible infeasibilities. The scheduling model is based on a resource-task-network representation. The scheduling model does not consider multipurpose tanks (i.e. swing tanks) nor the delivery scheduling problem (i.e. distribution or shipping problem).

Castillo and Mahalec (2014b) presented a three-level decomposition approach to solve the gasoline blending problem using discrete-time models at each level. They included recipe optimization using linear or nonlinear models, blend size threshold constraints, and most of the operational features described by Li and Karimi (2011). The discrete-time formulation for scheduling horizons of 1 or 2 weeks with 1-h time periods resulted in a large size scheduling model at the 3rd level that required to be solved in subintervals. These subintervals were solved sequentially, first in a forward direction and then in a reverse direction. Solutions computed by this approach were better and the execution times for large problems were two orders of magnitude shorter than those from previous works (Li et al., 2010; Li and Karimi, 2011). For most of their nonlinear examples, their algorithm computed better solutions than MINLP solvers BARON, ANTIGONE, and GloMIQO, with execution times an order of magnitude shorter. Analogously to Li and Karimi (2011), Castillo and Mahalec (2014b) did not penalize the delivery of the same product order from different tanks.

This work introduces Multi-Period Inventory Pinch-Continuous time algorithm (MPIP-C) for scheduling of processes described by linear and nonlinear models. MPIP-C algorithm decomposes the original scheduling problem into (i) blend recipe optimization, (ii) approximate scheduling, and (iii) detailed scheduling. The 1st level determines the blend recipes (by solving a discrete-time LP or NLP model), and the 2nd level computes an approximate schedule via discrete-time MILP. The time periods at the 1st level are initially delineated by inventory pinch points (Castillo et al., 2013; Castillo and Mahalec, 2014a). Discrete-time model for approximate scheduling is a modification of a model used in our previous work (Castillo and Mahalec, 2014a). The 3rd level model (i.e. the detailed scheduling problem) is a continuous-time MILP which includes additional constraints arising from the approximate scheduling solution. If there are infeasibilities encountered at the 2nd or the 3rd level, they are resolved by subdividing the corresponding periods at the 1st level. The problem does not have a feasible solution if the 1st level is infeasible.

The rest of this article is structured as follows. Section 2 presents the problem statement. Section 3 presents an overview of the

algorithm and of the models used at each level. Section 4 describes the examples used in this work. Section 5 discusses the results obtained using the MPIP-C algorithm. As summarized in Section 6, the computational results show that the proposed MPIP-C algorithm computes optimal or near-optimal solutions with execution times which are much smaller (for both linear and nonlinear models) than those required by the full-space model.

2. Problem statement

The gasoline blend scheduling problem is summarized as follows:

Given a short-term scheduling horizon, a set of blend components and their properties, a supply profile of blend components, a set of products and their property specifications, a set of delivery orders for each product, a blending system (i.e. storage tanks, blenders, and their interconnections) and its initial conditions, determine (a) the blend recipes, (b) production and delivery sequences, (c) inventory profiles, and (d) swing tanks product allocation, while minimizing the cost of the blended materials plus the switching costs (i.e. number of blend runs, number of tanks delivering the same order, and product transitions in the swing tanks) and the demurrage costs.

The blending system is subject to the following constraints:

1. If a blender is to produce a product, it must blend at least a minimum amount.
2. A blender can produce at most one product at any time. Once it begins blending, it must operate for some minimum time before it can switch to another product.
3. A blender requires a minimum setup time during a product changeover
4. A blender can feed at most one product tank at any time (industrial practice).
5. Product tanks can only store one product at any time.

The assumptions made are:

1. Flow rate profile of each component from the upstream process is piecewise constant.
2. Component quality profile is piecewise constant.
3. Perfect mixing occurs in each blender.
4. There is only one tank for a given blend component.
5. Only product tanks defined as swing tanks can change its product service (i.e. change from storing one product to store another).
6. Changeover times between products are negligible for swing tanks.
7. For each blender, changeover times between product blending are product-dependent but sequence-independent.
8. Each order involves only one product (one original order involving different products can be broken into orders of each specific product).
9. Each order is completed during the scheduling horizon.

For illustration, Fig. 1 shows a sample blending system with five dedicated component tanks, two blenders, four product tanks (two dedicated tanks and two swing tanks), and three different finished products.

3. Inventory-pinch based scheduling algorithm combining discrete-time and continuous-time models

This work employs a three-level decomposition of the problem as shown in Fig. 2, which is similar to the one presented by Castillo and Mahalec (2014b) but it includes several important

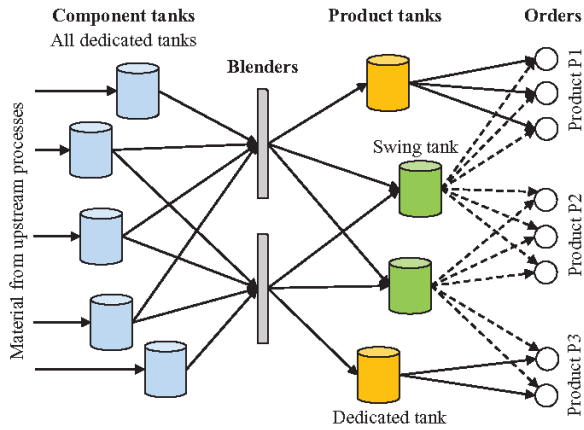


Fig. 1. Sample gasoline blending system.

differences. In order to speed-up computations, there is no product sequencing at the 2nd level, and the model at the 3rd level employs a continuous-time formulation instead of a discrete representation of the time domain. Since the proposed algorithm is a continuous-time version of the multiperiod inventory pinch algorithm, it is denoted as MPIP-C algorithm. In this section we describe our decomposition approach and the equations required at each level. Subscripts $L1$, $L2$, and $L3$ identify to which level a variable or parameter corresponds.

3.1. Time grids, time slots, and the inventory pinch concept

The inventory pinch points (see Fig. 3) correspond to the times where the cumulative average total production (CATP) curve

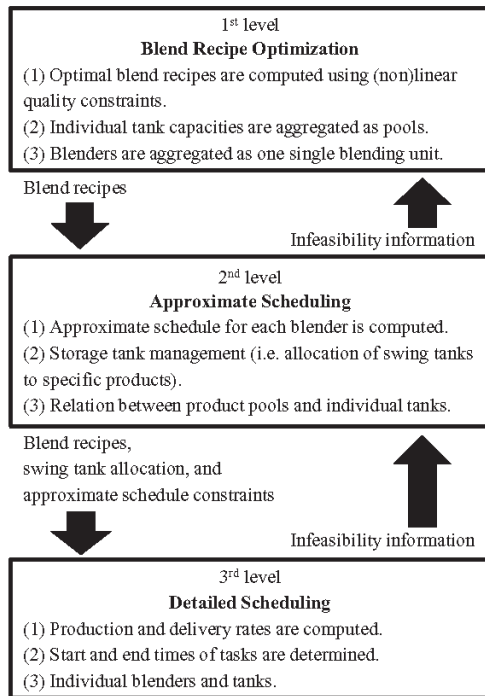


Fig. 2. Decomposition framework for blend scheduling.

changes its slope in order to remain above the cumulative total demand (CTD) curve (Castillo et al., 2013; Castillo and Mahalec, 2014a). Product inventories are at the minimum allowed limits at the inventory pinch points. The concept can be applied directly to multiple products (i.e. the demand of all products is aggregated in such case), and minimum inventory limits and target inventories can be incorporated easily.

The algorithm starts at the 1st level with determination of the inventory pinch points, which delineate the periods at this level ($L1$ -periods). Additional $L1$ -periods are delineated by the times when changes in the quality or price of the blend components take place. If there are no inventory constraints, optimal blend recipes between the pinch points remain constant. Once blend recipes are computed at the 1st level, they are used as fixed blend recipes at the 2nd level to compute an approximate schedule via discrete-time MILP model. The set of $L1$ -periods is $\mathbf{K} = \{1, \dots, k, \dots, K-1, K\}$, and the set of $L2$ -periods is $\mathbf{M} = \{1, \dots, m, \dots, M-1, M\}$. An integer number of $L2$ -periods compose an $L1$ -period and this is encoded in set $\mathbf{KM} = \{(k, m)\}$.

$L2$ -periods are chosen based on operational decisions. $L2$ -periods must be long enough to complete the smallest allowed blend run and they must be longer than the minimum time that a swing tank may hold a specific product. $L2$ -periods are not required to have the same length. Time period boundaries of $L2$ -periods are delineated by the period boundaries from the 1st level, points in time when the supply profile of blend components changes, and by clock-time chosen by the scheduler. Additionally, it is important to reduce as much as possible the number of order delivery windows spanning more than one $L2$ -period. This reduction of the time delivery windows will decrease the number of binary variables associated with the delivery of orders (i.e. $z_{L2}(j,o,m)$ and $z_{L3}(j,o,n)$). The following guidelines help to determine if a time delivery window can be reduced:

1. The order can be fulfilled within the adjusted delivery window by a single tank.
2. Adjustments to the time delivery windows should not move existent inventory pinch points nor generate new inventory pinch points (otherwise the minimum blend cost will increase).
3. Based on the previous guideline, delaying the start of a delivery window is preferred instead of shortening the end time of the window.

These guidelines are easy to check without solving an optimization problem. Fig. 4 shows a graphical example where some of the original delivery windows are contracted.

The 3rd level model uses a continuous-time formulation with time slots and specific time grids for each unit. The continuous-time model used in this work is the one presented in Castillo-Castillo and Mahalec (2015) since it considers several key operational constraints without many discrete variables. The model as presented in Castillo-Castillo and Mahalec (2015) is referred in this work as the full-space model. Following the same principle as with the 1st and 2nd levels, an integer number of time slots is associated with an $L2$ -period. Set $\mathbf{MN} = \{(m, n)\}$ is used to represent this information. Eqs. (1)–(3) indicate that the slots associated with $L2$ -period m should end within the interval of such $L2$ -period. Note that the first slot associated with $L2$ -period m may start during $m-1$. Fig. 5 shows an example of the time grids at each level.

$$T_{L2}^{end}(m-1) \leq T_{bc,L3}(i, n) \leq T_{L2}^{end}(m) \quad \forall i, n \in \mathbf{N1}, m : (m, n) \in \mathbf{MN} \quad (1)$$

$$T_{L2}^{end}(m-1) \leq T_{pr,L3}(j, n) \leq T_{L2}^{end}(m) \quad \forall j, n \in \mathbf{N1}, m : (m, n) \in \mathbf{MN} \quad (2)$$

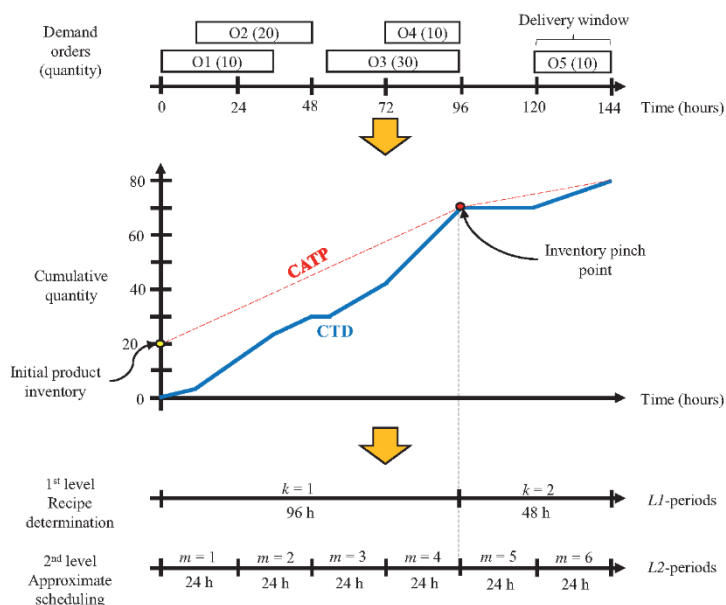


Fig. 3. Example of an inventory pinch point and time grids.

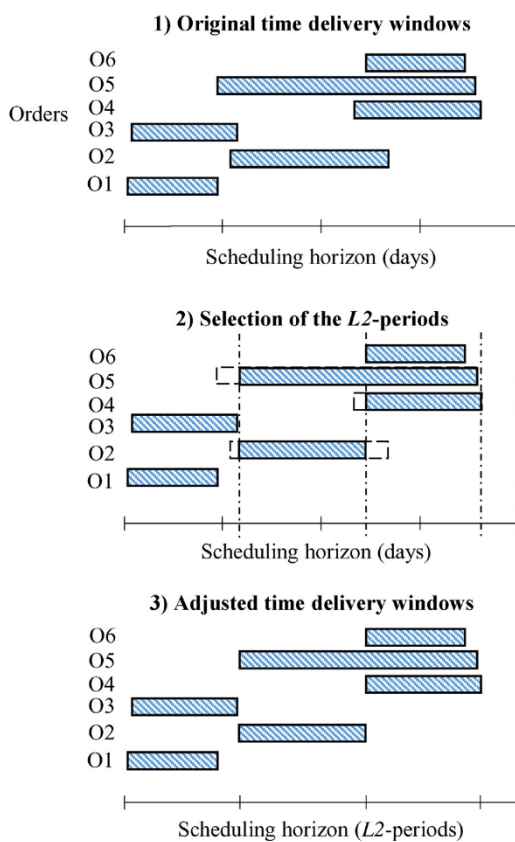


Fig. 4. Time delivery windows can be adjusted according to the L2-period boundaries.

$$T_{L2}^{end}(m-1) \leq T_{b,L3}(bl, n) \leq T_{L2}^{end}(m) \quad \forall bl, n \in \mathbf{N1}, m : (m, n) \in \mathbf{MN} \quad (3)$$

In this decomposition framework, the boundaries of the L2-periods delineate the boundaries of the fixed time periods of the continuous-time model from Castillo-Castillo and Mahalec (2015) and they are used to define set $\mathbf{ON} = \{(o, n) \mid \text{order } o \text{ may be delivered during time slot } n\}$ and set $\mathbf{JON} = \{(j, o, n) \mid \text{tank } j \text{ may deliver order } o \text{ during time slot } n\}$.

3.2. Optimizing operating conditions (L1 level)

The 1st level determines which operating conditions are going to be used at the following levels. For the gasoline blending problem, these operating conditions are the blend recipes. The 1st level model uses a discrete-time formulation with no integer variables. Depending on the blending rules being used, this model will be either linear or nonlinear. We use the model presented by Castillo and Mahalec (2014a) which uses the inventory pinch concept to delineate the boundaries of the L1-periods, blenders are lumped into a single one, and individual product tanks are modeled as product pools. Only demand, product quality, inventory capacity, and maximum blending capacity constraints are considered. The solution of the 1st level model represents a lower bound for the blend cost of the original problem (Castillo et al., 2013, Castillo and Mahalec, 2014a, 2014b) and it is imposed as such in the 3rd level model.

In this work, some of the examples assume the research octane number (RON) and the motor octane number (MON) as properties that blend nonlinearly following the ethyl RT-70 models (Singh et al., 2000; Healey et al., 1959). Such models define RON and MON as functions of the blend components sensitivity (i.e. $sens(i) = Q_{bc}(i; \text{RON}') - Q_{bc}(i; \text{MON}')$), and olefins and aromatics content. The model parameter values used by Singh et al. (2000) are: $a_1 = 0.03224$, $a_2 = 0.00101$, $a_3 = 0$, $a_4 = 0.04450$, $a_5 = 0.00081$, and

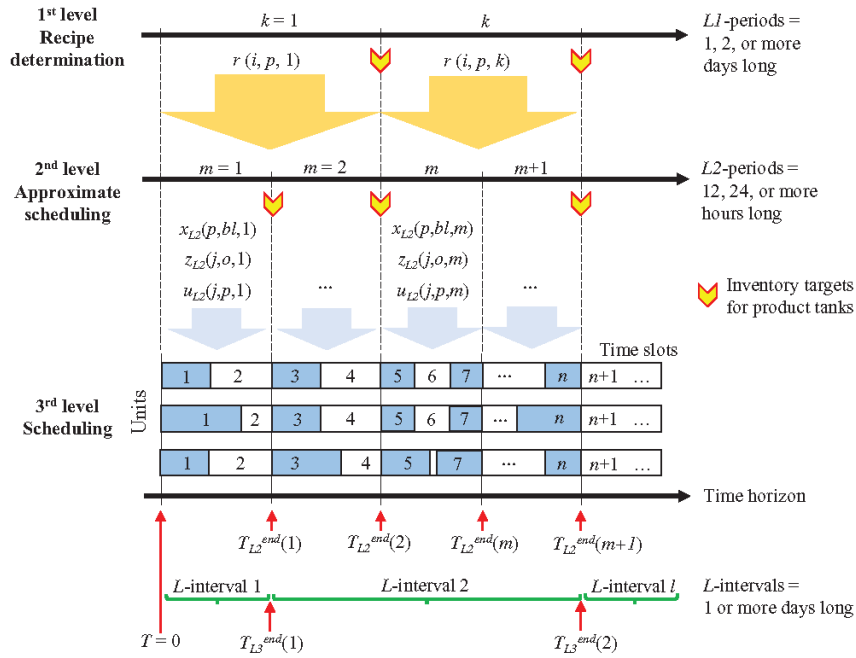


Fig. 5. Graphical example of the proposed decomposition approach.

$a_6 = -0.0645$. Eqs. (4)–(14) are appended to the 1st level model from Castillo and Mahalec (2014a) for those specific examples.

$$r_{avg,L1}^{RON}(p, k) = \sum_i r_{L1}(i, p, k) \cdot Q_{bc}(i, e, k) \quad \forall e \neq RON', p, k \geq 1 \quad (4)$$

$$Ol_{avg,L1}(p, k) = \sum_i r_{L1}(i, p, k) \cdot Q_{bc}(i, e, k) \quad \forall e \neq OLF', p, k \geq 1 \quad (9)$$

$$r_{avg,L1}^{MON}(p, k) = \sum_i r_{L1}(i, p, k) \cdot Q_{bc}(i, e, k) \quad \forall e \neq MON', p, k \geq 1 \quad (5)$$

$$Ar_{avg,L1}(p, k) = \sum_i r_{L1}(i, p, k) \cdot Q_{bc}(i, e, k) \quad \forall e \neq ARO', p, k \geq 1 \quad (10)$$

$$sens_{avg,L1}(p, k) = \sum_i r_{L1}(i, p, k) \cdot sens(i) \quad \forall i, p, k \geq 1 \quad (6)$$

$$Ol_{avg,L1}^{sq}(p, k) = \sum_i r_{L1}(i, p, k) \cdot [Q_{bc}(i, e, k)]^2 \quad \forall e \neq OLF', p, k \geq 1 \quad (11)$$

$$sens_{avg,L1}^{RON}(p, k) = \sum_i r_{L1}(i, p, k) \cdot Q_{bc}(i, e, k) \cdot sens(i) \quad \forall e \neq RON', p, k \geq 1 \quad (7)$$

$$sens_{avg,L1}^{MON}(p, k) = \sum_i r_{L1}(i, p, k) \cdot Q_{bc}(i, e, k) \cdot sens(i) \quad \forall e \neq MON', p, k \geq 1 \quad (8)$$

$$Ar_{avg,L1}^{sq}(p, k) = \sum_i r_{L1}(i, p, k) \cdot [Q_{bc}(i, e, k)]^2 \quad \forall e \neq ARO', p, k \geq 1 \quad (12)$$

$$\begin{aligned} Q_{pr,L1}(p, e, k) &= r_{avg,L1}^{RON}(p, k) + a_1 (sens_{avg,L1}^{RON}(p, k) - r_{avg,L1}^{RON}(p, k) \cdot sens_{avg,L1}(p, k)) \\ &+ a_2 \left(Ol_{avg,L1}^{sq}(p, k) - [Ol_{avg,L1}(p, k)]^2 \right) \\ &+ a_3 \left([Ar_{avg,L1}^{sq}(p, k)]^2 - 2 [Ar_{avg,L1}(p, k)] [Ar_{avg,L1}(p, k)]^2 + [Ar_{avg,L1}(p, k)]^4 \right) \\ &\forall e \neq RON', p, k \geq 1 \end{aligned} \quad (13)$$

$$\begin{aligned} Q_{pr,L1}(p, e, k) &= r_{avg,L1}^{MON}(p, k) + a_4 (sens_{avg,L1}^{MON}(p, k) - r_{avg,L1}^{MON}(p, k) \cdot sens_{avg,L1}(p, k)) \\ &+ a_5 \left(Ol_{avg,L1}^{sq}(p, k) - [Ol_{avg,L1}(p, k)]^2 \right) \\ &+ \left(\frac{a_6}{10000} \right) \left([Ar_{avg,L1}^{sq}(p, k)]^2 - 2 [Ar_{avg,L1}(p, k)] [Ar_{avg,L1}(p, k)]^2 + [Ar_{avg,L1}(p, k)]^4 \right) \\ &\forall e \neq MON', p, k \geq 1 \end{aligned} \quad (14)$$

3.3. Approximate scheduling model (L2 level)

The 2nd level determines an approximate schedule via a discrete-time MILP model with fixed operating conditions as defined by the 1st level solution. As mentioned in Section 3.1, the L2-periods are selected depending on operational decisions. The 2nd level model links the product pools with individual tanks, involves decisions such as product allocation of swing tanks, which product and how much to blend in each blender, and which order to deliver from each tank and how much, in each L2-period. More than one product can be blended in each blending unit in a given L2-period. Constraints such as minimum blend size, minimum running times and setup times for the blenders, inventory limits, and minimum and maximum blending rates are included in this model. Eqs. (16)–(22) are appended to the 2nd level model presented by Castillo and Mahalec (2014a) in order to determine which tanks will deliver specific demand orders and try to minimize the number of different tanks sending product to the same order. Eq. (15) is the objective function when minimizing switching costs, which includes a penalty for delivering an order from different tanks.

Eq. (15) minimizes the number of blend runs, product changeovers in the swing tanks, and deliveries to the same order from different tanks (during the same L2-period). Eq. (16) forces all blenders to work at some point during the scheduling horizon (this equation assumes that the number of blend runs required is greater than the number of blenders). Eq. (17) establishes that the delivered amount of a certain product from one tank is equal to the delivered amount from that tank to the specific orders constituted by that product, during a L2-period. Eq. (18) constraints the delivery of order o to be during a L2-period which contains completely or partially the corresponding delivery window (i.e. the parameter $of_{L2}(o,m)=1$ in such a case). Eq. (19) observes that a tank can only deliver order o if it is holding the type of product associated with that order. Eq. (20) constraints binary variable $z_{L2}(j,o,m)$ to be equal to 1 if the tank j is delivering order o during L2-period m . Eq. (21) establishes that the delivered amount must be equal to the demand. Continuous variable $ze_{L2}(o,m)$ represents the number of delivery runs from all tanks to order o during L2-period m , minus one. Eq. (22) constrains the minimum value for $ze_{L2}(o,m)$, which is penalized in Eq. (15). The term -1 in Eq. (22) appears because at least one delivery run is always needed from one tank if the decision is to send product.

$$\min Z_{L2} = \sum_{m=1}^M \left(\sum_{(bl,p) \in \mathbf{BP}} c_2(bl) \cdot x_{L2}(p, bl, m) + \sum_j c_3(j) \cdot ue_{L2}(j, m) \right) + c_4 \cdot \sum_{m=1}^M \sum_o ze_{L2}(o, m) \quad (15)$$

$$\sum_{m=1}^M \sum_{p:(p,bl) \in \mathbf{BP}} x_{L2}(p, bl, m) \geq 1 \quad \forall bl \quad (16)$$

$$Deliver_{p,r,L2}(j, p, m) = \sum_{o:(p,o) \in \mathbf{PO}} Deliver_{order,L2}(j, o, m) \quad \forall (j, p) \in \mathbf{JP}, m \geq 1 \quad (17)$$

$$Deliver_{order,L2}(j, o, m) \leq Demand(o) \cdot of_{L2}(o, m) \quad \forall (j, o) \in \mathbf{JO}, m \geq 1 \quad (18)$$

$$Deliver_{order,L2}(j, o, m) \leq Demand(o) \cdot u_{L2}(j, p, m) \quad \forall (j, p) \in \mathbf{JP}, (j, o) \in \mathbf{JO}, (p, o) \in \mathbf{PO}, m \geq 1 \quad (19)$$

$$Deliver_{order,L2}(j, o, m) \leq Demand(o) \cdot z_{L2}(j, o, m) \quad \forall (j, o) \in \mathbf{JO}, m \geq 1 \quad (20)$$

$$\sum_{m=1}^M \sum_{j:(j,o) \in \mathbf{JO}} Deliver_{order,L2}(j, o, m) = Demand(o) \quad \forall o \quad (21)$$

$$ze_{L2}(o, m) \geq \left[\sum_{j:(j,o) \in \mathbf{JO}} z_{L2}(j, o, m) \right] - 1 \quad \forall o, m \geq 1 \quad (22)$$

3.4. Continuous-time scheduling model (L3 level)

The 3rd level scheduling model is based on the full-space model presented in Castillo-Castillo and Mahalec (2015). After solving the 1st and 2nd levels, there is some information that can be integrated into the 3rd level model. The operating conditions from the 1st level (i.e. the blend recipes), the swing tank allocation, and the approximate production and delivery schedule from the 2nd level can be fixed. This will reduce the model size and lead to shorter execution times. For large problems it may be more desirable to use a time-based decomposition of the scheduling horizon, i.e. the horizon is subdivided in smaller intervals which are solved in sequence. Note that since the blend recipes, the swing tank allocation, and the approximate production and delivery schedule are fixed, the solution from the 3rd level scheduling model will be always equal to or higher than the optimal solution from the original full-space model.

3.4.1. Fixing the blend recipes

Different scheduling problems in the literature consider the use of preferred blend recipes and, from an operational point of view, keeping the blend recipes as constant as possible may be desired. At the 3rd level, we fixed the blend recipes to be equal to those computed at the 1st level. Eqs. (23) and (24) are added in order to fix the blend recipes according to the 1st level solution, i.e. $r_{L1}(i,p,k)$. Equations corresponding to product composition and quality constraints are dropped from the continuous-time scheduling model at the 3rd level since the blend recipes are fixed. Therefore, the 3rd level scheduling model is always a MILP, even if the full-space is a MINLP model (i.e. it uses nonlinear blending equations). The set $\mathbf{KN} = \{(k, n)\}$ indicates which time slots are within each L1-period (it can be constructed from the intersection of sets \mathbf{KM} and \mathbf{MN}). A small tolerance factor δ_1 is included in order to account for numerical differences between the levels, e.g. in our case studies $\delta_1 = 1 \times 10^{-6}$.

$$V_{comp,L3}(i, bl, n) \geq (r_{L1}(i, p, k) - \delta_1) \cdot V_{blend,L3}(bl, n) - F_{blend}^{\max}(bl) \cdot H \cdot (1 - x_{L3}(p, bl, n)) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, k : (k, n) \in \mathbf{KN} \quad (23)$$

$$V_{comp,L3}(i, bl, n) \leq (r_{L1}(i, p, k) + \delta_1) \cdot V_{blend,L3}(bl, n) + F_{blend}^{\max}(bl) \cdot H \cdot (1 - x_{L3}(p, bl, n)) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, k : (k, n) \in \mathbf{KN} \quad (24)$$

Eq. (25) constrains the last blend slots of an $L1$ -period to be equal to the time when such $L1$ -period ends (i.e. when the blend recipe changes). Eq. (6) is required to compute a feasible solution more easily when using the time-based decomposition described in Section 3.4.5.

$$T_{b,L3}(bl, n) = T_{L1}^{end}(k) \quad \forall bl, k, n : (k, n) \in \mathbf{KNend} \quad (25)$$

3.4.2. Fixing the swing tank allocation

By exploiting the fact that the tank allocation is known from the 2nd level solution, before solving the 3rd level model, binary variable $u_{L3}(j,p,n)$ becomes a parameter by fixing its values using the following relationship: $u_{L3}(j,p,n) = u_{L2}(j,p,m), \forall (j,p) \in \mathbf{JP}, n \in \mathbf{N1}, m : (m, n) \in \mathbf{MN}$.

3.4.3. Reducing the number of possible production sequences

It may be possible to infer that, during a subinterval of the horizon, only some specific products are (or are not) required to be produced. The solution from the 2nd level is used to implement this idea. Eq. (26) is added in order to constrain the scheduling model to the 2nd level solution. This reduces the number of possible production sequences.

$$x_{L3}(p, bl, n) \leq x_{L2}(p, bl, m) \quad \forall (p, bl) \in \mathbf{BP}, n \in \mathbf{N1}, m : (m, n) \in \mathbf{MN} \quad (26)$$

3.4.4. Reducing the number of possible delivery sequences

The approximate delivery schedule can be used to construct set **JON**. Variable $z_{L2}(j,o,m)$ from the 2nd level is equal to 1 if product tank j delivers material to order o during $L2$ -period m , and 0 otherwise. Therefore, set **JON** = $\{(j,o,n) \mid z_{L2}(j,o,m) = 1 \text{ for } (j,o) \in \mathbf{JO}, (m,n) \in \mathbf{MN}\}$ in the MPIP-C algorithm.

3.4.5. Time-based decomposition

Although the reduction of binary variables, use of fixed recipes, and constraints from the approximate scheduling decisions allow to solve larger problems (meaning e.g. larger horizons, more different products, swing tanks, blenders), at some point the model size will become large enough that the execution times will become prohibitive once again. In such a case, we use a time-based decomposition of the scheduling horizon, i.e. the horizon is partitioned in subintervals denoted as L -intervals, and these subintervals are solved sequentially starting from the first one. This approach does not guarantee global optimal solutions but will lead to good quality solutions very rapidly.

An integer number of $L2$ -periods constitute one L -interval and set **LM** = $\{(l, m)\}$ indicates such relation. The time slots associated with those $L2$ -periods become associated with the corresponding L -interval. Analogously to the full-space continuous-time model, the following sets are defined for each L -interval model:

N0_l = { n all the time slots belonging to L -interval l }
N1_l = { n all the time slots belonging to L -interval l , minus the first slot of the L -interval}
N2_l = { n all the time slots belonging to L -interval l , minus the last slot of the L -interval}
N3_l = { n all the time slots belonging to L -interval l , minus the first and last slots of the L -interval}
O_l = { o order o can be delivered within L -interval l according to the 2nd level solution}

Sets **N0_l**, **N1_l**, **N2_l**, **N3_l**, and **O_l** replace the previous sets **N0**, **N1**, **N2**, **N3**, and **O**, respectively, in the continuous-time scheduling model from Castillo-Castillo and Mahalec (2015) in order to solve the model only for L -interval l . Eq. (27) fixes the end time for the last unit slots in an L -interval. Fig. 5 shows an example of the time

horizon at the 3rd level decomposed into L -intervals. Note that, in order to link the L -intervals, the last slot of an L -interval is the first slot of the next L -interval.

$$\begin{aligned} T_{bc,L3}(i, n) &= T_{pr,L3}(j, n) = T_{b,L3}(bl, n) \\ &= T_{L3}^{end}(l) \quad \forall i, j, bl, l, n : (l, n) \in \mathbf{LNend} \end{aligned} \quad (27)$$

3.4.6. Product inventory targets

When solving the scheduling problem with more than one L -interval and using only the sequential approach, extra constraints are required in order to reduce backtracking due to infeasible solutions. Since both the 1st and the 2nd level are solved for the entire horizon, their solutions provide global information. We have chosen the product inventory levels at the end of each L -interval and at the end of each recipe regimen to match those from the corresponding 2nd level solution, since the component tanks will be then associated by the fixed blend recipe. Therefore the set of time slots that require these constraints is **Ntarget** = $\{n \mid n \text{ is the last slot of an } L\text{-interval or the last slot before the blend recipe changes}\}$.

Eqs. (28) and (29) are added in order to fix the product inventory levels from the 2nd level solution. A small factor δ_2 is included in order to account for numerical differences between the levels, e.g. in our case studies $\delta_2 = 1 \times 10^{-6}$.

$$V_{pr,L3}(j, n) \geq V_{pr,L3}^{target}(j, n) - \delta_2 \quad \forall j, n \in \mathbf{N1}_l, n \in \mathbf{Ntarget} \quad (28)$$

$$V_{pr,L3}(j, n) \leq V_{pr,L3}^{target}(j, n) + \delta_2 \quad \forall j, n \in \mathbf{N1}_l, n \in \mathbf{Ntarget} \quad (29)$$

Where $V_{pr,L3}^{target}(j, n)$ is equal to the inventory level of tank j of the $L2$ -period associated with the end of an L -interval, or a recipe change, computed at the 2nd level.

As an example, the following sets are found in Fig. 5:

LNend = $\{(1, 2), (2, n)\}$
Ntarget = $\{2, 4, n\}$
KM = $\{(1, 1), (1, 2), (k, m), (k, m+1)\}$
MN = $\{(1, 1), (1, 2), (2, 3), (2, 4), (m, 5), (m, 6), (m, 7), (m+1, \dots), (m+1, n)\}$
KN = pairs of elements (k, n) from the intersection of sets KM and MN .
LM = $\{(1, 1), (2, 2), (2, m), (2, m+1)\}$
N0_{l=1} = $\{0, 1, 2\}$, N1_{l=1} = $\{1, 2\}$, N2_{l=1} = $\{0, 1\}$, and N3_{l=1} = $\{1\}$.
N0_{l=2}} = $\{2, 3, 4, 5, 6, 7, \dots, n\}$, N1_{l=2}} = $\{3, 4, 5, 6, 7, \dots, n\}$, N2_{l=2}} = $\{2, 3, 4, 5, 6, 7, \dots, n-1\}$, and N3_{l=2}} = $\{3, 4, 5, 6, 7, \dots, n-1\}$.

Regarding the set **Ntarget**, slot 2 appears in this set because it is the last slot of an L -interval, while slot 4 appears because recipe changes afterwards, and slot n for both previous reasons.

3.4.7. Versions of the 3rd level scheduling model

The 3rd level continuous-time scheduling model is denoted as F-NoSimRD or F-SimRD, depending on the operational scenario regarding simultaneous receipt and delivery by product tanks. "SimRD" means that simultaneous receipt/delivery by product tanks is permitted, and "NoSimRD" indicates that simultaneous receipt/delivery by product tanks is not allowed. The "F" stands for fixed recipes. Model F-NoSimRD is constituted by model L-NoSimRD (Castillo-Castillo and Mahalec, 2015), minus the composition and quality constraints, plus Eqs. (1)–(3), and (23)–(29). Similarly, model F-SimRD is constituted by model L-SimRD (Castillo-Castillo and Mahalec, 2015), minus the composition and quality constraints, plus Eqs. (1)–(3) and (23)–(29).

The 3rd level model is solved in three phases: a feasibility check, an optimization phase, and the schedule adjustment step.

For the feasibility check, the scheduling model is denoted as F-SimRD-feas (or F-NoSimRD-feas). The cost coefficients c_1, c_2, c_3 , and c_4 are equal to zero, i.e. the objective is to find only a feasible solution.

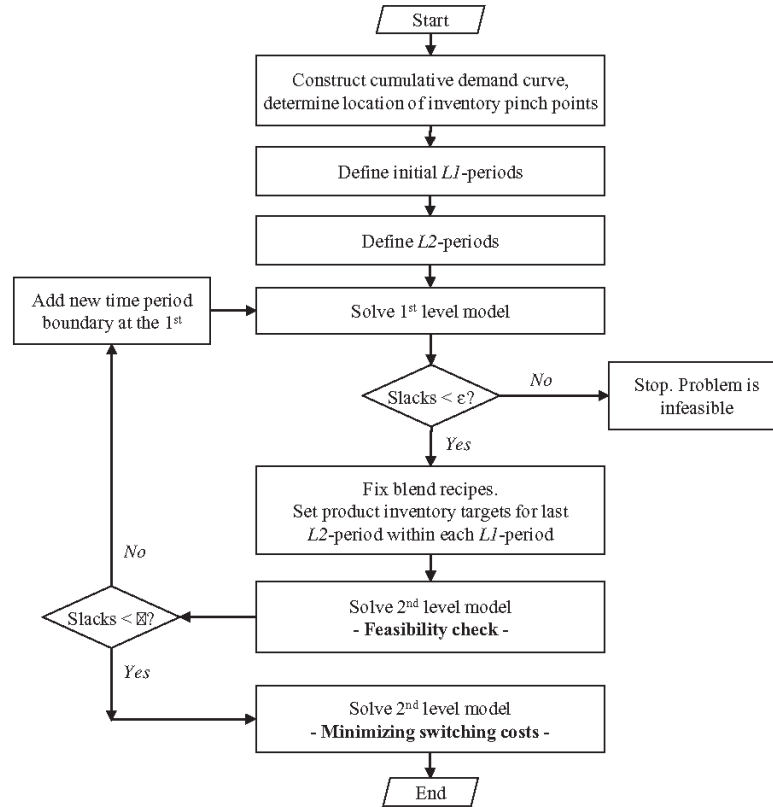


Fig. 6. MPIP approximate scheduling flowchart.

For the optimization phase, all the cost coefficients are used, i.e. the objective is to minimize materials, switching, demurrage, and slack costs. The model is denoted as F-SimRD-opt (or F-NoSimRD-opt).

Model F-SimRD-adj (or F-NoSimRD-adj) is employed for the scheduling adjustment procedure to reduce blending rates variations. In this model, production and delivery sequences are fixed and the objective function minimizes the differences between the average and actual blending rates.

3.5. MPIP-C scheduling algorithm steps

The main steps of the MPIP-C algorithm can be divided into two loops. The inner loop computes the blend recipes and approximate schedule by iterating between the 1st and 2nd level models until the sum of slack variables are equal to or less than the tolerance ϵ . This inner loop is denoted as the MPIP approximate scheduling algorithm and it is shown in Fig. 6 (Castillo and Mahalec, 2014a). Summarizing Section 3.1, the initial boundaries of the L1-periods are based on:

- Inventory pinch points.
- Changes in the quality of blend components.
- Changes in the cost/price of components/products.

And the boundaries of the L2-periods are selected based on:

- Time period boundaries from the 1st level.
- Changes in the supply profile of blend components.
- Order delivery windows.

- Minimum time expected for a swing tank to be in one specific service (i.e. holding one specific product).
- Minimum time expected to complete the smallest blend run allowed.

The outer loop, shown in Fig. 7, deals with the solution of the 3rd level model, determines the number of time slots to obtain a feasible solution or if the blend recipes need to be updated by adding one more L1-period and resolve the 1st level model, thus returning to the inner loop.

If desired, solution from the MPIP-C algorithm can be used as the starting point for the full-space model, which can then be solved via MILP of global MINLP solver. At the beginning of their calculations, these solvers compute rapidly the optimality gap corresponding to the MPIP-C solution; if the optimality gap is sufficiently small, the calculation can stop at that point.

4. Test problems and computing machine used in this study

All the required models were implemented in GAMS IDE 24.3.2. CPLEX 12.6 was used for MILP models, and BARON 14.0.3 and ANTIGONE 1.1 were employed for MINLP models. CPLEX 12.6 and ANTIGONE 1.1 were selected for LP and NLP models at the 1st level, respectively. All problems have been solved on DELL PowerEdge T310 (Intel® Xeon® CPU, 2.40GHz, and 12 GB RAM) running Windows Server 2008 R2 OS.

Two sets of test problems have been used in this study:

Test set #1 is composed by a subset of problems from Li and Karimi (2011): Example number 3, 4, 7, 8, 9, 12, and 14. Scheduling horizon is 8 days, linear blending constraints are employed, and component tanks have different supply flowrates along the

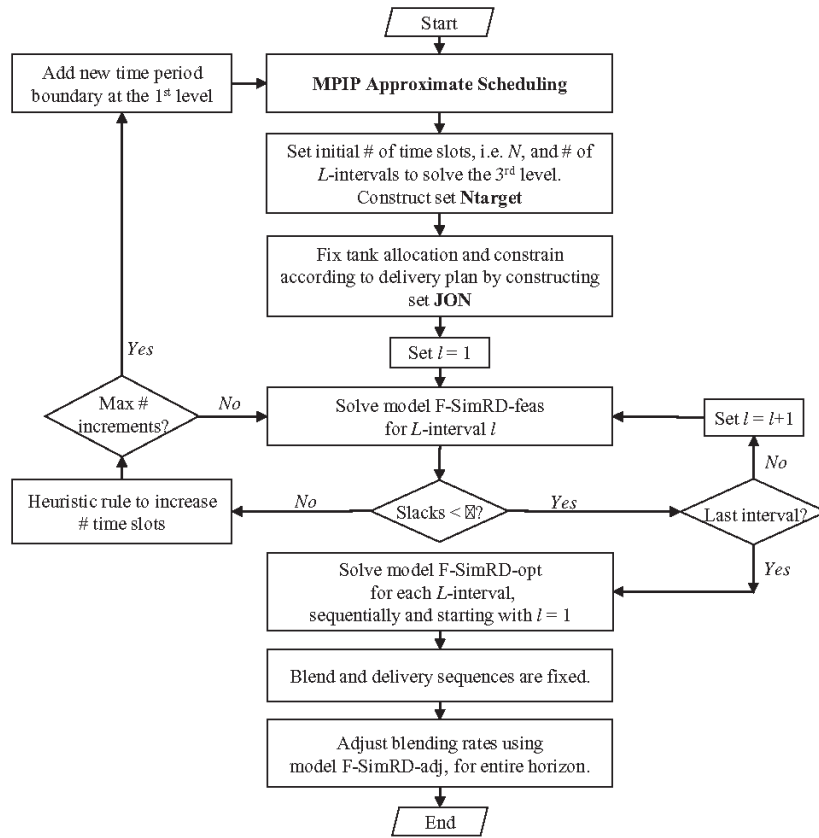


Fig. 7. MPIP-C scheduling algorithm flowchart.

horizon. Quality properties under specification are RON, Reid vapor pressure, sulfur content, specific gravity, aromatics content, olefin content, benzene content, oxygenates, and flammability limit. According to Li and Karimi (2011), they use the same addition bases and index correlations as Li et al. (2010); therefore, only sulfur content and oxygenates blend on a weight basis. However, Li and Karimi (2011) do not consider specific gravity in example 3, hence in this work sulfur content is assumed to blend linearly on a volumetric basis in example 3. Table 1 shows a description of the blending system for each example.

Test set #2 is composed by example number 4, 8, 12, and 14 from Li and Karimi (2011); however, RON and MON properties are considered to blend nonlinearly following the ethyl RT-70 models. RON index correlation from Li et al. (2010) was used to compute the actual RON values and product specifications. Li and Karimi (2011) do not specify MON values, and these were assumed in this

work. For simplicity, MON minimum product specifications were set equal to zero in order to observe only the effect of the RON constraint in the optimum. RON and MON values and specifications are shown in Table 2.

Fig. 8 shows the cumulative curves for example 12 and example 14. Examples 3, 4, 7, 8, 9, and 12 have a single inventory pinch point at time $T = 190$ h, while example 14 does not have an inventory pinch point. Therefore, examples 3–12 require two $L1$ -periods (boundary between them at 190 h mark), and example 14 only one $L1$ -period (entire scheduling horizon).

Table 3 shows the $L2$ -periods, their corresponding time slots, and the orders that can be delivered during each $L2$ -period. Note that for examples 3–12, the start of the last $L2$ -period matches the inventory pinch point location.

Table 1
Summary of the examples from Test Set #1 and #2 (Li and Karimi, 2011).

Example ID	# Blenders	# Orders	# Products	# Product tanks	# Quality properties (weight basis)
3	1	12	4	11	5
4	1	15	4	11	9(2)
7	1	20	4	11	9(2)
8	2	20	4	11	9(2)
9	2	23	5	11	9(2)
12	2	35	5	11	9(2)
14	3	45	5	11	9(2)

Table 2
RON and MON values for examples from Test Set #2.

Property	Blend components									
	C1	C2	C3	C4	C5	C6	C7	C8	C9	
RON	75	90.3	95.6	97.3	83	100	115	118	81	
MON	66	80.8	80.5	91.7	74	100	109	100	72	
Product specifications [min, max]										
	P1		P2		P3		P4		P5	
RON	[95,200]		[96,200]		[94,200]		[90,200]		[98,200]	
MON	[0,200]		[0,200]		[0,200]		[0,200]		[0,200]	

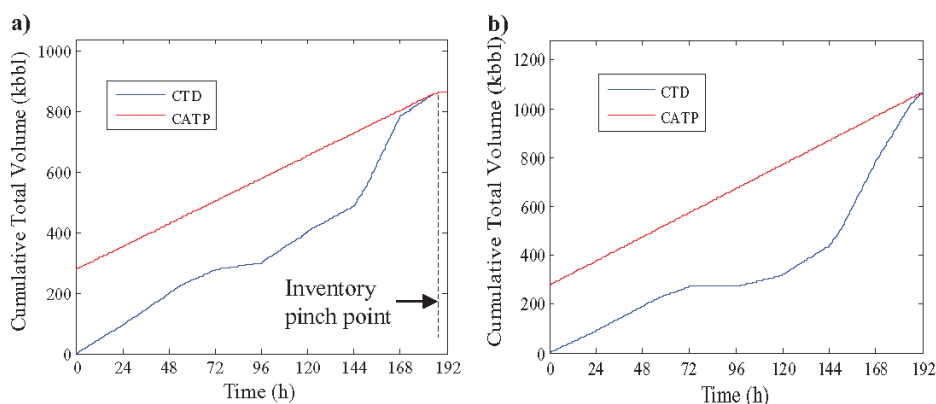


Fig. 8. Cumulative curves and inventory pinch points for (a) Ex. 12 and (b) Ex. 14.

5. Computational performance of MPIP-C algorithm

Castillo-Castillo and Mahalec (2015) showed that their continuous-time full-space blend scheduling model enables exact solutions of medium-size models when there are no penalties for multiple deliveries from the tanks. Global optimal solutions become much more difficult to compute when minimum number of multiple deliveries from the tanks is desired, and when using nonlinear blending equations. MPIP-C algorithm represents an alternative to compute good quality solutions for those cases and with short execution times. This section summarizes the results of MPIP-C algorithm.

5.1. Test set #1

Simultaneous receipt and delivery by product tanks are considered in this case. Table 4 shows the model size of the full-space model, and the models at each level of the MPIP-C algorithm, for examples from Test Set #1 when employing the data from Table 2. Since the examples have very similar cumulative curves, 1st level model size does not change significantly. 1st level model size increases with the number of products required to be blended and with the number of L1-periods. It can be seen that the 3rd level model has approximately half of the total number of binary variables required by the full-space model.

Table 3
L2-Periods, time slots, and orders that can be delivered in each period (Test Set #1 and #2).

Ex	L2-Period	Duration (h)	Slots	Orders that can be delivered
3, 4	1	100	1, 2	01–07, 012–015
	2	90	3–5	08–011
	3	2	6	–
7	1	80	1–5	01–07, 012–019
	2	70	6, 7	08
	3	40	8	08–011, 020
	4	2	9	–
8	1	80	1–3	01–07, 012–019
	2	70	4, 5	08
	3	40	6, 7	08–011, 020
	4	2	8	–
9	1	80	1, 2	01–07, 012–019
	2	70	3, 4	08, 021
	3	40	5, 6	08–011, 020, 022, 023
	4	2	7	–
12	1	24	1, 2	01–04, 013, 015, 019, 033
	2	26	3, 4	05–07, 012, 013, 015, 019, 033
	3	26	5, 6	014–018, 033
	4	24	7, 8	027, 028
	5	20	9, 10	021, 024, 029–032
	6	30	11, 12	08, 034, 035
	7	40	13	08–011, 020, 022, 023, 025, 026
	8	2	14	–
14	1	24	1, 2	01–04, 013, 015, 019, 026
	2	26	3, 4	05–07, 012, 013, 015, 019, 026
	3	26	5, 6	014–018, 026
	4	24	7, 8	–
	5	20	9, 10	021, 024, 045
	6	30	11, 12	08, 027–031
	7	42	13	08–011, 020, 022, 023, 025, 032–044

Table 4
Model size of the full-space model, and the models at each level of the MPIP-C algorithm, for Test Set #1 (linear examples).

Example ID	3	4	7	8	9	12	14
Full-space linear model L-SimRD-opt							
# Slots	6	6	9	8	7	14	13
# Eqs	6730	7756	15,252	15,725	13,958	27,007	33,309
# Cont	1946	2189	3978	3653	3263	7166	7889
# Bins	397	433	804	728	642	1194	1276
MPIP-C 1st level model (recipe optimization)							
# L1-periods	2	2	2	2	2	2	1
# Eqs	423	487	487	487	588	588	316
# Cont	185	185	185	185	224	224	120
MPIP-C 2nd level model (approximate schedule)							
# L2-periods	3	3	4	4	4	8	7
# Eqs	2305	2560	3870	4343	4880	12,648	13,151
# Cont	1155	1248	1800	2052	2366	5539	5790
# Bins	393	447	699	827	924	2511	2666
MPIP-C 3rd level model F-SimRD-opt (detailed schedule)							
# Slots	6	6	9	8	7	14	13
# Eqs	5391	5785	11,043	11,622	10,309	19,788	24,643
# Cont	1861	2027	3550	3379	3059	6794	7551
# Bins	164	178	372	378	316	529	698

#Eqs = number of equations, #Cont = continuous variables, #Bins = number of binary variables.

Table 5 presents the results for test set #1. For small problems (examples 3–9) MPIP-C computes optimal results, while for examples 12 and 14, the results have 0.13% and 0.09% optimality gaps, respectively. The optimality gap was computed by providing MPIP-C solution as a starting point to the full-space model; the

first reported lower bound by CPLEX corresponds to the MPIP-C solution. Execution times for small problems are of the same order of magnitude as those for the new version of the full-space MILP model (Castillo-Castillo and Mahalec, 2015). For large problems, the execution times for MPIP-C algorithm are significantly smaller.

Table 5
Full-space and MPIP-C results, SimRD scenario, not penalizing deliveries ($c_4 = 0$). Test Set #1 (linear examples).

Ex.	Algorithm	Obj. func. value ($\times 10^3$ \$)	Gap (%)	Blend cost	# BR	# TST	# DR	Total CPU time (s)
3	MPIP ^a	3179.1	0	3139.1	2	0	19	14.2
	Full-Space ^b	3179.1	0	3139.1	2	0	17	2.6
	MPIP-C	3179.1	0	3139.1	2	0	18	1.6
	FS-WS	3179.1	0	3139.1	2	0	18	2.4
4	MPIP ^a	4576.7	0	4536.7	2	0	23	16.2
	Full-Space ^b	4576.7	0	4536.7	2	0	20	2.3
	MPIP-C	4576.7	0	4536.7	2	0	21	1.7
	FS-WS	4576.7	0	4536.7	2	0	21	2.6
7	MPIP ^a	8120.3	0	8040.3	4	0	29	17.0
	Full-Space ^b	8120.3	0	8040.3	4	0	31	63.1
	MPIP-C	8120.3	0	8040.3	4	0	29	4.9
	FS-WS	8120.3	0	8040.3	4	0	29	2.8
8	MPIP ^a	8120.3	0	8040.3	4	0	28	17.3
	Full-Space ^b	8120.3	0	8040.3	4	0	28	37.8
	MPIP-C	8120.3	0	8040.3	4	0	26	5.9
	FS-WS	8120.3	0	8040.3	4	0	26	4.2
9	MPIP ^a	10,818.8	0	10,704.3	5	1	30	33.2
	Full-Space ^b	10,818.8	0	10,704.3	5	1	32	83.4
	MPIP-C	10,818.8	0	10,704.3	5	1	29	5.1
	FS-WS	10,818.8	0	10,704.3	5	1	29	4.2
12	MPIP ^a	15,281.7	0.13	15,147.2	6	1	42	129
	Full-Space ^b	15,281.7	0.13	15,147.2	6	1	49	43,200 ^c
	MPIP-C	15,281.7	0.13	15,147.2	6	1	46	11.9
	FS-WS	15,261.7	0	15,147.2	5	1	45	418
14	MPIP ^a	21,181.4	0.09	21,046.9	6	1	53	333
	Full-Space ^b	21,161.4	0	21,046.9	5	1	62	12,106
	MPIP-C	21,181.4	0.09	21,046.9	6	1	53	19.7
	FS-WS	21,161.4	0	21,046.9	5	1	53	1611

FS-WS = full-space model with warm start (MPIP-C solution as starting point).

#BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

^a MPIP scheduling algorithm solution from Castillo and Mahalec (2014b).

^b Full-space model solution from Castillo-Castillo and Mahalec (2015).

^c Reached maximum allocated time.

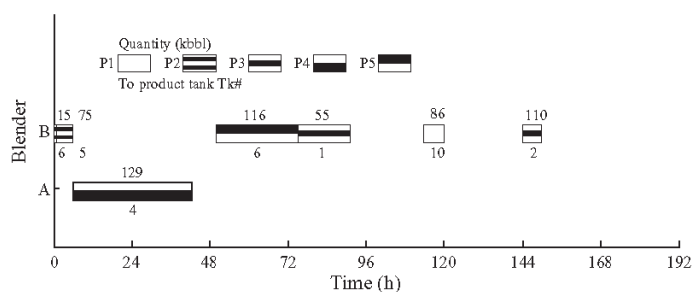


Fig. 9. Blend schedule for Ex. 12, Test Set #1, SimRD, MPIP-C solution.

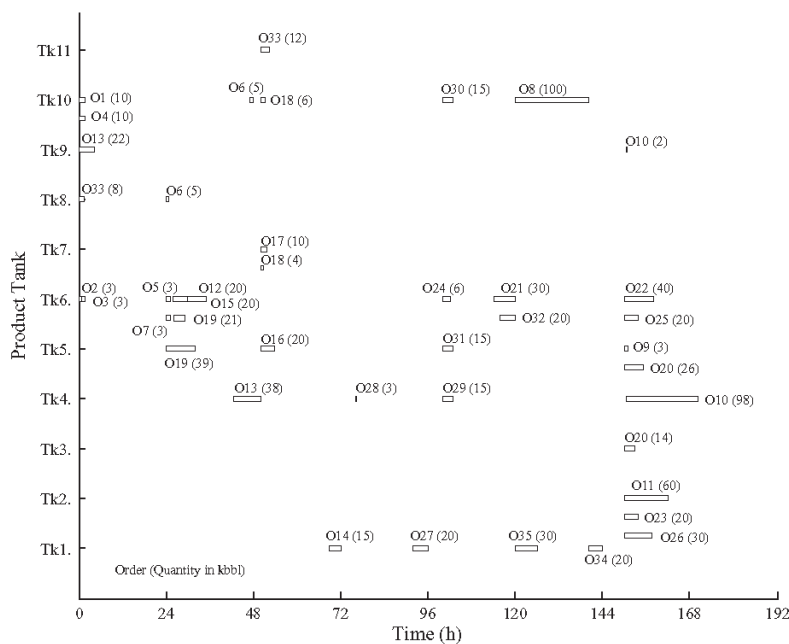


Fig. 10. Delivery schedule for Ex. 12, Test Set #1, SimRD, MPIP-C solution.

Shown in Table 5 are also the solutions which are obtained by providing MPIP-C result as a starting point to the full-space continuous-time model (row “FS-WS”). Using this procedure, example 12 was solved to optimality in 11.9s+418s=430s, while example 14 was solved to 0% optimality gap in

19.7s+1,611s=1631s. This is significantly faster than the full-space model without starting point.

Figs. 9 and 10 show the production and delivery schedule, respectively, computed by MPIP-C algorithm for example 12, while Figs. 11 and 12 show the corresponding

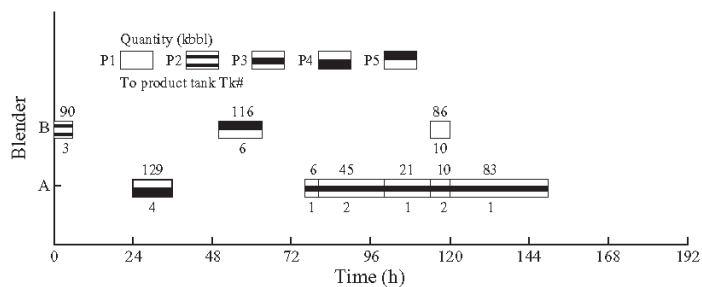


Fig. 11. Blend schedule for Ex. 12, Test Set #1, SimRD, FS-WS solution.

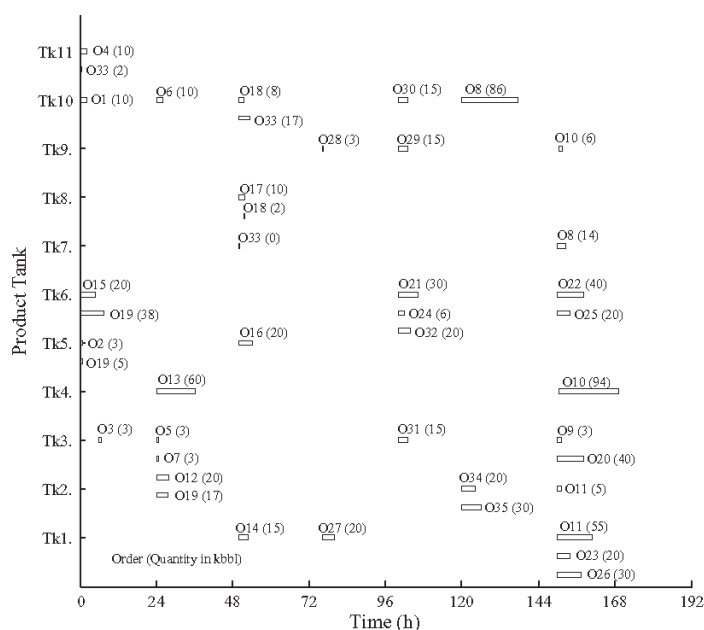


Fig. 12. Delivery schedule for Ex. 12, Test Set #1, SimRD, FS-WS solution.

schedules obtained by the FS-WS approach for the same example.

5.2. Test set #2

For these examples, it is assumed that simultaneous receipt and delivery by product tanks is not allowed. Table 6 shows the model size of the full-space model, and the models at each level of the MPIP-C algorithm, for examples from Test Set #2 when employing

Table 6
Model size of the full-space model, and the models at each level of the MPIP-C algorithm, for Test Set #2 (nonlinear examples).

Example ID	4	8	12	14
Full-space nonlinear model N-NoSimRD-opt				
# Slots	6	8	14	13
# Eqs	6169	12,297	22,699	27,729
# Cont	2028	3536	7095	8088
# Bins	433	728	1194	1276
# Nonlinear terms	156	416	728	1014
MPIP-C 1st level model (recipe optimization)				
# L1-periods	2	2	2	1
# Eqs	591	591	718	381
# Cont	273	273	334	175
# Nonlinear terms	64	64	80	40
MPIP-C 2nd level model (approximate schedule)				
# L2-periods	3	4	8	7
# Eqs	2560	4343	12,648	13,151
# Cont	1248	2052	5539	5790
# Bins	447	827	2511	2666
MPIP-C 3rd level model F-NoSimRD-opt (detailed schedule)				
# Slots	6	8	14	13
# Eqs	4876	9462	17,436	21,064
# Cont	1881	3137	6516	7238
# Bins	178	378	532	695

#Eqs = number of equations, #Cont = continuous variables, #Bins = number of binary variables.

the data from Table 2. It can be seen that the number of binary variables of the full-space model is the same for linear and nonlinear models (see Table 4) since the difference is only the addition of equations and continuous variables for the octane blending model. Compared with the linear cases, the nonlinear 1st level model has more equations and continuous variables; however, the 2nd level model has the same size in both cases (this is expected since the 2nd level model only depends on the number of L2-periods). The size of the 3rd level model depends on the solution from the 2nd level; therefore, it is not significantly affected by using a linear or nonlinear blending model. Once again, the 3rd level model needs around 50% of the total number of binary variables required by the full-space model.

The stopping criteria for Test Set #2 examples is 0.01% optimality gap or the maximum allocated time. Table 7 shows the results from the full-space MINLP model, MPIP-C algorithm, and the full-space MINLP model with MPIP-C solution as starting point. ANTIGONE is used to solve the 1st level model of the MPIP-C algorithm. Although the full-space model computes solutions with small optimality gaps, it may require more than 900 s to find solutions with gaps smaller than 5%. For the large-scale problem (example 14), MPIP-C algorithm computes a solution within 1% of the optimum in 24 s, while ANTIGONE and BARON require 1205 s and 3634 s, respectively, to compute a solution within 5% of the optimum. When using the MPIP-C solution as starting point for the full-space model, only examples 4 and 8 (small problems) can be solved to proven optimality within the allocated time.

Table 8 shows the execution times at each level of the MPIP-C algorithm for examples 4, 8, 12, and 14, for both linear and nonlinear cases. It can be seen that the 2nd and 3rd level execution times are very similar in magnitude (2nd and 3rd levels are always linear), while the 1st level execution times increased at least 6 times when the nonlinear blending equations are employed. This suggests that using more complex nonlinear blending models will only affect the 1st level execution times, since the 2nd and 3rd level use fixed recipes.

Table 7Full-space and MPIP-C results, NoSimRD scenario, not penalizing deliveries ($c_d = 0$). Test Set #2 (nonlinear examples).

Ex	Algorithm	MINLP solver	Obj. func. value ($\times 10^3$ \$)	Gap (%)	CPU time (s) to reach:			Blend cost ($\times 10^3$ \$)	# BR	# TST	# DR
					Stop criteria	Final upper bound	Opt. gap < 5%				
4	Full-Space ^a	ANTIGONE	4633.0	0.01	856	856	141	4593.0	2	0	21
	Full-Space ^a	BARON	4633.0	0.01	15	15	11	4593.0	2	0	21
	MPIP-C	–	4632.7	–	3	3	–	4592.7	2	0	22
	FS-WS	ANTIGONE	4632.7	0.01	121	0	0	4592.7	2	0	22
	FS-WS	BARON	4632.7	0.01	7	0	0	4592.7	2	0	22
8	Full-Space ^a	ANTIGONE	8205.4	0.03	10,800 ^b	9246	380	8125.4	4	0	28
	Full-Space ^a	BARON	8207.4	0.05	10,800 ^b	1388	79	8127.4	4	0	26
	MPIP-C	–	8203.1	–	6	6	–	8123.1	4	0	28
	FS-WS	ANTIGONE	8203.1	0.01	1209	0	0	8123.1	4	0	28
	FS-WS	BARON	8203.1	0.01	16	0	0	8123.1	4	0	28
12	Full-Space ^a	ANTIGONE	15,406.8	0.16	43,200 ^b	43,054	907	15,272.3	6	1	42
	Full-Space ^a	BARON	15,453.1	0.45	43,200 ^b	7933	7933	15,318.6	6	1	45
	MPIP-C	–	15,402.6	–	17	17	–	15,268.1	6	1	44
	FS-WS	ANTIGONE	15,402.6	0.13	43,200 ^b	0	0	15,268.1	6	1	44
	FS-WS	BARON	15,402.6	0.13	43,200 ^b	0	0	15,268.1	6	1	44
14	Full-Space ^a	ANTIGONE	21,283.1	0.19	43,200 ^b	12,839	1205	21,128.6	7	1	56
	Full-Space ^a	BARON	21,497.7	1.20	43,200 ^b	7204	3634	21,383.2	5	1	56
	MPIP-C	–	21,263.1	–	24	24	–	21,128.6	6	1	55
	FS-WS	ANTIGONE	21,263.1	0.09	43,200 ^b	0	0	21,128.6	6	1	55
	FS-WS	BARON	21,263.1	0.09	43,200 ^b	0	0	21,128.6	6	1	55

FS-WS = full-space model with warm start (MPIP-C solution as starting point).

#BR = number of blend runs, #TST = number of transitions by swing tanks, #DR = number of delivery runs.

^a Full-space model solution from Castillo-Castillo and Mahalec (2015).^b Reached maximum allocated time.**Table 8**

Comparison of the execution times at each level of the MPIP-C algorithm.

MPIP-C level	CPU time (s)		
	1st level	2nd level	3rd level
Solver	CPLEX	CPLEX	CPLEX
Test Set #1 (linear)			
Ex. 4	0.123	0.384	1.193
Ex. 8	0.185	0.592	5.169
Ex. 12	0.200	3.612	8.088
Ex. 14	0.145	4.912	14.658
MPIP-C level	CPU time (s)		
	1st level	2nd level	3rd level
Solver	ANTIGONE	CPLEX	CPLEX
Test Set #2 (nonlinear)			
Ex. 4	0.766	0.268	1.222
Ex. 8	0.890	0.387	4.475
Ex. 12	2.652	3.460	10.125
Ex. 14	1.605	5.423	16.513

6. Conclusions

In this paper we have presented Multi-Period Inventory Pinch scheduling algorithm with Continuous-time model used for detailed scheduling (MPIP-C scheduling algorithm). The algorithm uses inventory-pinch concept to decouple optimization of the blend recipes from the combinatorial aspects of scheduling. It also combines the strengths of discrete-time models (what to produce, in which time period, and how much to produce, where to store, where from to deliver) with continuous-time model capabilities to compute accurately when to start or to stop an operation.

Multi-period inventory pinch, an essential feature of MPIP-C algorithm, enables computation of approximate schedule by iterations between the top two levels. The 1st level optimizes a linear or nonlinear blending model (LP or NLP, respectively) and the 2nd level optimizes approximate scheduling model (MILP) with fixed recipes from the 1st level.

For small size scheduling problems and linear blending rules, MPIP-C algorithm requires execution times of the same magnitude as the reduced-size full-space continuous-time model (Castillo-Castillo and Mahalec, 2015). However, as the problem size grows, decomposition employed by MPIP-C enables computation of optimal or near optimal solutions within very short amounts of time. In addition, the execution times for the full-space model grow very rapidly. Rapid convergence of MPIP-C algorithm makes possible to use its solution as a starting point for a full-space model. For medium size problems such approach leads to an optimal solution within 1 h or less. However, for large problems the global optimum still cannot be computed within 12 h. Execution times for MPIP-C are determined by the speed of solving the nonlinear problem at the top level. If more elaborate nonlinear models are used at the 1st level, they will only increase the computational times at the 1st level but not at the 2nd and 3rd levels.

Due to the structure of MPIP-C algorithm, we expect that even larger problems can be solved efficiently. This is possible as long as the approximate schedule (2nd level) is computed very rapidly, since the approximate scheduling solution can be used to partition the time horizon at the 3rd level into smaller intervals. If these intervals are small, their continuous-time models can be solved very rapidly. Even though MPIP-C algorithm is heuristic in nature, most of the solutions obtained are at least as good as the results obtained by full-space models over much longer execution times.

Acknowledgments

Support by Ontario Research Foundation and McMaster Advanced Control Consortium is gratefully acknowledged.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at <http://dx.doi.org/10.1016/j.compchemeng.2015.08.005>.

References

- Bassett MH, Pekny JF, Reklaitis GV. Decomposition techniques for the solution of large-scale scheduling problems. *AIChE J* 1996;42(12):3373–87.
- Birewar DB, Grossmann IE. Simultaneous production planning and scheduling in multiproduct batch plants. *Ind Eng Chem Res* 1990;29:570–80.
- Castillo PA, Kelly JD, Mahalec V. Inventory pinch algorithm for gasoline blend planning. *AIChE J* 2013;59(10):3748–66.
- Castillo PA, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling – part I: gasoline blend planning. *AIChE J* 2014a;60(6):2158–78.
- Castillo PA, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling – part II: gasoline blend scheduling. *AIChE J* 2014b;60(7):2475–97.
- Castillo-Castillo PA, Mahalec V. Improved continuous-time model for gasoline blend scheduling. *Comput Chem Eng* 2016;84:627–46.
- Chen X, Wang N. Optimization of short-time gasoline blending scheduling problem with a DNA based hybrid genetic algorithm. *Chem Eng Process* 2010;49:1076–83.
- Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. *Comput Chem Eng* 2001;25:627–34.
- Gothe-Lundgren M, Lundgren JT, Persson JA. An optimization model for refinery production scheduling. *Int J Prod Econ* 2002;78:255–70.
- Healey WC, Maasen CW, Peterson RT. A new approach to blending octanes. In: *Proc. 24th meeting of American Petroleum Institute's Division of Refining*; 1959.
- Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending. *Ind Eng Chem Res* 2003;42:825–35.
- Jia Z, Ierapetritou M. Efficient short-term scheduling of refinery operations based on a continuous time formulation. *Comput Chem Eng* 2004;28:1001–19.
- Kolodziej SP, Grossmann IE, Furman KC, Sawayac NW. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. *Comput Chem Eng* 2013;53:122–42.
- Li J, Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res* 2011;50:9156–74.
- Li J, Karimi IA, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. *AIChE J* 2010;56:441–65.
- Li J, Misener R, Floudas CA. Continuous-time modeling and global optimization approach for scheduling of crude oil operations. *AIChE J* 2012;58:205–26.
- Mendez CA, Grossmann IE, Harjunkoski I, Kabore P. A simultaneous optimization approach for off-line blending. *Comput Chem Eng* 2006;30:614–34.
- Pinto JM, Joly M, Moro LFL. Planning and scheduling models for refinery operations. *Comput Chem Eng* 2000;24:2259–76.
- Shah NK, Ierapetritou MG. Short-term scheduling of a large-scale oil-refinery operations: incorporating logistic details. *AIChE J* 2011;57:1570–84.
- Singh A, Forbes JF, Vermeer PJ, Woo SS. Model-based real-time optimization of automotive gasoline blending operations. *J Process Control* 2000;10:43–58.
- Terrazas-Moreno S, Grossmann IE. A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chem Eng Sci* 2011;66:4307–18.
- Wolpert DH, Macready WG. No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1997;1(1):67–82.
- Wu D, Ierapetritou MG. Decomposition approaches for the efficient solution of short-term scheduling problems. *Comput Chem Eng* 2003;27:1261–76.
- Zhao J, Wang N. A bio-inspired algorithm based on membrane computing and its application to gasoline blending scheduling. *Comput Chem Eng* 2011;35:272–83.

Chapter 7: Global Optimization Algorithm for Large-Scale Refinery Planning Models with Bilinear Terms

This chapter has been published in the *Industrial & Engineering Chemistry Research Journal*. Complete citation:

Castillo Castillo, P. A., Castro, P. M., & Mahalec, V. (2017). Global optimization algorithm for large-scale refinery planning models with bilinear terms. *Industrial & Engineering Chemistry Research*, 56(2), 530–548. *American Chemical Society*, doi: 10.1021/acs.iecr.6b01350

Permission from © American Chemical Society.

Chapters 2–6 show the development steps of a heuristic algorithm (i.e., MPIP-C). In Chapter 7, a deterministic global optimization algorithm for mixed-integer bilinear programs is presented. This method computes estimates of the global solution by solving an MILP relaxation of the original model. The relaxation is derived using either Piecewise McCormick or Normalized Multiparametric Disaggregation. By increasing the number of partitions, and reducing the domain of the variables, the estimates of the global solution are improved. The case study used in Chapter 6 is an oil refinery planning problem.

Global Optimization Algorithm for Large-Scale Refinery Planning Models with Bilinear Terms

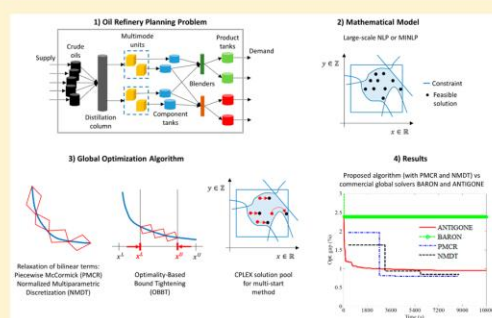
Pedro Castillo Castillo,[†] Pedro M. Castro,^{*,‡,§} and Vladimir Mahalec[†]

[†]Department of Chemical Engineering, McMaster University, Hamilton, ON L8S 4A7, Canada

[‡]Centro de Matemática Aplicações Fundamentais e Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal

S Supporting Information

ABSTRACT: We propose a global optimization algorithm for mixed-integer nonlinear programming (MINLP) problems arising from oil refinery planning. It relies on tight mixed-integer linear programming (MILP) relaxations that discretize the bilinear terms dynamically using either piecewise McCormick (PMCR) or normalized multiparametric disaggregation (NMDT). Tight relaxations help finding a feasible solution of the original problem via a local nonlinear solver, with the novelty being the generation of multiple starting points from CPLEX's solution pool and the parallel execution. We show that optimality-based bound tightening (OBBT) is essential for large-scale problems, even though it is computationally expensive. To reduce execution times, OBBT is implemented in parallel. The results for a refinery case study, featuring units with alternative operating modes, intermediate storage tanks, and single- and multiple-period supply and demand scenarios, show that the algorithm's performance is comparable to commercial solvers BARON and ANTIGONE.



1. INTRODUCTION

A petroleum refinery plant transforms the raw crude oils into valuable products that are found almost anywhere in our everyday life such as liquid fuels, plastics, oils, solvents, asphalt, and many chemicals used in production processes across different type of industries. As a brief description, a petroleum refinery separates the crude oil into several fractions with different quality properties (e.g., density, sulfur content, octane number, etc.). These fractions are subsequently converted into intermediate products through various reaction and/or separation processes, and finally, intermediates are sold, sent to another chemical plant (e.g., production of polyethylene, polystyrene), or used as blend components for liquid fuels (e.g., gasoline, kerosene, diesel).

Oil refineries are composed of several intertwined complex physical and chemical processes. For simplicity, they are commonly divided into three main sections:^{1,2} crude oil unloading and mixing section, processing units, and the blending and shipping of final products. Refineries operate in a highly competitive and dynamic market where significant variations can occur in product demand and crude oil prices, and their profit margins are relatively small compared to other industries (e.g., pharmaceutical). In addition, they must operate under strict safety, environmental, and governmental regulations. For these reasons, production planning is an important and widespread tool in oil refineries. One of the main issues regarding production planning of an oil refinery is the nonlinear nature of the processes involved. Although these nonlinearities

can be avoided by assuming the processes to be linear, inaccuracy of the resulting model can make the solution unsuitable for real life applications (i.e., infeasible) or suboptimal. Commercial software based on linear programming (LP) or successive linear programming (SLP) techniques have allowed computation of production plans for large-scale refinery plants; however, global optimal solutions cannot be guaranteed with such strategies. Several production planning models and optimization algorithms have been developed by researchers in this area, focusing on increasing the accuracy of the planning models and/or the performance of the solution algorithms. Bengtsson and Nonas³ and Shah et al.⁴ present an overview of the improvements and challenges found on the different planning levels (i.e., strategic, tactical, and operational) of oil refinery operations.

The current trend in production planning is to use nonlinear programming models to capture the nature of the processes involved. Moro et al.⁵ represented a processing unit employing a general nonlinear model with the following variables: feed flow rate, feed properties, unit operating conditions, product flow rates, and product properties. They tested their nonlinear approach using a diesel plant from Petrobras RPBC as an example, and they computed a better solution than the

Received: April 8, 2016

Revised: December 6, 2016

Accepted: December 16, 2016

Published: December 16, 2016

then-current situation (plan made on the basis of prior experience and manual calculations). Neiro and Pinto⁶ presented a general modeling framework for a refinery system by using the general unit model from Moro et al.⁵ and adding mathematical models for storage tanks and pipelines. This framework was based on a discrete-time, mixed-integer nonlinear programming (MINLP) formulation. Li et al.⁷ developed and integrated nonlinear models for the crude distillation unit (CDU) and the fluidize-bed catalytic cracker (FCC) into the refinery planning problem. For the CDU, they employed a swing cut method where the size of the swing cuts were determined using true boiling point (TBP) data of the feed and the cut-points of the different CDU operating modes. For the FCC, the product fractions were correlated to the conversion of the FCC. In both units, the quality properties are calculated using nonlinear correlations. Alhajri et al.⁸ proposed a swing cut method for the CDU based on a polynomial formulation, and fractional polynomials for the processing unit models. Elkamel et al.⁹ extended their work to optimize the production plan of a refinery while reducing the carbon dioxide emissions. Alattas et al.¹⁰ presented a multiple-period MINLP model for the refinery planning problem using a nonlinear model for the CDU based on the fractionation index of a distillation column. In contrast to the swing cut methods, the fractionation index model incorporates thermodynamic principles such as relative volatility and phase equilibrium. Zhang et al.¹¹ developed a multiple-period MINLP model that aims to optimize the production plan of a refinery site accounting for the material and energy requirements. Nonlinearities appear in the pour point blending equations.

Some linear programs have been published as well, especially for problems incorporating uncertainty or integrating scheduling decisions. Pongsakdi et al.¹² described a linear program for the refinery planning problem by employing a fixed-yield approach for all units, and by fixing the quality properties to average values obtained from plant data or constraining them with linear inequalities. This LP model was used as the basis to develop a stochastic model that considers uncertainty in the demand forecasts. Kuo and Chang¹³ presented a multiple-period mixed-integer linear programming (MILP) model for the short-term planning of the petroleum supply chain. The model considered crude-oil procurement activities, refinery sites, and distribution centers. However, quality properties were assumed to be fixed because no quality blending equations were included.

Given that linear models are not accurate enough to represent the processing units in the oil refineries, the current trend to increase the use and complexity of the nonlinear models is expected to continue. However, one of the main issues is that these nonlinear models are nonconvex; therefore, traditional convex optimization techniques are not suitable if the global optimum is required.

1.1. Global Optimization of Mixed-Integer Nonlinear Problems. Nonlinear programming (NLP) models are usually handled with commercial local nonlinear solvers such as CONOPT, MINOS, and KNITRO. These type of solvers are gradient-based, which means that they can stop at a local optimum depending on the starting point. Convex MINLP models can be solved to global optimality with algorithms based on branch-and-bound strategies, outer-approximation, generalized benders decomposition, and extended cutting plane methods; and with commercial solvers like DICOPT,

ALPHA-ECP, and SBB. A review on MINLP solution strategies can be found in D'Ambrosio and Lodi.¹⁴

The need for global optimal solutions has increased the use of stochastic and deterministic global optimization strategies.

Stochastic global optimization methods usually explore the feasible region of the problem using a population of solutions. On the basis of the global and/or local information retrieved by the current and previous populations, and according to the rules of the specific algorithm, new elements of the population are created to replace some of the existing solutions. The idea is that after several major iterations, the population will move toward the global optimal solution. Some examples of these methods are genetic algorithms, particle swarm optimization, differential evolution, memetic algorithms, ant colony optimization, and scatter search. The main disadvantage of stochastic global algorithms is that the quality of the final solution after a finite number of iterations, or finite amount of time, is not known; i.e., there is no guarantee that a global optimum has been found. Several reviews and surveys on stochastic global optimization methods can be found in the literature.

Deterministic global optimization algorithms require to compute not only good feasible solutions but also estimates of the best possible solution of the problem. The latter is obtained by following the construction of a convex relaxation of the original nonconvex problem. The solution of such convex model represents an estimate of the best possible solution of the original nonconvex problem (i.e., the global optimum). The main idea in global optimization algorithms is to reduce the difference between the best feasible solution and the estimate of the best possible solution to a nonzero tolerance ϵ . To compute better estimates of the global optimum, the quality (tightness) of the relaxed model is usually increased by employing a spatial branch-and-bound framework, by adding cuts, by reducing the domain of the variables, and/or by increasing the number of partitions in the case of piecewise relaxation techniques. Feasible solutions are commonly computed using information from the convex model solution. Floudas and Gounaris¹⁵ present an overview of the recent developments in deterministic global optimization techniques.

Convex relaxations are obtained by replacing the nonlinear terms and/or equations in the original model with corresponding convex underestimators (and overestimators). Convex relaxations can be obtained using the re-formulation-linearization technique (RLT)^{16,17} or α BB convex underestimators.^{18,19} The tightest convex under and overestimators are called convex envelopes, and they have been formulated for some specific nonlinear terms. For instance, in the case of bilinear terms, i.e., x_1x_2 , the set of linear constraints known as the McCormick envelopes,²⁰ provide the tightest relaxation.^{21,22}

Because the maximum difference between the original bilinear term x_1x_2 and its corresponding McCormick envelopes is proportional to the area of the domain,¹⁸ i.e., $(x_1^U - x_1^L)(x_2^U - x_2^L)$, bound tightening techniques (also known as domain reduction techniques) applied to the variables involved in the bilinear terms can improve the tightness of the McCormick envelopes. Another option to improve the relaxation is to discretize one of the variables from the bilinear term into NP partitions and, for each partition, generate the associated McCormick envelopes. This approach is known as piecewise McCormick relaxation (PMCR). Because this method requires the introduction of binary variables, there is a trade-off between the quality of the

relaxation (better quality = more partitions = more binary variables) and the computation effort required to compute it.

Given the maturity of MILP solvers, the current trend for solving nonconvex NLP and MINLP problems with only bilinear and quadratic terms is to employ piecewise linear relaxations. Figure 1 shows the shape of the bilinear term

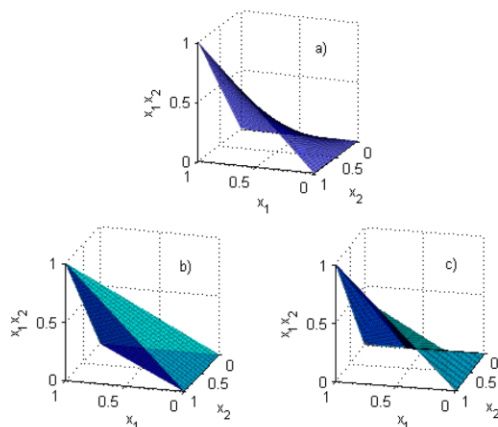


Figure 1. (a) Nonconvex function x_1x_2 , (b) corresponding McCormick envelopes, and (c) piecewise McCormick relaxation with three partitions.

x_1x_2 in the $[0, 1]^2$ domain, the corresponding McCormick envelopes, and a uniform, univariate, piecewise McCormick relaxation.

Andrade et al.²³ presented a multiple-period NLP model that computes the flows and quality properties of all streams in a refinery network, the tank inventories, and the operating mode of the units, to maximize the profit. Bilinear terms appear in the property balance equations to explicitly determine the value of the quality properties of the streams. To solve this problem, they relaxed the bilinear terms using standard McCormick envelopes, thus deriving a LP model whose solution is an upper bound of the problem. Several initial points for a local nonlinear solver were generated by perturbing the LP solution. After a predefined number of iterations, or if the optimality gap was less than the tolerance, the algorithm stopped. However, because the relaxed model is solved only once at the beginning of their algorithm, the optimality gap tolerance might not be met. In spite of this, the authors pointed out the importance of computing good feasible solutions relatively fast, and having at least one estimate of the global optimum.

Faria and Bagajewicz²⁴ developed a global optimization algorithm based on a piecewise linear relaxation of the nonlinear model that can contract the bounds of variables involved in bilinear terms and concave functions, individually and/or simultaneously, with several variants. The general idea to tighten the bounds of different variables at the same time is to forbid the specific partitions where the current MILP solution is located (adjacent partitions could be forbidden too), and then resolve the MILP model. If the new solution is worse than the previous one (i.e., the global optimum is in the forbidden region), then the bounds of the variables are contracted to those defined by the forbidden intervals, the new domain is repartitioned and the procedure is repeated. When the bounds can no longer be contracted, the number of partitions can be increased or the algorithm can switch

to a branch-and-bound strategy. One of the novelties of their method is that their bound contraction technique can be applied to variables not being partitioned by the piecewise relaxation scheme. They compare different partitioning strategies for the relaxation of bilinear and concave functions and apply their algorithm to water management and pooling problems.

Castro²⁵ proposed a bound tightening method for the non-partitioned variables in a piecewise McCormick relaxation. The idea is to determine the upper and lower bounds of the nonpartitioned variables for each partition by solving several instances of an LP model derived from the convex relaxation of the original NLP problem (one for each bound of a variable). In general, results show that this method computes smaller optimality gaps than conventional piecewise McCormick relaxation and without too much extra computational effort with a small number of partitions.

Misener et al.²⁶ developed a piecewise linear relaxation of the bilinear terms where the number of binary variables increases logarithmically with the number of partitions. In this formulation, if the number of partitions is NP, then the number of binary variables required by the relaxation method is $\log_2(NP)$. They integrated this relaxation scheme into a global optimization algorithm and compared its performance with a MILP formulation derived from the disjunctive program representing the piecewise McCormick relaxation. They applied their algorithm to several types and instances of the pooling problem. They concluded that the linear relaxation scheme is better for less than eight partitions, and that the logarithmic scheme is more advantageous for a larger number of partitions.

Kolodziej et al.²⁷ presented the derivation and formulation of the multiparametric disaggregation technique (MDT) for bilinear terms, a piecewise linear relaxation where the number of binary variables increases logarithmically with the number of partitions by $\log_{10}(NP)$. Kolodziej et al.²⁸ applied MDT to the gasoline blend scheduling problem and showed that this discretization technique can use different numerical bases (not only 10). Castro and Grossmann²⁹ presented an optimization-based bound tightening (OBBT) method for the variables involved in bilinear terms that relied on the MDT rather than the standard McCormick relaxation.

With MDT, the number of partitions depends not only on the discretization resolution but also on the domain of the variables (i.e., x^L and x^U). This is not desirable in global optimization algorithms because the domain of the variables is usually reduced with optimization-based tightening methods, feasibility-based tightening methods, and/or spatial branch-and-bound (B&B) strategies. To always discretize the interval $[0, 1]$, Castro³⁰ developed the normalized multiparametric disaggregation technique (NMDT). For the same discretization level, the number of partitions is the same for every discretized variable even if their domain is different. NMDT thus becomes conceptually similar to the univariate and uniform piecewise McCormick relaxation, while having the advantage of being more efficiently computationally for 10 or more partitions.

1.2. Main Features of New Global Optimization

Algorithm. We now propose a global optimization algorithm for MINLP problems with bilinear terms restricted to the product of continuous variables. The main element is to rely on a MILP relaxation from either PMCR or NMDT. Starting with a small number of partitions for each discretized variable, typically 4 for PMCR and 10 for NMDT, the MILP relaxation is solved by commercial solver CPLEX to compute the lower

bound, taking advantage of its solution pool feature to generate a few good initialization points. Multiple instances of a NLP version of the original MINLP (derived after fixing the binary variables to the values from the MILP solutions) are solved in parallel by a fast local solver, to maximize the likelihood of finding the global optimal solution. Every time a new upper bound is found, optimality-based bound tightening featuring the standard McCormick relaxation is performed to further reduce the variables' domain. This stage also takes advantage of the GAMS parallel and distributed grid computing option to minimize the clock wall time, at the expense of weaker bounds compared to those that would be obtained after going through the variables sequentially. GAMS is a modeling system for developing and solving optimization problems using mathematical programming techniques.

The iterations of the algorithm involve small increments in the number of partitions per variable. The goal is to be able to improve the quality of the relaxation without using too much time, so that low optimality gaps can be computed. The algorithm terminates when time runs out or the gap becomes lower than the given tolerance. Compared to commercial global optimization solvers BARON³¹ and ANTIGONE,³² the new algorithm explores piecewise relaxation and optimality-based bound tightening methods to a greater extent (more partitions per variable with all bilinearly appearing variables subject to OBBT), which can be viewed as an alternative to spatial B&B.

The rest of the paper is organized as follows: the refinery planning problem is defined in the next section and the associated MINLP planning model is presented in section 3. Subsequently, the mathematical models for the subproblems of the global optimization algorithm are described in section 4. The proposed algorithm is delineated in section 5, with the remaining sections containing the data used in our example problems, the numerical results, and the conclusions.

2. PROBLEM DEFINITION

2.1. Refinery Planning Problem. The refinery planning problem that we address in this paper is stated as follows:

Given

A planning horizon $[0, H]$ discretized in time periods n of known length L_n .

A set of crude oils, their quality properties (e.g., crude assay), and supply profiles.

A set of finished products (e.g., gasoline, diesel, kerosene), their quality specifications, and demand along the planning horizon.

A set of storage tanks and their maximum, minimum, and initial inventories.

A set of blenders and their minimum and maximum blending rates.

Processing units (i.e., crude distillation unit, hydrotreaters, catalytic reformer, etc.) with minimum and maximum production rates and their operating modes.

The interconnections between all the processing units and storage tanks (i.e., the refinery network is fixed).

Determine

The production volumes in each unit, each mode, and each time period.

The storage tank inventories at the boundaries of the time periods.

The volume blended of finished products in each blender and time period.

While

Minimizing the total cost (cost of raw materials plus operating cost), if all the product demand is fixed, or

Maximizing the profit (sales revenue minus total cost), if at least one product demand is not fixed.

General assumptions

Quality properties of crude oils, alkylate, and *n*-butane streams arriving to the refinery, are known and fixed.

Each crude oil has a dedicated storage tank.

Each product grade has a dedicated storage tank.

Initial product inventories are within quality specifications.

Tanks can receive and deliver material during the same time period.

Perfect mixing occurs in the blenders.

In the blenders, quality properties of the products are computed using weighted-averages, on either a volumetric or weight basis.

Time periods can represent days, weeks, or months.

2.2. Specific Assumptions. We present a general formulation for the output flows and output qualities of the units. Each unit can have its specific model. The ones that we employ in this work have the following assumptions:

The crude distillation unit employs a fixed yield method and the qualities of the distillation cuts are computed using linear blending equations.

For the other processing units, the yield and output qualities (except specific gravity and sulfur content) are fixed.

Minimum and/or maximum constraints for the quality properties of the inlet streams to the units are not enforced.

Because time periods can represent days, weeks, or months, we do not force a single product in each blender per time period, nor a single operating mode in a unit with multiple modes. A scheduling model will thus be required to address the feasibility of the production plan but it is beyond the scope of this paper.

3. REFINERY PLANNING MODEL

The model is based on a discrete-time formulation and can be classified as a nonconvex, mixed-integer nonlinear model, due to the presence of bilinear terms and binary variables. The notation used in this section is presented at the end of the paper; however, we describe next some of the most important sets, indices, parameters, and variables in the model. Note that an index takes all its possible values unless otherwise indicated. The quality balance equations have been written in a way that avoids trilinear terms.

Time periods are represented by set $\mathbf{N} = \{n\}$, streams are represented by set $\mathbf{S} = \{s\}$, blenders by $\mathbf{B} = \{b\}$, storage tanks by $\mathbf{T} = \{t\}$, all other units by $\mathbf{U} = \{u\}$, and the quality properties by $\mathbf{Q} = \{q\}$. Set \mathbf{S} (streams) not only includes the physical streams present in the actual refinery network but also can represent individual blend components and products. Set \mathbf{U} includes splitters, mixers, distillation columns, separators, and reactors. Set \mathbf{U} does not include blenders or storage tanks. Source and demand nodes are considered as subsets of \mathbf{S} . Inlet streams of units, tanks, and blenders are represented by sets $\mathbf{UI} = \{(u, s)\}$, $\mathbf{TI} = \{(t, s)\}$, and $\mathbf{BI} = \{(b, s)\}$, respectively. In the same way, outlet streams are represented by sets \mathbf{UO} , \mathbf{TO} , and \mathbf{BO} . Because distinct materials flow through specific streams, it is not necessary to track all quality properties in all streams. For example, if a stream goes to the gasoline pool, then it is not necessary to compute cetane index for that stream.

Set $\mathbf{SQ} = \{(s, q)\}$ indicates which quality properties are tracked in each stream.

The length of a time period n is defined by parameter L_n . Variable $VS_{s,n}$ is the volumetric flow through stream s during time period n . Variable $VU_{u,n}$ is the volumetric flow fed to unit u during n . The value of quality property q in stream s during period n is represented by variable $QS_{s,q,n}$. The inventory level of tank t at the end of period n is denoted by variable $VT_{t,n}$. Figure 2 shows the unit representation employed in this work.

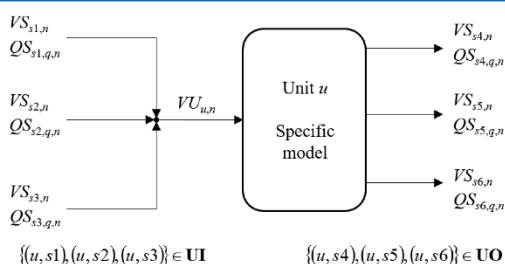


Figure 2. Unit representation used in this work.

The specific models employed in each unit can be found in the Supporting Information.

3.1. Objective Function. When product demand is fixed, the objective is to minimize the total cost (eq 1), which is defined in eq 2. When product demand is not fixed, eq 1 should be $\min Z = -\text{profit}$. Equation 3 is used to compute the profit, whereas eq 4 calculates the sales revenue. Equations 5–9 calculate the other cost terms.

$$\min Z = \text{Total cost} \quad (1)$$

$$\begin{aligned} \text{total cost} = & \text{materials cost} + \text{HT cost} + \text{PU cost} \\ & + \text{CDU cost} + \text{inventory cost} \end{aligned} \quad (2)$$

$$\text{profit} = \text{revenue} - \text{total cost} \quad (3)$$

$$\text{revenue} = \sum_n \sum_{s \in \text{DST}} \text{PS}_s \text{VS}_{s,n} \quad (4)$$

$$\text{materials cost} = \sum_n \sum_{s \in \text{SRC}} \text{CS}_s \text{VS}_{s,n} \quad (5)$$

$$\text{PU cost} = \sum_n \sum_{u \in \text{PU}} \text{CU}_u \text{VU}_{u,n} \quad (6)$$

$$\text{CDU cost} = \sum_n \sum_{u \in \text{CDU}} \text{CU}_u \text{VU}_{u,n} \quad (7)$$

$$\text{HT cost} = \sum_n \sum_{u \in \text{HTU}} \text{CU}_u \text{RSUL}_{u,n} \quad (8)$$

$$\text{inventory cost} = \sum_n \sum_t \text{IC}_t \cdot (\text{VT}_{t,n} - \text{VT}_t^{\min}) \quad (9)$$

3.2. Source Nodes. Volumetric flow from the source nodes must be between the minimum and maximum available amounts, as stated by eq 10. The qualities of the source streams are fixed to prespecified values (i.e., they are assumed to be known); see eq 11.

$$A_{s,n}^{\min} \leq \text{VS}_{s,n} \leq A_{s,n}^{\max} \quad \forall n, s \in \text{SRC} \quad (10)$$

$$\text{QS}_{s,q,n} = \text{QS}_{s,q,n}^{\text{fix}} \quad \forall n, s \in \text{SRC}, q: (s, q) \in \mathbf{SQ} \quad (11)$$

3.3. Destination Nodes (Demand Nodes). Volumetric flow into the destination nodes must be between the minimum and maximum demand requirements, as per eq 12. We do not enforce quality constraints on demand nodes because by imposing the minimum and maximum quality specifications on the blenders' output and assuming an initial on-spec material in the tanks, the streams into the demand nodes will be within specifications.

$$D_{s,n}^{\min} \leq \text{VS}_{s,n} \leq D_{s,n}^{\max} \quad \forall n, s \in \text{DST} \quad (12)$$

3.4. Blenders. Before describing the equations related to the blenders, note that index s' is an alias of index s (i.e., both represent streams from set \mathbf{S}). Material balance around a blender is given by eqs 13–15. Equation 16 states that if product s is to be blended, then the blender must produce an amount greater than the minimum specified, $\text{VB}_{s,n}^{\min}$. Maximum blending capacity is enforced by eqs 17 and 18. Binary variable $\text{bvb}_{s,n}$ is equal to 1 if product stream s is blended during period n , and 0 otherwise. Note that eqs 16 and 17 force $\text{VB}_{s,n}$ to be equal to zero if $\text{bvb}_{s,n}$ is 0.

$$\text{VB}_{s',n} = \sum_{s: (b,s) \in \text{BI}} \text{VBC}_{s,s',n} \quad \forall n, b, s': (b, s') \in \text{BO} \quad (13)$$

$$\sum_{s': (b,s') \in \text{BO}} \text{VBC}_{s,s',n} = \text{VS}_{s,n} \quad \forall n, b, s: (b, s) \in \text{BI} \quad (14)$$

$$\text{VB}_{s,n} = \text{VS}_{s,n} \quad \forall n, b, s: (b, s) \in \text{BO} \quad (15)$$

$$\text{VB}_{s,n} \geq \text{VB}_{s,n}^{\min} \text{bvb}_{s,n} \quad \forall n, b, s: (b, s) \in \text{BO} \quad (16)$$

$$\text{VB}_{s,n} \leq \text{VBR}_b^{\max} L_n \text{bvb}_{s,n} \quad \forall n, b, s: (b, s) \in \text{BO} \quad (17)$$

$$\sum_{s: (b,s) \in \text{BO}} \text{VB}_{s,n} \leq \text{VBR}_b^{\max} L_n \quad \forall n, b \quad (18)$$

An estimated blending time is calculated using eqs 19 and 20. In case the minimum blending rate VBR_b^{\min} is equal to 0, then eq 19 is not included in the model (in this work we consider nonzero values). Equation 21 ensures that the total blending time of blender b during period n is not greater than the length of such time period.

$$\text{TB}_{s,n} \leq \frac{\text{VB}_{s,n}}{\text{VBR}_b^{\min}} \quad \forall n, b, s: (b, s) \in \text{BO} \quad (19)$$

$$\text{TB}_{s,n} \geq \frac{\text{VB}_{s,n}}{\text{VBR}_b^{\max}} \quad \forall n, b, s: (b, s) \in \text{BO} \quad (20)$$

$$\sum_{s: (b,s) \in \text{BO}} \text{TB}_{s,n} \leq L_n \quad \forall n, b \quad (21)$$

Equation 22 constrains the product composition between the minimum and maximum specified limits.

$$\begin{aligned} \text{BR}_{s,s'}^{\min} \text{VB}_{s',n} & \leq \text{VBC}_{s,s',n} \leq \text{BR}_{s,s'}^{\max} \text{VB}_{s',n} \\ \forall n, b, s: (b, s) \in \text{BI}, s': (b, s') \in \text{BO} \end{aligned} \quad (22)$$

We assume that the quality properties blend linearly, on either a volumetric basis or a weight basis, eqs 23–26.

$$\begin{aligned} \text{QVBC}_{s,s',q,n} &= \text{VBC}_{s,s',n} \text{QS}_{s,q,n} \\ \forall n, b, s: (b, s) \in \text{BI}, s': (b, s') \in \text{BO}, q: (s, q) \in \text{SQB} \end{aligned} \quad (23)$$

$$\begin{aligned} \text{QMBC}_{s,s',q,n} &= \text{QVBC}_{s,s',q,n} \text{QS}_{s',qg',n} \\ \forall n, b, s: (b, s) \in \text{BI}, s': (b, s') \in \text{BO}, \\ q \in \{q|q \in \text{QLW} \wedge q: (s, q) \in \text{SQB}\} \end{aligned} \quad (24)$$

$$\begin{aligned} \text{QS}_{s',q}^{\min} \text{VB}_{s',n} &\leq \sum_{s:(b,s) \in \text{BI}} \text{QVBC}_{s,s',q,n} \leq \text{QS}_{s',q}^{\max} \text{VB}_{s',n} \\ \forall n, b, s': (b, s') \in \text{BO}, \\ q \in \{q|q \in \text{QLV} \wedge q: (s', q) \in \text{SQB}\} \end{aligned} \quad (25)$$

$$\begin{aligned} \text{QS}_{s',q}^{\min} \sum_{s:(b,s) \in \text{BI}} \text{QVBC}_{s,s',q,n} &\leq \sum_{s:(b,s) \in \text{BI}} \text{QMBC}_{s,s',q,n} \\ &\leq \text{QS}_{s',q}^{\max} \sum_{s:(b,s) \in \text{BI}} \text{QVBC}_{s,s',q,n} \\ \forall n, b, s': (b, s') \in \text{BO}, \\ q \in \{q|q \in \text{QLW} \wedge q: (s', q) \in \text{SQB}\} \end{aligned} \quad (26)$$

Note that although variables $\text{VBC}_{s,s',n}$, $\text{VB}_{s,n}$, $\text{QVBC}_{s,s',q,n}$, $\text{QMBC}_{s,s',q,n}$, $\text{TB}_{s,n}$ and $\text{bvb}_{s,n}$ are related to the blenders, the subscript b is not employed in our notation. The reason is that such variables are related to the corresponding blenders by sets **BI**, **BO**, and **SQB**.

3.5. Storage Tanks. The material balance around a tank is given by eqs 27 and 28. The minimum and maximum inventory levels are respected by including eq 29. Equation 30 ensures that the total withdrawal rate must be between the minimum and maximum limits. Equations 27–30 are used for all storage tanks.

$$\begin{aligned} \sum_{s:(t,s) \in \text{TI}} \text{VS}_{s,n} + \text{VT}_t^{\text{ini}} &= \text{VT}_{t,n} + \sum_{s':(t,s') \in \text{TO}} \text{VS}_{s',n} \\ \forall t, n = 1 \end{aligned} \quad (27)$$

$$\begin{aligned} \sum_{s:(t,s) \in \text{TI}} \text{VS}_{s,n} + \text{VT}_{t,n-1} &= \text{VT}_{t,n} + \sum_{s':(t,s') \in \text{TO}} \text{VS}_{s',n} \\ \forall t, n > 1 \end{aligned} \quad (28)$$

$$\text{VT}_t^{\min} \leq \text{VT}_{t,n} \leq \text{VT}_t^{\max} \quad \forall t, n \quad (29)$$

$$\text{VTR}_t^{\min} L_n \leq \sum_{s:(t,s) \in \text{TO}} \text{VS}_{s,n} \leq \text{VTR}_t^{\max} L_n \quad \forall t, n \quad (30)$$

Equation 31 defines the quality of the outlet streams of a tank as a function (which may be represented by a set of equality and/or inequality constraints) of the quality of the inlet streams, the quality of the initial inventory, the volumetric flows of the inlet streams, and the initial and final inventory levels. Note that the quality of the initial inventory is represented by the quality of the outlet stream of the previous period (i.e., perfect mixing in the tank is assumed).

$$\begin{aligned} \text{QS}_{s,q,n} &= g_1(\text{QS}_{s',q,n'} \text{QS}_{s,q,n-1} \text{VT}_{t,n} \text{VT}_{t,n-1}^{\dots}) \\ \forall n, t, s: (t, s) \in \text{TO}, s': (t, s') \in \text{TI}, q: (s, q) \in \text{SQ} \end{aligned} \quad (31)$$

The actual form of eq 31 depends on the specific model employed in each tank (Supporting Information).

3.6. Processing Units with a Single Operating Mode.

As mentioned at the beginning of section 3, set **U** does not include blenders nor storage tanks. The remaining processing units in the refinery network can be classified depending on their function (e.g., crude distillation columns, mixers, splitters, hydrotreating units, etc.) and based on their different number of operating modes (e.g., single or multiple). The equations presented in this subsection are defined for all units with a single operating mode (i.e., $u \in \text{SMU}$).

The volumetric flow fed to a unit during a given time period is defined by eq 32. The feed must be between the minimum and maximum processing capacities of the unit, as stated by eq 33. Equations 32 and 33 are written for all units with a single operating mode.

$$\text{VU}_{u,n} = \sum_{s:(u,s) \in \text{UI}} \text{VS}_{s,n} \quad \forall n, u \in \text{SMU} \quad (32)$$

$$\text{VUR}_u^{\min} L_n \leq \text{VU}_{u,n} \leq \text{VUR}_u^{\max} L_n \quad \forall n, u \in \text{SMU} \quad (33)$$

The volumetric flow of an outlet stream from unit u is defined as a function of the individual inlet flows, the total feed flow to unit u , the quality of the individual inlet streams, and/or other operating variables.

$$\begin{aligned} \text{VS}_{s,n} &= g_2(\text{VS}_{s',n'} \text{VU}_{u,n'} \text{QS}_{s',q,n'}^{\dots}) \\ \forall n, u \in \text{SMU}, s: (u, s) \in \text{UO}, s': (u, s') \in \text{UI}, \\ q: (s, q) \in \text{SQ} \end{aligned} \quad (34)$$

The value of quality property q in outlet stream s of unit u is defined as a function of the quality of the individual inlet streams, the individual inlet flows, the total feed flow to unit u , and/or other operating variables.

$$\begin{aligned} \text{QS}_{s,q,n} &= g_3(\text{QS}_{s',q,n'} \text{VS}_{s',n'} \text{VU}_{u,n'}^{\dots}) \\ \forall n, u \in \text{SMU}, s: (u, s) \in \text{UO}, s': (u, s') \in \text{UI}, \\ q: (s, q) \in \text{SQ} \end{aligned} \quad (35)$$

The actual form of eqs 34 and 35 will depend on the specific unit under consideration (Supporting Information). As an example, in the following subsections we show the models employed in this work for the splitters and the crude distillation unit (i.e., we assume these units have a single operating mode).

3.6.1. Splitters. The splitter model is very simple. Equation 36 constrains the total outlet flow of the splitter to be equal to its total feed. Equation 37 establishes that the quality of the outlets is identical to the quality of the inlet. The entire model for a splitter is constituted by eqs 32 and 33 and eqs 36 and 37. Note that eqs 36 and 37 are the actual form of eqs 34 and 35, respectively, when applied to a splitter.

$$\sum_{s:(u,s) \in \text{UO}} \text{VS}_{s,n} = \text{VU}_{u,n} \quad \forall n, u \in \text{SP} \quad (36)$$

$$\begin{aligned} QS_{s,q,n} &= QS_{s',q,n} \quad \forall n, u \in \mathbf{SP}, s: (u, s) \in \mathbf{UO}, \\ s': (u, s') \in \mathbf{UI}, q: (s, q) \in \mathbf{SQ} \end{aligned} \quad (37)$$

3.6.2. Crude Distillation Units. The CDU is described here using a simple fixed yield model with a single operating mode. Therefore, the output flows and the output qualities depend only on the feed composition. The amount of material processed by the CDU is determined by eq 32. Equation 33 constrains the CDU to operate between its minimum and maximum capacity. Equation 38 computes the size of the final cuts using a simple fixed yield approach. The crude mix recipe of a CDU during period n (i.e., set of variables $CMR_{s,n}$) is defined by eqs 39 and 40. It is assumed that the quality properties blend linearly, on either a volumetric or weight basis. The quality of the outlet streams is determined by eqs 41 and 42. Note that to use eqs 41 and 42, the specific gravity must be assumed to blend linearly on a volumetric basis. Equation 38 is the actual form of eq 34 for a CDU, whereas eqs 39–42 correspond to eq 35. The entire CDU model consists of eqs 32, 33, and 38–42.

$$\begin{aligned} VS_{s',n} &= \sum_{s:(u,s) \in \mathbf{UI}} YCDU_{s,s'}^{\text{fix}} VS_{s,n} \\ \forall n, u \in \mathbf{CDU}, s': (u, s') \in \mathbf{UO} \end{aligned} \quad (38)$$

$$\begin{aligned} VS_{s,n} &= VU_{u,n} CMR_{s,n} \\ \forall n, u \in \mathbf{CDU}, s: (u, s) \in \mathbf{UIR} \end{aligned} \quad (39)$$

$$\sum_{s:(u,s) \in \mathbf{UI}} CMR_{s,n} = 1 \quad \forall n, u \in \mathbf{CDU} \quad (40)$$

$$\begin{aligned} QS_{s',q,n} &= \sum_{s:(u,s) \in \mathbf{UI}} QCO_{q,s,s'}^{\text{fix}} CMR_{s,n} \\ \forall n, u \in \mathbf{CDU}, s': (u, s') \in \mathbf{UO}, \\ q \in \{q|q \in \mathbf{QLV} \wedge q: (s, q) \in \mathbf{SQ}\} \end{aligned} \quad (41)$$

$$\begin{aligned} QS_{s',q,n} QS_{s',q',n} &= \sum_{s:(u,s) \in \mathbf{UI}} QCO_{q,s,s'}^{\text{fix}} QCO_{q',s,s'}^{\text{fix}} CMR_{s,n} \\ \forall n, u \in \mathbf{CDU}, s': (u, s') \in \mathbf{UO}, \\ q \in \{q|q \in \mathbf{QLW} \wedge q: (s, q) \in \mathbf{SQ}\} \end{aligned} \quad (42)$$

3.7. Processing Units with Multiple Operating Modes.

If a unit has more than one distinct operating mode, each mode is represented as a different unit u . Then, by using the set of real units with multiple operating modes (\mathbf{RU}), the set of units representing an operating mode (\mathbf{MMU}), and the set that associates the different operating modes to each real unit (\mathbf{RUU}), the minimum and maximum physical capacity constraints of a real unit are enforced.

If unit u is from set \mathbf{MMU} and it is going to operate during period n , then binary variable $bv_{u,n}$ is equal to 1, and it must process more than the minimum rate and at least a minimum amount $VU_{u,n}^{\text{min}}$ as established by eqs 44 and 45, respectively. If $bv_{u,n} = 0$, then unit u does not operate in time period n and $VU_{u,n} = 0$ by eq 44. The units representing different operating modes of a real unit must not exceed the maximum capacity of the real unit; see eq 46. An estimate of the time spent in each mode is calculated by using eqs 47 and 48. If the minimum processing rate is zero, i.e., $VUR_u^{\text{min}} = 0$, then eq 47 is not

included in the model. In this work, $VUR_u^{\text{min}} > 0$ for all u . The time spent in all operating modes must not exceed the length of the corresponding time period (eq 49).

$$VU_{u,n} = \sum_{s:(u,s) \in \mathbf{UI}} VS_{s,n} \quad \forall n, u \in \mathbf{MMU} \quad (43)$$

$$VU_{u,n} \leq VUR_u^{\text{max}} L_n bv_{u,n} \quad \forall n, u \in \mathbf{MMU} \quad (44)$$

$$VU_{u,n} \geq VU_{u,n}^{\text{min}} bv_{u,n} \quad \forall n, u \in \mathbf{MMU} \quad (45)$$

$$VUTR_{ru}^{\text{min}} L_n \leq \sum_{u:(ru,u) \in \mathbf{RUU}} VU_{u,n} \leq VUTR_{ru}^{\text{max}} L_n \quad \forall n, ru \quad (46)$$

$$TU_{u,n} \leq \frac{VU_{u,n}}{VUR_u^{\text{min}}} \quad \forall n, u \in \mathbf{MMU} \quad (47)$$

$$TU_{u,n} \geq \frac{VU_{u,n}}{VUR_u^{\text{max}}} \quad \forall n, u \in \mathbf{MMU} \quad (48)$$

$$\sum_{u:(ru,u) \in \mathbf{RUU}} TU_{u,n} \leq L_n \quad \forall n, ru \quad (49)$$

Similarly to units with single operating modes, the volumetric flow of an outlet stream, and the value of quality property q of an outlet stream, are functions of the individual inlet flows, the total feed flow to unit u , the quality of the individual inlet streams, and/or other operating variables (eqs 50 and 51). Their actual form will depend on the specific unit under consideration (Supporting Information).

$$\begin{aligned} VS_{s,n} &= g_q(VS_{s',n}, VU_{u,n}, QS_{s',q,n}, \dots) \\ \forall n, u \in \mathbf{MMU}, s: (u, s) \in \mathbf{UO}, s': (u, s') \in \mathbf{UI}, \\ q: (s, q) \in \mathbf{SQ} \end{aligned} \quad (50)$$

$$\begin{aligned} QS_{s,q,n} &= g_S(QS_{s',q,n}, VS_{s',n}, VU_{u,n}, \dots) \\ \forall n, u \in \mathbf{MMU}, s: (u, s) \in \mathbf{UO}, s': (u, s') \in \mathbf{UI}, \\ q: (s, q) \in \mathbf{SQ} \end{aligned} \quad (51)$$

3.8. Maximum Number of Operating Modes and Blended Products per Period.

We allow different operating modes to be run during the same time period. In addition, we allow the blenders to produce more than one specific product in each period. Equations 52 and 53 can be included in the model if these two assumptions are not valid. UMM_{ru} is the maximum number of modes in which a real unit ru can operate during a given period, whereas BMP_b is the maximum number of distinct products that can be produced by blender b within a time period.

$$\sum_{u:(ru,u) \in \mathbf{RUU}} bv_{u,n} \leq UMM_{ru} \quad \forall n, ru \quad (52)$$

$$\sum_{s:(b,s) \in \mathbf{BO}} bvb_{s,n} \leq BMP_b \quad \forall n, b \quad (53)$$

The reason to not include eqs 52 and 53 in our model is that if the time periods are too long (e.g., weeks or months) then, most likely, multiple operating modes will be run and more than a single product will be blended in each period. Even if the time periods are only a few days long, a subsequent scheduling step can be included to determine exactly at what point in

time a unit will change operating mode. This is one of our future work directions.

3.9. Constraints for Quality Properties of Inlet Streams. If the quality properties of inlet streams to the units must be between minimum and maximum limits, then eqs 54 and 55 can be added to the planning model. Note that these constraints are not used in the examples shown in sections 6 and 7 of this paper. The reason to present these equations is their usefulness when correlations for the processing units (which are commonly developed for a given range of the input quality properties) are employed.

$$\begin{aligned} \text{QS}_{s,q,m} &\geq \text{QSUI}_{s,q}^{\min} \\ \forall n, u, s: (u, s) &\in \text{UI}, q \in \{q|q: (s, q) \in \text{SQ} \wedge \\ q: (u, q) &\in \text{UQ}\} \end{aligned} \quad (54)$$

$$\begin{aligned} \text{QS}_{s,q,m} &\leq \text{QSUI}_{s,q}^{\max} \\ \forall n, u, s: (u, s) &\in \text{UI}, q \in \{q|q: (s, q) \in \text{SQ} \wedge \\ q: (u, q) &\in \text{UQ}\} \end{aligned} \quad (55)$$

3.10. Model Overview. Summarizing, the MINLP model to be solved in this work is composed of eqs 1–51. The bilinear terms appear in eqs 23, 24, 39, and 42, and they can appear in eqs 31, 34, 35, 50, and 51, depending on the specific functions being employed. The reader is encouraged to read the Supporting Information for a detailed description of the equations employed and the bilinear terms found in the model.

4. ELEMENTS OF THE GLOBAL OPTIMIZATION ALGORITHM

The refinery planning MINLP problem from section 3 can be represented by **P**. The continuous variables are denoted by x and the binary variables by y . \mathbf{M} is the set of constraints plus the objective function, which are represented by general bilinear functions $f_m(x, y)$. Set \mathbf{BL} represents the pairs of variables (i, j) involved in bilinear terms $x_i x_j$. lx is the length of vector x , and ly is the length of vector y .

$$\begin{aligned} \min f_0(x, y) \\ \text{s.t.} \\ f_m(x, y) &\leq 0 \quad \forall m \in \mathbf{M} \setminus \{0\} \\ f_m(x, y) &= \sum_{(i,j) \in \mathbf{BL}} a_{ijm} x_i x_j + B_m x + C_m y + d_m \quad \forall m \in \mathbf{M} \\ 0 &\leq x^L \leq x \leq x^U \\ x &\in \mathcal{R}^{lx}, y \in \{0, 1\}^{ly} \end{aligned} \quad (\mathbf{P})$$

4.1. Lower Bounding. Problem **P** is the relaxation of **P**. Bilinear terms $x_i x_j$ are substituted by variables w_{ij} , thus linearizing $f_m(x, y)$ into $f_m^R(x, y)$. The values of the w_{ij} variables are determined by a set of linear constraints. The feasible region of such constraints is represented by W .

$$\begin{aligned} \min f_0^R(x, y) \\ \text{s.t.} \\ f_m^R(x, y) &\leq 0 \quad \forall m \in \mathbf{M} \setminus \{0\} \\ f_m^R(x, y) &= \sum_{(i,j) \in \mathbf{BL}} a_{ijm} w_{ij} + B_m x + C_m y + d_m \quad \forall m \in \mathbf{M} \\ 0 &\leq x^L \leq x \leq x^U \\ x &\in \mathcal{R}^{lx}, y \in \{0, 1\}^{ly}, w \in W \subset \mathcal{R}^{|\mathbf{BL}|} \end{aligned} \quad (\mathbf{PR})$$

If the piecewise McCormick relaxation (PMCR) is used, then the feasible region W will be given by eqs 56–64, where NP represents the number of partitions being used. All the variables are continuous except for binary variable $z_{j,np}$.

$$w_{ij} \geq \sum_{np=1}^{\text{NP}} (\hat{x}_{ij,np}^L x_{j,np}^L + \hat{x}_{j,np}^L x_i^L - z_{j,np} x_i^L x_{j,np}^L) \quad \forall (i, j) \in \mathbf{BL} \quad (56)$$

$$w_{ij} \geq \sum_{np=1}^{\text{NP}} (\hat{x}_{ij,np}^U x_{j,np}^U + \hat{x}_{j,np}^U x_i^U - z_{j,np} x_i^U x_{j,np}^U) \quad \forall (i, j) \in \mathbf{BL} \quad (57)$$

$$w_{ij} \leq \sum_{np=1}^{\text{NP}} (\hat{x}_{ij,np}^L x_{j,np}^L + \hat{x}_{j,np}^U x_i^U - z_{j,np} x_i^U x_{j,np}^L) \quad \forall (i, j) \in \mathbf{BL} \quad (58)$$

$$w_{ij} \leq \sum_{np=1}^{\text{NP}} (\hat{x}_{ij,np}^U x_{j,np}^U + \hat{x}_{j,np}^L x_i^L - z_{j,np} x_i^L x_{j,np}^U) \quad \forall (i, j) \in \mathbf{BL} \quad (59)$$

$$x_i = \sum_{np=1}^{\text{NP}} \hat{x}_{ij,np} \quad \forall (i, j) \in \mathbf{BL} \quad (60)$$

$$x_j = \sum_{np=1}^{\text{NP}} \hat{x}_{j,np} \quad \forall j: (i, j) \in \mathbf{BL} \quad (61)$$

$$\sum_{np=1}^{\text{NP}} z_{j,np} = 1 \quad \forall j: (i, j) \in \mathbf{BL} \quad (62)$$

$$x_i^L z_{j,np} \leq \hat{x}_{ij,np} \leq x_i^U z_{j,np} \quad \forall (i, j) \in \mathbf{BL}, np \in \{1, \dots, \text{NP}\} \quad (63)$$

$$x_{j,np}^L z_{j,np} \leq \hat{x}_{j,np} \leq x_{j,np}^U z_{j,np} \quad \forall j: (i, j) \in \mathbf{BL}, np \in \{1, \dots, \text{NP}\} \quad (64)$$

Note that the lower and upper bounds for discretized variables x_j^L , $x_{j,np}^L$, and $x_{j,np}^U$ are computed before solving **PR** with eqs 65 and 66.

$$x_{j,np}^L = x_j^L + \frac{(x_j^U - x_j^L)(np - 1)}{\text{NP}} \quad \forall j: (i, j) \in \mathbf{BL}, np \in \{1, \dots, \text{NP}\} \quad (65)$$

$$x_{j,np}^U = x_j^U - \frac{(x_j^U - x_j^L)np}{\text{NP}} \quad \forall j: (i, j) \in \mathbf{BL}, np \in \{1, \dots, \text{NP}\} \quad (66)$$

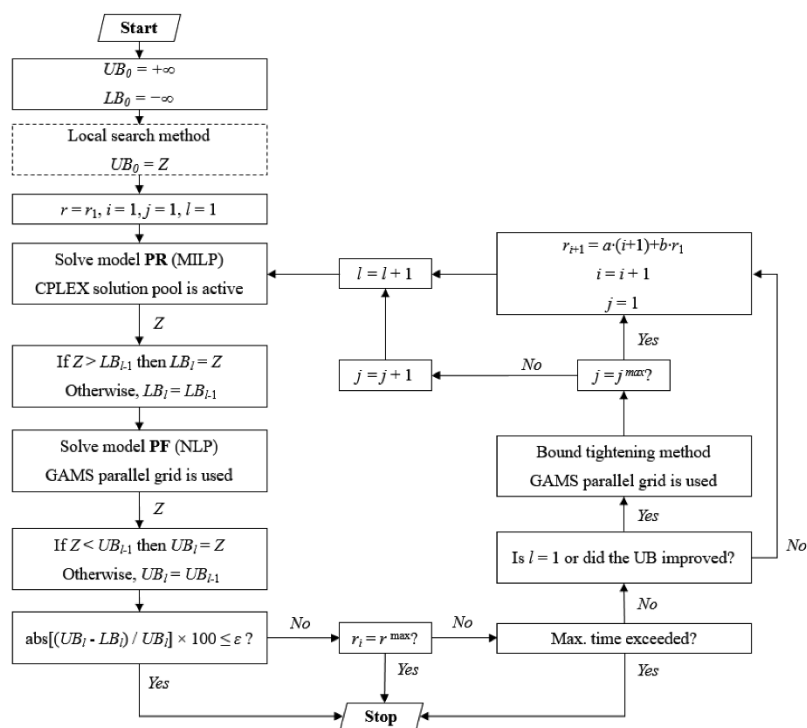


Figure 3. Global algorithm flowchart (minimization problem).

If normalized multiparametric disaggregation technique (NMDT) is employed, then eqs 67 to 77 will delineate feasible region W . The binary variables are z_{jkl} and the rest are continuous. In this case, the number of partitions is defined by $NP = 10^{-p}$, where p is a negative integer specified by the user.

$$w_{ij} = x_i x_j^L + v_{ij}(x_j^U - x_j^L) \quad \forall (i, j) \in \mathbf{BL} \quad (67)$$

$$x_j = x_j^L + \lambda_j(x_j^U - x_j^L) \quad \forall j: (i, j) \in \mathbf{BL} \quad (68)$$

$$\lambda_j = \sum_{l=p}^{-1} \sum_{k=0}^9 10^l \cdot k \cdot z_{jkl} + \Delta \lambda_j \quad \forall j: (i, j) \in \mathbf{BL} \quad (69)$$

$$0 \leq \Delta \lambda_j \leq 10^p \quad \forall j: (i, j) \in \mathbf{BL} \quad (70)$$

$$v_{ij} = \sum_{l=p}^{-1} \sum_{k=0}^9 10^l \cdot k \cdot \hat{x}_{ijkl} + \Delta v_{ij} \quad \forall (i, j) \in \mathbf{BL} \quad (71)$$

$$x_i^L \Delta \lambda_j \leq \Delta v_{ij} \leq x_i^U \Delta \lambda_j \quad \forall (i, j) \in \mathbf{BL} \quad (72)$$

$$\Delta v_{ij} \leq 10^p(x_i - x_i^L) + x_i^L \Delta \lambda_j \quad \forall (i, j) \in \mathbf{BL} \quad (73)$$

$$\Delta v_{ij} \geq 10^p(x_i - x_i^U) + x_i^U \Delta \lambda_j \quad \forall (i, j) \in \mathbf{BL} \quad (74)$$

$$x_i = \sum_{k=0}^9 \hat{x}_{ijkl} \quad \forall (i, j) \in \mathbf{BL}, l \in \{p, \dots, -1\} \quad (75)$$

$$\sum_{k=0}^9 z_{jkl} = 1 \quad \forall j: (i, j) \in \mathbf{BL}, l \in \{p, \dots, -1\} \quad (76)$$

$$x_i^L z_{jkl} \leq \hat{x}_{ijkl} \leq x_i^U z_{jkl}$$

$$\forall (i, j) \in \mathbf{BL}, l \in \{p, \dots, -1\}, k \in \{0, \dots, 9\} \quad (77)$$

There is no specific rule to select which variables should be discretized in PMCR or NMDT. In this work, the second variable in the bilinear term, as written in eqs 23, 24, 31, 34, 35, 39, 42, 50, and 51, is the one discretized. When PR is solved, it is the best possible solution at termination that is employed as the lower bound for the objective function, which is not necessarily the best feasible solution found.

4.2. Upper Bounding. Constrained problem PF is derived from P after fixing the binary variables to the values from the solution of PR, i.e., $y = \hat{y}$.

$$\min f_0(x)$$

s.t.

$$f_m(x) \leq 0 \quad \forall m \in \mathbf{M} \setminus \{0\}$$

$$f_m(x) = \sum_{(i,j) \in \mathbf{BL}} a_{ijm} x_i x_j + B_m x + C_m \hat{y} + d_m \quad \forall m \in \mathbf{M}$$

$$0 \leq x^L \leq x \leq x^U$$

$$x \in \mathcal{R}^{lx}$$

(PF)

4.3. Bound Tightening. The bounds of all variables x_{ij} involved in bilinear terms (both discretized and nondiscretized, i.e., $h \in \{h \mid (h,j) \in \mathbf{BL}_v, (i,h) \in \mathbf{BL}\}$) are updated by solving one minimization (leads to lower bound x_{ij}^L) and one maximization problem (x_{ij}^U) per variable. The MILP problems in this step are denoted as PRBmin and PRBmax. Note that the bound of a variable is updated with the best possible solution at

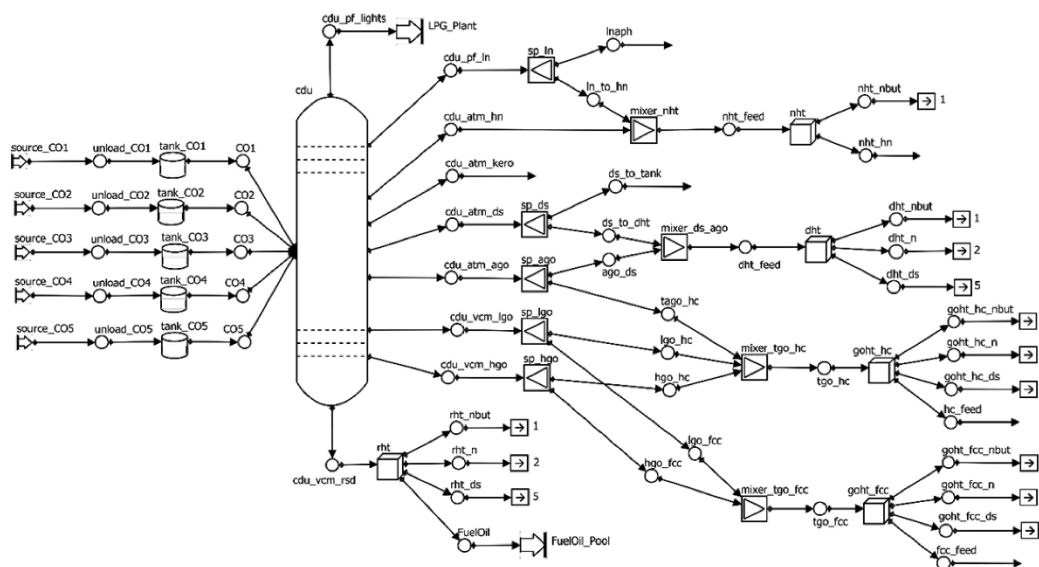


Figure 4. Crude oil unloading section, including CDU and hydrotreaters.

termination and not with the best solution found. Also, we add the constraint that the objective function of original problem **P** must be lower than or equal to the current upper bound **UB** (the best feasible solution found).

$$\begin{aligned}
 x_h^L &:= \min x_h, x_h^U := \max x_h \\
 \text{s.t.} \\
 f_m^R(x, y) &\leq 0 \quad \forall m \in \mathbf{M} \setminus \{0\} \\
 f_m^R(x, y) &= \sum_{(i,j) \in \mathbf{BL}} a_{ijm} w_{ij} + B_m x + C_m y + d_m \quad \forall m \in \mathbf{M} \\
 f_0^R(x, y) &\leq \mathbf{UB} \\
 0 &\leq x^L \leq x \leq x^U \\
 x &\in \mathcal{R}^{kx}, y \in \{0, 1\}^{ly}, w \in W \subset \mathcal{R}^{bl}
 \end{aligned}$$

(PRBmin/max)

For our examples, the feasible region W for models **PRBmin/max** is derived using $\text{NP} = 1$ in eqs 55–65 of the piecewise McCormick relaxation, which is equivalent to the standard McCormick relaxation.²⁰

5. GLOBAL OPTIMIZATION ALGORITHM

The proposed global optimization algorithm can be applied to any mixed-integer nonlinear program where the nonlinearities are due to strictly bilinear and quadratic terms involving continuous variables. The flowchart for a minimization problem is shown in Figure 3, and a detailed description follows. Note that we use the term “resolution” and “resolution factor” to describe a general parameter r that controls the number of partitions. The resolution is given by parameters p and NP when **NMDT** and **PMCR** are used, respectively.

Step 1: Set the initial lower bound (**LB**) equal to $-\infty$, and the initial upper bound (**UB**) to $+\infty$.

Step 2: This step is optional but it is recommended for medium- and large-size models to find an initial feasible solution fairly quickly. Our local search method employs a general branch-and-bound algorithm where at each node in the

tree, a local **NLP** solver is employed. If no feasible solution is found, repeat the procedure with a different starting point; otherwise, go to step 3.

Step 3: Set the following parameters to their initial values: resolution of the discretization method ($r = r_1$), total number of iterations ($l = 1$), number of iterations where resolution is increased ($i = 1$), and number of iterations without changing the current resolution ($j = 1$).

Step 4: Solve relaxed problem **PR** using the **CPLEX** **MILP** solver activating the solution pool and parallel options. Set the pool capacity to k (usually, $k > 5$) and the number of threads. The solution pool consists of integer feasible solutions found by **CPLEX** while solving model **PR** before reducing the optimality gap below the specified tolerance (or before reaching the maximum time limit). Not all the solutions in the pool have the same objective function value.

Step 5: Update the lower bound **LB** in the case of improvement. The best possible solution of model **PR** at termination is used to determine if **LB** needs to be updated (recall section 4.1).

Step 6: Update and fix the values of the binary variables according to the pool solutions of **PR** and solve in parallel the k instances of **PF** using a local **NLP** solver.

Step 7: Update the upper bound **UB** if necessary.

Step 8: Compute the optimality gap and if it is equal to or smaller than the tolerance ϵ , then stop. Otherwise, continue to step 9.

Step 9: If the resolution of the discretization method is equal to the maximum allowed (i.e., $r_i = r^{\max}$), or if the total time exceeds the specified limit, stop. Otherwise continue to step 10.

Step 10: If this is the first iteration of the algorithm (i.e., $l = 1$), or if the upper bound has improved more than a prespecified percentage, then apply the optimality-based bound tightening method.

Step 11: If the number of iterations without changing the resolution of the discretization method is equal to the maximum (i.e., $j = j^{\max}$), then go to step 12. Otherwise, set $l = l + 1$ and go back to step 4.

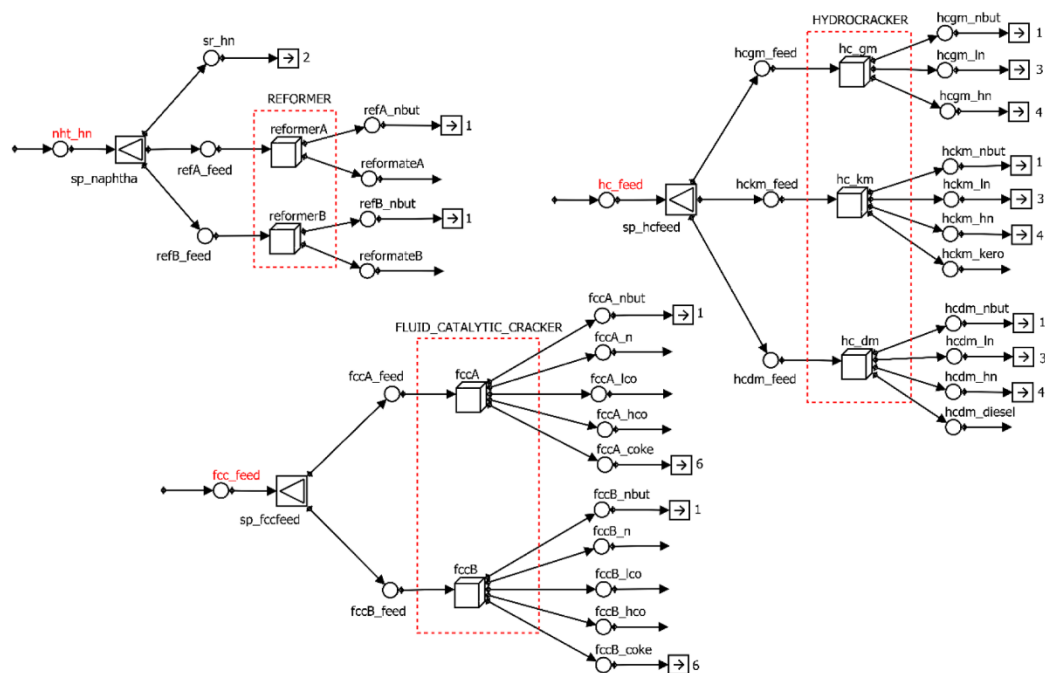


Figure 5. Processing units section. (Units inside the red-dashed boxes represent the different operating modes of a real unit.)

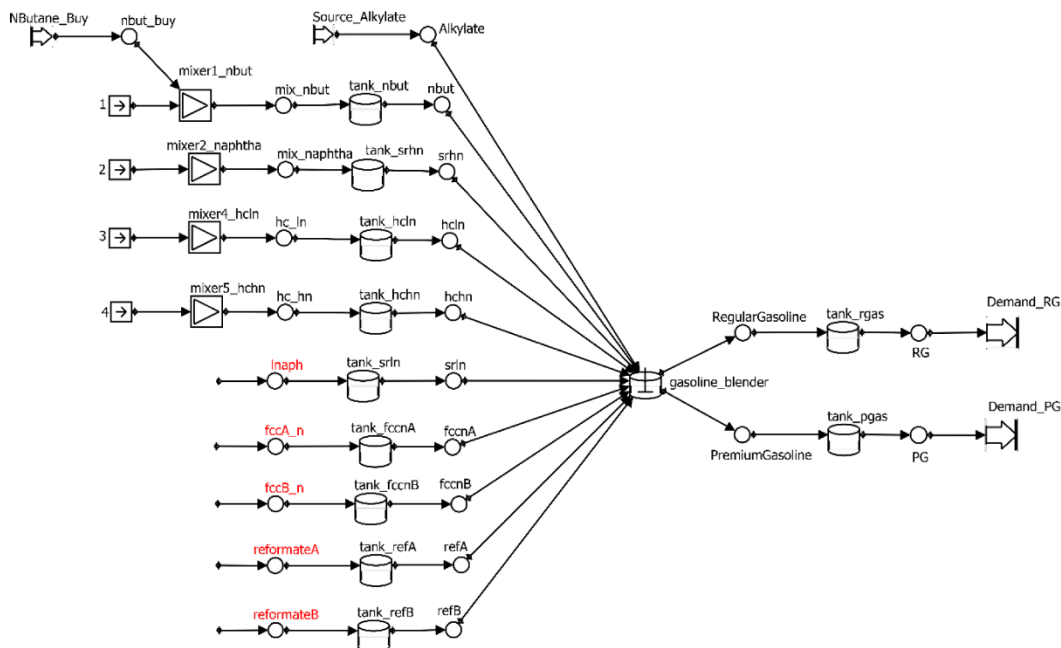


Figure 6. Gasoline blending section.

Step 12: Increase the resolution factor r . Afterward, set $i = i + 1$, $j = 1$, $l = l + 1$, and go back to step 4.

We have chosen to increase the resolution factor r as follows: $r_{i+1} = a(i + 1) + br_i$, where r_1 is the initial resolution, i is the iteration number associated with the current resolution, and a and b are prespecified parameters. For the alternative

discretization techniques, we have for PMCR, r is equal to the total number of partitions NP , and we use $r_1 = 4$, $a = 4$, $b = 0$, and $r^{\max} = 24$; it means that the algorithm starts with 4 partitions (per discretized variable) and adds another 4 in each iteration i ($NP = 4, 8, \dots, 24$); for NMDT, r is equivalent to parameter p , and we use $r_1 = -1$, $a = -1$, $b = 0$, and $r^{\max} = -3$;

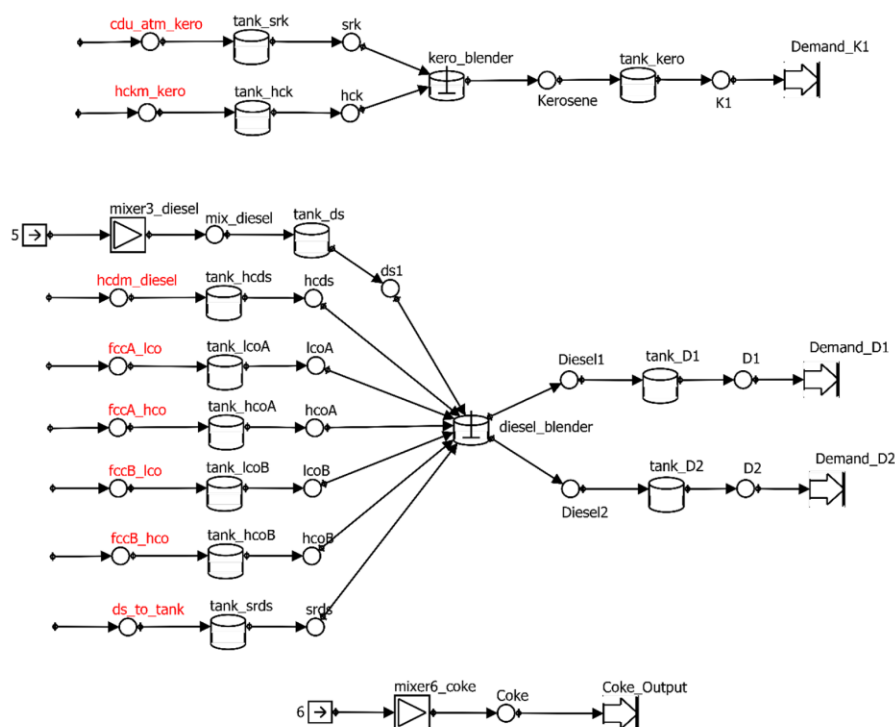


Figure 7. Diesel and kerosene blending pools and coke output.

therefore, the algorithm with NMDT starts with 10 partitions and increases by 10 in each iteration i ($NP = 10, 100, 1000$). Note that usual values for parameter b are 0 and 1. With $b = 0$, then r_1 must be less than $2|a|$, for the resolution to increase in the second iteration. Setting $b = 1$ is recommended for a high initial resolution and small increments per iteration.

Parameter j^{\max} is set equal to 10.

6. INDUSTRIAL CASE STUDY

6.1. Refinery System. We used the free and open source diagram editor software called DIA to create the flowsheet of the refinery plant shown in Figures 4–7 and to input the name of the streams, the name of the units, and the mathematical model to be used for each unit. Then we coded a Python script that retrieves such data from the DIA file and creates a single GAMS file with the corresponding models required. We entered the numerical data into an Excel file that is read by the GAMS file when this is run.

In the system, there is one crude distillation unit (cdu), five hydrotreaters (nht, dht, goth_fcc, goth_hc, and rht), one catalytic reforming unit (with two operating modes: reformerA and reformerB), one hydrocracking unit (with three operating modes: hc_gm, hc_km, and hc_dm), one fluid catalytic cracking unit (with two operating modes: fccA and fccB), three blenders (one for each distinct product pool), and several storage tanks. The maximum processing rate of the CDU is 120 kbbbl/day. We consider five different type of crude oils (CO1 to CO5), two gasoline grades (PG and RG), one kerosene grade (K1), and two diesel grades (D1 and D2). There are eight quality properties under consideration: specific gravity “sg”, research octane number “ron”, motor octane number “mon”, Reid vapor pressure “rvp” (psig), aromatics

content “arom” (% vol), sulfur content “sul” (% wt), cetane index “cin”, and pour point “pour” (°R). We assume that these quality properties blend linearly on a volumetric basis, except for the sulfur content, which we consider to blend linearly on a weight basis.

6.2. Data for Example Problems. Due to space limitations, we only present part of the data in the main text (enough to give the reader a sense of how big and complex the problem is). The rest of the data required by the model are included in the Supporting Information.

The volumetric yield data of the crude oils are shown in Table 1. Quality specifications of finished products are described in Table 2. We assume that we can purchase alkylate and *n*-butane to use as blend components for gasoline products, if necessary. Their minimum and maximum available amounts of alkylate and *n*-butane are 0 and 120 kbbbl/day. Economic data are shown in Table 3. Note that we do not consider

Table 1. Crude Oil Yield Data (% vol) (Parameter $YCDU_{ss}^{\text{fix}} \times 100$)

cut	crude oil				
	CO1	CO2	CO3	CO4	CO5
LS	0.65	2.30	2.20	0.78	1.20
LN	11.19	12.05	7.80	11.76	9.05
HN	6.91	15.65	13.80	10.78	12.25
KR	5.92	9.50	13.30	3.93	8.30
DS	5.51	7.30	13.15	4.85	6.25
AGO	5.51	7.30	13.80	4.85	6.25
LGO	14.33	19.10	6.25	15.54	15.30
HGO	14.33	12.60	13.20	15.54	14.00
RSD	35.65	14.20	16.50	31.99	27.40

Table 2. Product Quality Specifications (Parameters $Q_{s,q}^{\min}$ and $Q_{s,q}^{\max}$)

product	quality property								
	Sg	ron	mon	rvp	arom	Sul	cin	pour	
				Minimum					
RG	0.73	88.5	78.5	0	0	0	NA ^a	NA	
PG	0.73	92.5	82.5	0	0	0	NA	NA	
K1	0.75	NA	NA	NA	NA	0	NA	0	
D1	0.81	NA	NA	NA	NA	0	40	0	
D2	0.81	NA	NA	NA	NA	0	40	0	
				Maximum					
RG	0.81	150	150	15	60	0.001	NA	NA	
PG	0.81	150	150	15	60	0.001	NA	NA	
K1	0.85	NA	NA	NA	NA	0.3	NA	407	
D1	0.87	NA	NA	NA	NA	0.0015	100	470	
D2	0.87	NA	NA	NA	NA	0.0015	100	456	

^aNA = lc not applicable.

Table 3. Economic Data

raw material	CS, (\$/bbl)	product	PS, (\$/bbl)
CO1	40	RG	89
CO2	41	PG	112
CO3	42	K1	96
CO4	38	D1	98
CO5	36	D2	109
alkylate	129	lc coke	0
n-butane	32	LPG	0

revenue from the coke output and the lights from the CDU. We also do not price inventory (i.e., IC_t is equal to 0) because the demand and supply flow rates are fixed (the refinery can store the remaining material as either blend components or finished products). Quality data for crude oils, alkylate, and n-butane streams can be found in the Supporting Information.

We consider four different scenarios regarding product demand and crude oil supply, denoted as SC1, SC2, SC3, and SC4. Daily data for these scenarios can be found in the Supporting Information and are used to generate the data for the one- and three-time-period problems. The single time period considers aggregate data for the planning horizon of 1 week (Table 4), whereas each period in the three-time-period problem corresponds to one or more days; check Table 5.

Table 4. Demand and Supply Data for 1-Period Models in kbbl (Parameters $D_{s,n}^{\min} = D_{s,n}^{\max}$ and $A_{s,n}^{\min} = A_{s,n}^{\max}$)

demand/supply scenario:	SC1	SC2	SC3
time period:	1	1	1
days:	1–7	1–7	1–7
period duration L_n (days):	7	7	7
RG	370	330	330
PG	180	160	220
K1	75	65	60
D1	70	60	70
D2	150	130	130
CO1	60	80	80
CO2	270	250	260
CO3	280	250	240
CO4	60	80	60
CO5	60	40	30

Scenarios SC3 and SC4 have the same total supply and demand. Therefore, we have seven different examples denoted as SC1-TP1, SC1-TP3, SC2-TP1, SC2-TP3, SC3-TP1, SC3-TP3, and SC4-TP3; where SC# refers to the scenario, and TP# to the number of time periods.

Note that although the model constraints in eqs 10 and 12 allow for supply and demand to vary within a given interval, the example problems feature fixed values (i.e., $A_{s,n}^{\min} = A_{s,n}^{\max}$ for $s \in \{CO1, CO2, CO3, CO4, CO5\}$, and $D_{s,n}^{\min} = D_{s,n}^{\max}$ for $s \in \{RG, PG, K1, D1, D2\}$, for all n).

7. NUMERICAL RESULTS

All the examples were solved on a Windows Server 2008 R2 Enterprise, AMD Opteron Processor 6386 SE, 2.79 GHz, 64 GB RAM, and 32 cores available. GAMS IDE 24.5.4 was used to implement the mathematical models and algorithms. BARON 15.9, ANTIGONE 1.1, and SBB were employed to solve the MINLPs, whereas CPLEX 12.6 and CONOPT 3.17A were respectively used for the MILP and NLP problems. The termination criteria for the MILP problems were: optimality gap of 0.01% or 1000 s. The CPLEX solution pool option was active for problems PR with a maximum pool capacity of 29 and the replacement option that generates diverse solutions. Because the best solution was also saved, a maximum of 30 instances of model PF were solved per iteration using the GAMS parallel computing grid facility. The instances of problems PRBmin and PRBmax were solved in parallel as well. For MILP models, CPLEX parallel option was active (in deterministic mode) with a maximum number of threads equal to 8. The termination criteria for the algorithm and commercial global solvers were an optimality gap of 0.01% or a maximum wall time equal to 10800 s.

The number of equations, total variables, binary variables, and bilinear terms of each problem are presented in Table 6. The optimality-based bound tightening problems have the same number of binaries. They feature 3487 equations and 1815 variables, and 11263 equations, and 5693 variables for one and three time periods, respectively. Table 7 shows the final results computed by ANTIGONE, BARON, and our global algorithm with the alternative PMCR and NMDT relaxations.

7.1. One-Time-Period Problems. Figure 8 shows the optimality gap vs time computed by ANTIGONE, BARON, and our proposed global algorithm for example SC2-TP1, which is representative of the results obtained for the

Table 5. Demand and Supply Data for 3-Period Models in kbbl (Parameters $D_{s,n}^{\min} = D_{s,n}^{\max}$ and $A_{s,n}^{\min} = A_{s,n}^{\max}$)

demand/supply scenario:	SC1			SC2			SC3			SC4		
	1	2	3	1	2	3	1	2	3	1	2	3
time period:	1	2	3	1	2	3	1	2	3	1	2	3
days:	1–5	6	7	1	2–5	6–7	1	2–4	5–7	1	2–4	5–7
duration L_n (days):	5	1	1	1	4	2	1	3	3	1	3	3
RG	210	80	80	40	130	160	40	170	120	50	230	50
PG	140	20	20	30	110	20	0	140	80	30	160	30
K1	50	15	10	10	30	25	0	30	30	10	40	10
D1	50	10	10	10	30	20	10	20	40	20	40	10
D2	120	10	20	0	110	20	20	60	50	20	110	0
CO1	60	0	0	0	80	0	30	30	20	30	30	20
CO2	200	0	70	40	170	40	50	130	80	50	130	80
CO3	130	80	70	40	90	120	60	140	40	60	140	40
CO4	60	0	0	0	60	20	30	30	0	30	30	0
CO5	30	30	0	0	40	0	30	0	0	30	0	0

Table 6. Computational Statistics Related to Problem Size^a

example ID	model P (MINLP)				model PR with PMCR (MILP)				model PR with NMDT (MILP)			
	no. of eqs	no. of vars	no. of bins	no. of BL terms	NP	no. of eqs	no. of vars	no. of bins	NP	no. of eqs	no. of vars	no. of bins
SC1-TP1, SC2-TP1, SC3-TP1	1505	1237	12	370	4	5804	3485	596	10	7806	5797	1472
					20	17388	11613	2932	100	12488	9417	2932
					24	20284	13645	3516	1000	17170	13037	4392
SC1-TP3, SC2-TP3, SC3-TP3, SC4-TP3	4531	3727	36	1294	4	19104	11315	1980	10	26062	19179	4896
					20	58336	38707	9756	100	42088	31439	9756
					24	68144	45555	11700	1000	58114	43699	14616

^aEquations = equations, vars = total number of variables, bins = binary variables, BL terms = bilinear terms.

Table 7. Statistics Related to Computational Performance

algorithm		example ID							
		SC1-TP1	SC2-TP1	SC3-TP1	SC1-TP3	SC2-TP3	SC3-TP3	SC4-TP3	
ANTIGONE 1.1	solution	55574	49885	53806	55572	49860	53796	55322	
	best LB	55569	49880	53801	55045	49717	53200	54931	
	opt gap (%)	0.01	0.01	0.01	0.95	0.29	1.11	0.71	
	total time (s)	333	240	83	10800	10800	10800	10800	
BARON 15.9	solution	55574	49885	53806	55568	49861	53797	55330	
	best LB	55304	49592	53477	54246	49202	52369	54031	
	opt gap (%)	0.49	0.59	0.61	2.38	1.32	2.65	2.35	
	total time (s)	10800	10800	10800	10800	10800	10800	10800	
new with PMCR	solution	55574	49885	53806	55567	49860	53789	55322	
	best LB	55434	49822	53667	55126	49725	53269	54958	
	opt gap (%)	0.25	0.13	0.26	0.79	0.27	0.97	0.66	
	total time (s)	2993	1311	2896	8727	10800	8780	8547	
new with NMDT	iterations l	7	7	7	7	8	7	7	
	NLP solves	153	107	155	134	172	153	110	
	OBBT calls	1	1	1	1	2	1	1	
	OBBT time (s)	339	325	326	1571	3240	1656	1483	
	solution	55574	49885	53806	55572	49860	53796	55322	
	best LB	55508	49856	53711	55102	49727	53206	54912	
	opt gap (%)	0.12	0.06	0.18	0.85	0.27	1.10	0.74	
	total time (s)	2717	1735	2750	8752	10800	5682	5394	
	iterations l	4	4	4	5	6	4	4	
	NLP solves	88	58	96	79	115	65	28	
OBBT calls	1	1	1	2	3	1	1		
OBBT time (s)	311	313	312	3133	4789	1629	1575		

one-time-period problems. In these examples, the first upper bound computed by all these methods is actually the global optimal solution.

Our algorithm starts with 4 partitions per discretized variable with PMCR and 10 for NMDT. It is thus no surprise that the initial lower bounds from NMDT are better, corresponding to

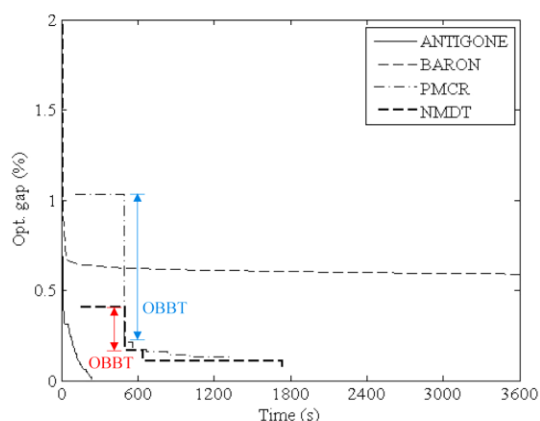


Figure 8. Optimality gap vs time for example SC2-TP1 (scenario SC2 with one time period).

optimality gaps that are on average 2.22 times lower than those for PMCR. Most MILP relaxation problems **PR** could be solved to optimality in less than 1000 s, the exceptions occurring in the last iterations (NP = 24 for PMCR and NP = 1000 for NMDT). Optimality-based bound tightening (OBBT) was applied in the first iteration ($l = 1$) and typically resulted in a larger reduction in optimality gap than the one obtained from the initial PMCR relaxation problem. This is why the discretization is not increased in the second iteration of the algorithm. Because the first upper bound was the optimal solution, OBBT was applied only once. The execution times for PMCR and NMDT were similar (average difference is 282 s) with the final optimality gap for NMDT being roughly half the one for PMCR; see Table 7.

Concerning the comparison to the commercial global solvers, ANTIGONE solves the problems to global optimality (opt gap < 0.01%), being thus faster than the other algorithms. As for BARON, the lower bound improves very slowly after the initial 100 s, leading to larger gaps at termination. We can thus conclude that our new algorithm performs better than BARON for the single-period problems but worse than ANTIGONE. Based on the log file generated by ANTIGONE for example SC2-TP1, shown in Table 8, one possible explanation might be that the inclusion of cutting planes works better than the further discretization of bilinear terms.

7.2. Three-Time-Period Problems. Figure 9 shows the optimality gap vs time computed for example SC1-TP3, which is representative of the three-time-period problems. In these examples, the first upper bound computed by the algorithms is in general different and may not be the global optimal solution because none can solve them to the specified tolerance $\varepsilon = 0.01\%$.

One important result from Table 7 is that our new algorithm using PMCR could find better solutions for problems SC1-TP3, SC3-TP3, and SC4-TP3 than at least one of the commercial solvers up to the maximum wall time limit of 3 h. Furthermore, the quality of the lower bound is better for all problems leading to slightly lower optimality gaps than ANTIGONE and at least 2 times smaller than BARON.

The reason the PCMR relaxation works better than NMDT is because (i) problem **PR** can only be solved to optimality in less than 1000 s for 4 partitions, leading to better lower bounds than those found with more partitions up to the same

Table 8. Algorithm Attributes for Example SC2-TP1 (Scenario SC2 with One Time Period)

	algorithm		
	ANTIGONE	PMCR	NMDT
nodes explored ^a	0	0	0
nodes remaining ^a	0	0	0
maximum tree depth ^a	0	0	0
cutting planes ^a	420	0	0
total time (CPU s)	240	1311	1735
preprocessing ^a	1	0	0
solving MILP relaxations	84	709	1246
searching for feasible solutions	15	249	161
variable bounds tightening	139	325	313
OBBT	130	325	313
FBBT ^{a,b}	78	0	0
branching ^a	0	0	0

^aNot applicable for our proposed global algorithm. ^bFBBT = feasibility-based bound tightening.

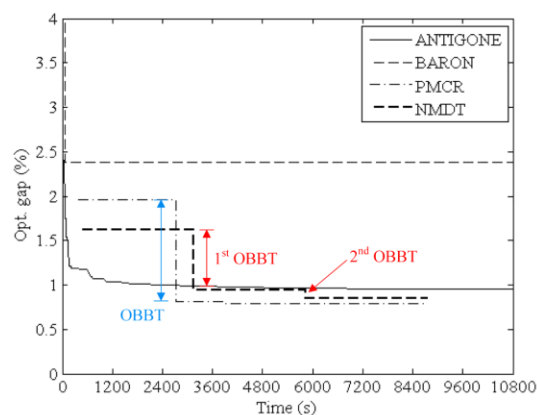


Figure 9. Optimality gap vs time for example SC1-TP3 (scenario SC1 with three time periods).

computational time, and (ii) it involves more iterations, leading to the solution of a larger number of NLP instances **PF** and ultimately to a better upper bound. A better upper bound reduces the feasible space of problems **PRBmin** and **PRBmax**, leading to a tighter domain of the model variables. Notice in Table 7 that in contrast to the single-period problems, OBBT is sometimes applied more than once.

Returning to the comparison with ANTIGONE, it is interesting to observe that this commercial solver significantly improves the lower bound in the first 1200 s, but after that, the gap is slowly reduced. On the basis of the log file generated by ANTIGONE for example SC1-TP3 and shown in Table 9, it seems that adding cutting planes does not represent a major benefit and that starting with a low number of partitions in PMCR, so as to have a tight lower bound and generate a better upper bound that further reduces the domain of the variables involved in bilinear terms through OBBT, is more important than exploring further increments in the number of partitions with NMDT (because increasing the number of partitions originated very large MILP problems that were difficult to solve).

Using GAMS parallel grid computing option decreased the time required to solve the NLP instances **PF** resulting from one

Table 9. Algorithm Attributes for Example SC1-TP3 (Scenario SC1 with Three Time Periods)

	algorithm		
	ANTIGONE	PMCR	NMDT
nodes explored ^a	0	0	0
nodes remaining ^a	1	0	0
maximum tree depth ^a	0	0	0
cutting planes ^a	4000	0	0
total time (CPU s)	10800	8727	8752
preprocessing ^a	1	0	0
solving MILP relaxations	926	6371	5032
searching for feasible solutions	350	707	507
variable bounds tightening	9519	1571	3133
OBBT	9471	1571	3133
FBBT ^{a,b}	1256	0	0
branching ^a	0	0	0

^aNot applicable for our proposed global algorithm. ^bFBBT = feasibility-based bound tightening.

iteration of the 1-time-period problems from 16 to 1.6 s, and from 54 to 4 s for the three-time-period problems. Regarding execution times for each call of OBBT, parallelization reduced times from 10 to 0.8 s and from 32 to 1.3 s for single- and three-time-period problems, respectively. Note that all execution times reported here are wall times, including those required by GAMS to generate the model and transfer information to/from the solvers.

7.3. Refinery Plan: Single-Period vs Multiple-Period Solution. The results in Table 7 show that the cost of the refinery plan computed by our proposed model for a given scenario can be lower with three time periods. The exception occurs for SC4-TP3, for which the best found solution of 55322 (can only improve to 54958) is worse than 53806, the optimal value for SC3-TP1 (recall that SC3-TP1 is the corresponding single-period problem of SC4-TP3). It tells us that, for our specific formulation of the planning problem, neither the single- nor the multiple-period problem is a relaxation of its counterpart, which deserves further explanation. The explanation will be based on two problem features: supply/demand constraints and intermediate storage tanks.

When dividing the weekly supply/demand between multiple time periods, we add more constraints to the multiple-period problem, causing the objective function value to degrade. It is the predominant effect if demand requirements are not evenly distributed along the planning horizon, especially if the majority of the demand occurs in the initial periods. Comparing the data between SC3-TP3 and SC4-TP3 in Table 5, one can see that part of the demand from days 5–7 has been anticipated, which caused the cost to increase above SC3-TP1. However, this trend of the objective function value worsening with the introduction of more demand constraints is not always observed in our proposed refinery planning model due to the consideration of intermediate storage tanks as blending tanks and upstream units being able to change their operating modes (and thus the quality of their outlet streams). Under certain demand patterns, usually those evenly distributed along the horizon, the effect of blending tanks will be observed as discussed in the next paragraph.

Blending tanks allow for a more flexible operation of the network because their quality properties can be changed from one period to the next. Recall that in our proposed model (eq 31), the quality of a tank at the end of a time period

depends on the quality of the inlet stream and that of the tank heel (i.e., the inventory level at the beginning of the time period). Quality of the blending tanks' inlet streams is affected by the operation of upstream units. For a single-period model, there will only be a set of operating modes for the entire horizon. Moreover, for a single period, the tank heel qualities are fixed to their initial known values, leading to fewer degrees of freedom. This is the predominant effect in the cost trend SC2-TP3 < SC2-TP1 and SC3-TP3 < SC3-TP1. The same trend has been observed when the classical crude oil blending problems³³ are solved to global optimality,^{34,35} in which crude supply and CDU demand constraints are for the given time horizon (single period) but multiple periods of operation are allowed.

Although planning models with multiple time periods may lead to more accurate solutions, they generate mathematical problems with more equations, variables, and nonlinear terms; therefore, higher computational effort is required to compute a global optimal solution. This is one of the motivations to develop more efficient global optimization algorithms, as well as better planning models.

7.4. Comparison of Production Plans. We present the single-period and multiple-period solutions for scenario SC2 computed by our algorithm using NMDT. In this scenario, the duration of time period 1 is 1 day, duration of period 2 is 4 days, and duration of period 3 is 2 days. The volumes processed by the major units are shown in Figure 10, the crude

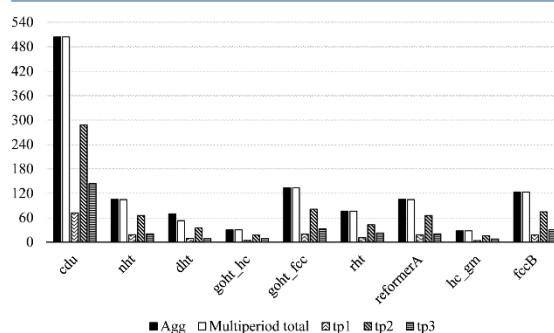


Figure 10. Feed to the major units (in kbbl, examples SC2-TP1 and SC2-TP3).

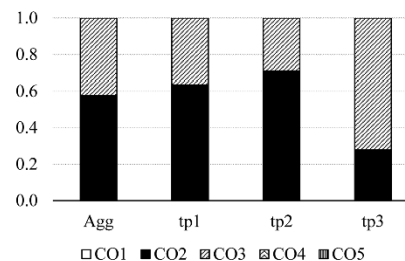


Figure 11. Crude mix recipe of the feed to the CDU (examples SC2-TP1 and SC2-TP3).

mix recipe of the feed to the CDU in Figure 11, and the blend recipes for the finished products in Figure 12. The total amount of each crude oil fed to the CDU, and the total amount blended of each product, along the planning horizon, is the same in the single-period and the multiple-period solutions. However, in the multiple-period solution, product diesel D1 is

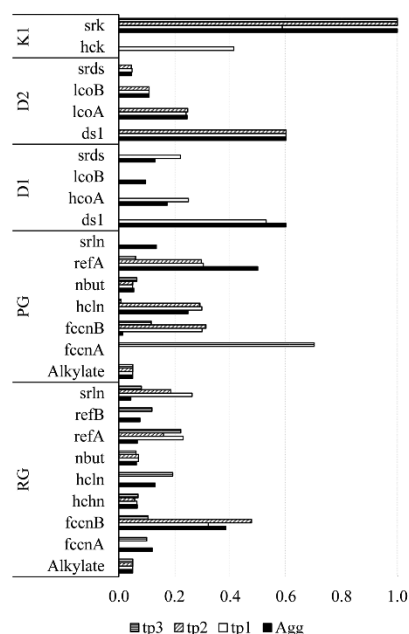


Figure 12. Blend recipes (examples SC2-TP1 and SC2-TP3).

only blended during period 1, and diesel D2 only during periods 1 and 2.

Regarding volumes processed by major units, the significant difference between the single-period and the multiple-period solution is found in the diesel hydrotreating unit “dht” (Figure 10). Because in the multiple-period solution diesel products are not blended during period 3, “dht” does not process as much material as in the single-period solution, nor does it operate with the same sulfur removal severity. Because “dht” is processing less material during period 3, the crude mix fed to the CDU changes significantly in this period to account for the sulfur content downstream: it uses more crude oil CO3 (Figure 11), which is the most valuable and has lower sulfur content. One of the reasons only the high quality and most expensive crude oils (CO2 and CO3) are used is the formulation of the objective function. We penalize the supply streams of crude oils, but these are fixed in our examples. We do not penalize the streams going into the crude distillation unit. Therefore, the model does not see the difference between using expensive and cheap materials. Other reasons are that we do not specify a fixed crude oil mix recipe, we do not force consumption of the arriving lots of crude oil, and there is enough storage to keep material in the crude oil tanks.

Differences in blend recipes are difficult to analyze in these examples because the initial inventories of blend components and finished products are above the minimum limits and there exist the possibility of multiple optimal blend recipes.

8. CONCLUSIONS

In this work, we introduced a new global optimization algorithm for nonconvex bilinear problems and applied it successfully to refinery planning problems with one and three time periods. The algorithm exploits the strengths of different relaxation approaches, ranging from those adding further binary variables to the problem, like piecewise McCormick and multiparametric

disaggregation, to the simplest standard McCormick relaxation involved in optimality-based bound tightening of all the variables participating in bilinear terms. Another novel aspect has been to rely on the solution pool option of CPLEX to work as a multiple starting point heuristic and increase the likelihood of finding the global optimal solution. The different instances of the original problem were solved in parallel and so was the bound tightening stage. Parallelization was found to substantially reduce the execution time. In addition to bound tightening, the lower bound moves toward the optimum by increasing the number of partitions in the piecewise relaxation.

Through the solution of a case study for seven different demand/supply scenarios, we have shown that our algorithm performs better than state-of-the-art commercial solver BARON, and slightly better than ANTIGONE when dealing with multiple time periods. For large-size problems, we observed that increasing the number of partitions beyond just a few leads to the generation of large MILPs that may prevent computing good estimates of the lower bound. As a consequence, the major improvements in optimality gap came after applying optimality-based bound tightening and resolving the MILP with the same number of partitions.

Using piecewise relaxation methods in optimality-based bound tightening will further reduce the variables domain but will also increase execution times; therefore, if the allocated time to solve the problem is large enough, such alternative could be fruitful. In this respect, developing and improving variable bound tightening methods (either optimality- or feasibility-based) will lead to more efficient deterministic global optimization algorithms. Future work will also involve a more thorough comparative study with commercial solvers, including different formulations of the refinery planning problem and other classes of MINLPs of the bilinear type.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.iecr.6b01350.

Complete mathematical models and tables with all the necessary data (supply and demand; quality; fixed values for quality properties; fixed yields; sulfur removal factor; storage tank quality; feed flow rates; inventory levels; and tank, mixer, and unit subsets) for examples presented in this work; complete list of nomenclature used (PDF)

■ AUTHOR INFORMATION

Corresponding Author

*P. M. Castro. E-mail: pmcastro@fc.ul.pt.

ORCID

Pedro M. Castro: 0000-0002-4898-8922

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

Support by Ontario Research Foundation, McMaster Advanced Control Consortium, and Fundação para a Ciência e Tecnologia, through the Investigador FCT 2013 program and project UID/MAT/04561/2013, is gratefully acknowledged. We thank Jeff Kelly for proposing the use of DIA software to create graphical representation of the plant and generate model equations. Jeff Kelly provided us with a prototype Python code

and DIA objects that generate a text file with the information entered in the DIA canvas (as is done in the software developed by Industrial Algorithms). We modified that Python script and created new DIA objects to create a GAMS file with the corresponding mathematical models.

NOMENCLATURE

Sets and Indices

$B = \{b\}$	blenders
$N = \{n\}$	time periods
$Q = \{q\}$	quality properties (e.g., motor octane number)
$RU = \{ru\}$	real units that have different operating modes
$S = \{s\}$	streams (note that s' is an alias of index s)
$T = \{t\}$	tanks
$U = \{u\}$	units, not including blenders nor tanks

Subsets

$BI = \{(b, s)\}$	inlet streams of blender b
$BO = \{(b, s)\}$	outlet streams of blender b
$CDU = \{u\}$	crude distillation units
$DST = \{s\}$	destination/demand streams
$HTU = \{u\}$	hydrotreating units
$MMU = \{u\}$	units that represent one operating mode of a real unit with multiple operating modes

$PU = \{u\}$ processing units that are not mixers, splitters, hydrotreaters, or crude distillation units

$QLV = \{q\}$ quality properties that blend linearly on a volumetric basis

$QLW = \{q\}$ quality properties that blend linearly on a weight basis

$RUU = \{(ru, u)\}$ real unit ru has operating modes represented by units u

$SP = \{u\}$ splitters

$SQ = \{(s, q)\}$ quality properties that are important to track in stream s

$SQB = \{(s, q)\}$ quality properties of stream s that are included in blending calculations

$SRC = \{s\}$ source streams

$SMU = \{u\}$ units representing the single operating mode of a real unit

$TI = \{(t, s)\}$ inlet streams of tank t

$TO = \{(t, s)\}$ outlet streams of tank t

$UI = \{(u, s)\}$ inlet streams of unit u

$UIR = \{(u, s)\}$ inlet streams of unit u minus one of them

$UO = \{(u, s)\}$ outlet streams of unit u

$UQI = \{(u, q)\}$ quality property q in the inlet stream of unit u is constrained to be within a minimum and/or maximum value

Parameters

$A_{s,n}^{\max}$	maximum availability of stream s during period n
$A_{s,n}^{\min}$	minimum availability of stream s during period n
BMP_b	maximum number of distinct products that blender b can produce in a given time period
$BR_{s,s'}^{\max}$	maximum volume fraction allowed of component s in blended product s'
$BR_{s,s'}^{\min}$	minimum volume fraction allowed of component s in blended product s'
CS_s	cost of stream s
CU_u	cost of operating unit u
$D_{s,n}^{\max}$	maximum demand of stream s during period n
$D_{s,n}^{\min}$	minimum demand of stream s during period n

IC_t	cost of one volumetric unit of stored material in tank t
L_n	length (duration) of time period n
PS_s	selling price of stream s
$QS_{s,q,n}^{\text{fix}}$	known quality value of property q in stream s during period n
$QS_{s,q}^{\min}$	minimum quality specification for property q in stream s
$QS_{s,q}^{\max}$	maximum quality specification for property q in stream s
$QSUI_{s,q}^{\min}$	minimum value for quality property q in stream s , which is an inlet stream to a unit
$QSUI_{s,q}^{\max}$	maximum value for quality property q in stream s , which is an inlet stream to a unit
$QCO_{q,s,s'}^{\text{fix}}$	quality value of property q in distillation cut s' when feeding CDU with crude oil s
UMM_{ru}	maximum number of modes in which real unit ru can operate during a time period
$VB_{s,n}^{\min}$	minimum amount to blend of product s if the decision to blend it is made
VBR_b^{\max}	maximum blending rate of blender b
VBR_b^{\min}	minimum blending rate of blender b
VI_t^{ini}	initial inventory level of tank t
VI_t^{\max}	maximum capacity of tank t
VI_t^{\min}	minimum inventory level of tank t
VTR_t^{\max}	maximum withdrawal rate from tank t
VTR_t^{\min}	minimum withdrawal rate from tank t
$VU_{u,n}^{\min}$	minimum volume to be processed by unit u in period n
VUR_u^{\max}	maximum processing rate of unit u
VUR_u^{\min}	minimum processing rate of unit u
$VUTR_{ru}^{\max}$	maximum processing rate of real unit ru
$VUTR_{ru}^{\min}$	minimum processing rate of real unit ru
$YCDU_{s,s'}^{\text{fix}}$	yield of cut s' from crude oil s (volume fraction)

Binary Variables

$bvb_{s,n}$ if equal to 1, then product s is blended during time period n

$bvu_{u,n}$ if equal to 1, then unit u (which is the representation of an operating mode of a real unit ru) operates during time period n

Continuous Variables

CDU cost	cost associated with operation of crude distillation units
$CMR_{s,n}$	crude mix recipe (volume fraction of stream s that constitutes feed of CDU during period n)
HT cost	cost associated with operation of hydrotreaters
inventory cost	cost associated with inventory levels
materials cost	cost of raw materials
profit	profit
PU cost	cost associated with operation of processing units
$QS_{s,q,n}$	value of quality property q in stream s during period n
$QVBC_{s,s',q,n}$	product of $VBC_{s,s',n}$ and $QS_{s,q,n}$
$QMBC_{s,s',q,n}$	product of $VBC_{s,s',n}$ and $QS_{s,q,n}$ and specific gravity
$QVS_{s,q,n}$	quality property q in stream s times volume of such stream during period n
$QVT_{t,q,n}$	quality q times inventory of tank t at the end of period n
revenue	revenue

RSUL _{u,n}	amount of sulfur removed by unit <i>u</i> during period <i>n</i>
TB _{s,n}	estimated time spent to blend product <i>s</i> during period <i>n</i>
total cost	total cost
TU _{u,n}	estimated processing time required by unit <i>u</i> during period <i>n</i>
VB _{s,n}	volume blended of stream <i>s</i> (finished product) during period <i>n</i>
VBC _{s,s',n}	volume from stream <i>s</i> (blend component) to stream <i>s'</i> (finished product) during period <i>n</i>
VS _{s,n}	volume through stream <i>s</i> during period <i>n</i>
VT _{t,n}	inventory level in tank <i>t</i> at the end of period <i>n</i>
VU _{u,n}	volume processed by unit <i>u</i> during period <i>n</i>
Z	variable to be minimized or maximized

REFERENCES

- Jia, Z.; Ierapetritou, M. G. Efficient Short-Term Scheduling of Refinery Operations Based on a Continuous Time Formulation. *Comput. Chem. Eng.* **2004**, *28*, 1001.
- Mendez, C. A.; Grossmann, I. E.; Harjunkski, I.; Kabore, P. A Simultaneous Optimization Approach for Off-Line Blending and Scheduling of Oil-Refinery Operations. *Comput. Chem. Eng.* **2006**, *30*, 614.
- Bengtsson, J.; Nonas, S. L. Refinery Planning and Scheduling: An Overview. In *Energy, Natural Resources and Environmental Economics*; Bjørndal, E., et al., Eds.; Springer-Verlag Berlin: Heidelberg, Germany, 2010; pp 115–130.
- Shah, N. K.; Li, Z.; Ierapetritou, M. G. Petroleum Refining Operations: Key Issues, Advances, and Opportunities. *Ind. Eng. Chem. Res.* **2011**, *50*, 1161.
- Moro, L. F. L.; Zanin, A. C.; Pinto, J. M. A Planning Model for Refinery Diesel Production. *Comput. Chem. Eng.* **1998**, *22*, S1039.
- Neiro, S. M. S.; Pinto, J. M. A General Modelling Framework for the Operational Planning of Petroleum Supply Chains. *Comput. Chem. Eng.* **2004**, *28*, 871.
- Li, W.; Hui, C. W.; Li, A. Integrating CDU, FCC and Product Blending Models into Refinery Planning. *Comput. Chem. Eng.* **2005**, *29* (9), 2010.
- Alhajri, I.; Elkamel, A.; Albahri, T.; Douglas, P. L. A Nonlinear Programming Model for Refinery Planning and Optimisation with Rigorous Process Models and Product Quality Specifications. *Int. J. Oil, Gas Coal Technol.* **2008**, *1* (3), 283.
- Elkamel, A.; Ba-Shammakh, M.; Douglas, P.; Croiset, E. An Optimization Approach for Integrating Planning and CO₂ Emission Reduction in the Petroleum Refining Industry. *Ind. Eng. Chem. Res.* **2008**, *47*, 760–776.
- Alattas, A. M.; Grossmann, I. E.; Palou-Rivera, I. Refinery Production Planning: Multiperiod MINLP with Nonlinear CDU Model. *Ind. Eng. Chem. Res.* **2012**, *51* (39), 12852.
- Zhang, B. J.; Liu, K.; Luo, X. L.; Chen, Q. L.; Li, W. K. A Multi-Period Mathematical Model for Simultaneous Optimization of Materials and Energy on the Refining Site Scale. *Appl. Energy* **2015**, *143*, 238–250.
- Pongsakdi, A.; Rangsunvigit, P.; Siemanond, K.; Bagajewicz, M. J. Financial Risk Management in the Planning of Refinery Operations. *Int. J. Prod. Econ.* **2006**, *103*, 64–86.
- Kuo, T. H.; Chang, C. T. Application of a Mathematic Programming Model for Integrated Planning and Scheduling of Petroleum Supply Networks. *Ind. Eng. Chem. Res.* **2008**, *47*, 1935–1954.
- D'Ambrosio, C.; Lodi, A. Mixed Integer Nonlinear Programming Tools: A Practical Overview. *J. Oper. Res.* **2011**, *9*, 329–349.
- Floudas, C. A.; Gounaris, C. E. A Review of Recent Advances in Global Optimization. *J. Glob. Optim.* **2009**, *45*, 3–38.
- Sherali, H. D.; Alameddine, A. A New Reformulation-Linearization Technique for Bilinear Programming Problems. *J. Glob. Optim.* **1992**, *2* (4), 379–410.
- Quesada, I.; Grossmann, I. E. Global Optimization of Bilinear Process Networks with Multicomponent Flows. *Comput. Chem. Eng.* **1995**, *19* (12), 1219.
- Androulakis, I. P.; Maranas, C. D.; Floudas, C. A. α BB: A Global Optimization Method for General Constrained Nonconvex Problems. *J. Glob. Optim.* **1995**, *7* (4), 337–363.
- Adjiman, C. S.; Dallwig, S.; Floudas, C. A.; Neumaier, A. A Global Optimization Method, α BB, for General Twice-Differentiable Constrained NLPs-I. Theoretical Advances. *Comput. Chem. Eng.* **1998**, *22* (9), 1137–1158.
- McCormick, G. P. Computability of Global Solutions to Factorable Nonconvex Programs: Part I—Convex Underestimating Problems. *Mathematical programming* **1976**, *10* (1), 147.
- Misener, R.; Floudas, C. A. Advances for the Pooling Problem: Modeling, Global Optimization, and Computational Studies. *Appl. Comput. Math.* **2009**, *8* (1), 3–22.
- Misener, R.; Floudas, C. A. Global Optimization of Large-Scale Generalized Pooling Problems: Quadratically Constrained MINLP Models. *Ind. Eng. Chem. Res.* **2010**, *49*, 5424–5438.
- Andrade, T.; Ribas, G.; Oliveira, F. A Strategy Based on Convex Relaxation for Solving the Oil Refinery Operations Planning Problem. *Ind. Eng. Chem. Res.* **2016**, *55*, 144–155.
- Faria, D. C.; Bagajewicz, M. J. A New Approach for Global Optimization of a Class of MINLP Problems with Applications to Water Management and Pooling Problems. *AIChE J.* **2012**, *58*, 2320–2335.
- Castro, P. M. Tightening Piecewise McCormick Relaxations for Bilinear Problems. *Comput. Chem. Eng.* **2015**, *72*, 300.
- Misener, R.; Thompson, J. P.; Floudas, C. A. APOGEE: Global Optimization of Standard, Generalized, and Extended Pooling Problems Via Linear and Logarithmic Partitioning Schemes. *Comput. Chem. Eng.* **2011**, *35*, 876–892.
- Kolodziej, S.; Castro, P. M.; Grossmann, I. E. Global Optimization of Bilinear Programs with a Multiparametric Disaggregation Technique. *J. Glob. Optim.* **2013**, *57* (4), 1039.
- Kolodziej, S. P.; Grossmann, I. E.; Furman, K. C.; Sawaya, N. W. A Discretization-Based Approach for the Optimization of the Multiperiod Blend Scheduling Problem. *Comput. Chem. Eng.* **2013**, *53*, 122–142.
- Castro, P. M.; Grossmann, I. E. Optimality-Based Bound Contraction with Multiparametric Disaggregation for the Global Optimization of Mixed-Integer Bilinear Problems. *J. Glob. Optim.* **2014**, *59* (2–3), 277.
- Castro, P. M. Normalized Multiparametric Disaggregation: An Efficient Relaxation for Mixed-Integer Bilinear Problems. *J. Glob. Optim.* **2016**, *64*, 765–784.
- Tawarmalani, M.; Sahinidis, N. V. A Polyhedral Branch-And-Cut Approach to Global Optimization. *Math. Program.* **2005**, *103* (2), 225.
- Misener, R.; Floudas, C. A. ANTIGONE: Algorithms for Continuous/Integer Global Optimization of Nonlinear Equations. *J. Glob. Optim.* **2014**, *59* (2–3), 503.
- Lee, H.; Pinto, J. M.; Grossmann, I. E.; Park, S. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Ind. Eng. Chem. Res.* **1996**, *35*, 1630–1641.
- Castro, P. M.; Grossmann, I. E. Global Optimal Scheduling of Crude Oil Blending Operations with RTN Continuous-time and Multiparametric Disaggregation. *Ind. Eng. Chem. Res.* **2014**, *53*, 15127–15145.
- Castro, P. Source-based discrete and continuous-time formulations for the crude oil pooling problem. *Comput. Chem. Eng.* **2016**, *93*, 382–401.

Chapter 8: Global Optimization of Nonlinear Blend-Scheduling Problems

This chapter has been published in the Engineering Journal (open access). Complete citation:

Castillo Castillo, P. A., Castro, P. M., & Mahalec, V. (2017). Global optimization of nonlinear blend-scheduling problems. *Engineering*, 3(2), 188–201. *Elsevier Ltd.*, doi: 10.1016/J.ENG.2017.02.005

In Chapter 8, the heuristic and rigorous optimization techniques from the previous two Chapters are used to solve nonlinear blend-scheduling problems. MPIP-C is faster computing feasible near-optimal solutions than the global optimization method. The lower bound on the blend cost computed by MPIP-C is larger than the initial one computed by the global optimization algorithm. These results show the importance of the MPIP-C technique and how it can improve a deterministic global optimization algorithm.



Research
Smart Process Manufacturing—Article

Global Optimization of Nonlinear Blend-Scheduling Problems

Pedro A. Castillo Castillo^a, Pedro M. Castro^b, Vladimir Mahalec^{a,*}

^a Department of Chemical Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada

^b Center for Mathematics, Fundamental Applications and Operations Research, Faculty of Sciences, University of Lisbon, Lisbon 1749-016, Portugal

ARTICLE INFO

Article history:

Received 7 December 2016

Revised 16 February 2017

Accepted 20 February 2017

Available online 28 March 2017

Keywords:

Global optimization

Nonlinear gasoline blending

Continuous-time scheduling model

Piecewise linear relaxations

ABSTRACT

The scheduling of gasoline-blending operations is an important problem in the oil refining industry. This problem not only exhibits the combinatorial nature that is intrinsic to scheduling problems, but also non-convex nonlinear behavior, due to the blending of various materials with different quality properties. In this work, a global optimization algorithm is proposed to solve a previously published continuous-time mixed-integer nonlinear scheduling model for gasoline blending. The model includes blend recipe optimization, the distribution problem, and several important operational features and constraints. The algorithm employs piecewise McCormick relaxation (PMCR) and normalized multiparametric disaggregation technique (NMDT) to compute estimates of the global optimum. These techniques partition the domain of one of the variables in a bilinear term and generate convex relaxations for each partition. By increasing the number of partitions and reducing the domain of the variables, the algorithm is able to refine the estimates of the global solution. The algorithm is compared to two commercial global solvers and two heuristic methods by solving four examples from the literature. Results show that the proposed global optimization algorithm performs on par with commercial solvers but is not as fast as heuristic approaches.

© 2017 THE AUTHORS. Published by Elsevier LTD on behalf of the Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Computing and implementing an optimal production schedule can reduce operational costs, increase profit margins, and avoid deviations from environmental constraints [1]. However, complex industrial plants can have multiple production, storage, and distribution subsystems, several distinct raw materials and intermediate and final products, and intricate connections between all these elements that make scheduling a difficult decision-making process.

Scheduling problems typically deal with four main decisions [1]: ① determining the required tasks to fulfill the corresponding objectives, requirements, and/or demand targets; ② assigning each task to a processing unit or resource that is available in the network; ③ defining the sequence in which the tasks will be executed; and ④ timing the tasks—that is, determining when to start and stop each one (Fig. 1). Optimal scheduling decisions are those that max-

imize or minimize a desired objective such as profit, total cost, lead time, and so forth. Scheduling software and tools based on mathematical programming is becoming more usual in practice.

The scheduling of gasoline-blending operations is an important and relevant industrial problem because gasoline accounts for 60%–70% of the total profit of an oil refinery [2–4]. In a gasoline-blending system, components from dedicated supply tanks are mixed in blending tanks or in-line blenders and sent to product tanks. Blending tanks can either have flow in or out (Fig. 2), resembling batch operation. In contrast, in-line blenders operate in a continuous manner (Fig. 3). In addition to the four decisions mentioned earlier, scheduling blend operations should also involve determining the blend recipes—that is, the amounts of components to mix such that products' quality properties meet given specifications.

The gasoline-blending system studied in this work is described in Fig. 3. Gasoline blending is carried out by one or more continuous

* Corresponding author.

E-mail address: mahalec@mcmaster.ca

<http://dx.doi.org/10.1016/j.eng.2017.02.005>

2095-8099/© 2017 THE AUTHORS. Published by Elsevier LTD on behalf of the Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

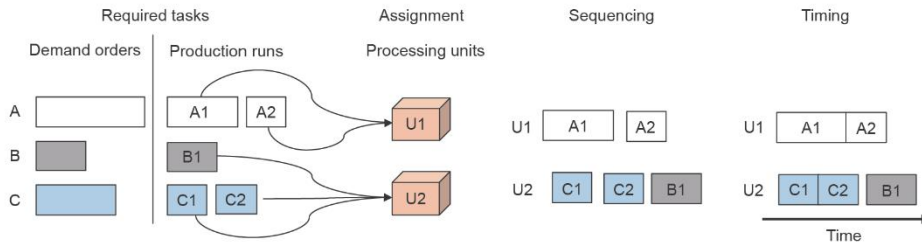


Fig. 1. Main scheduling decisions.

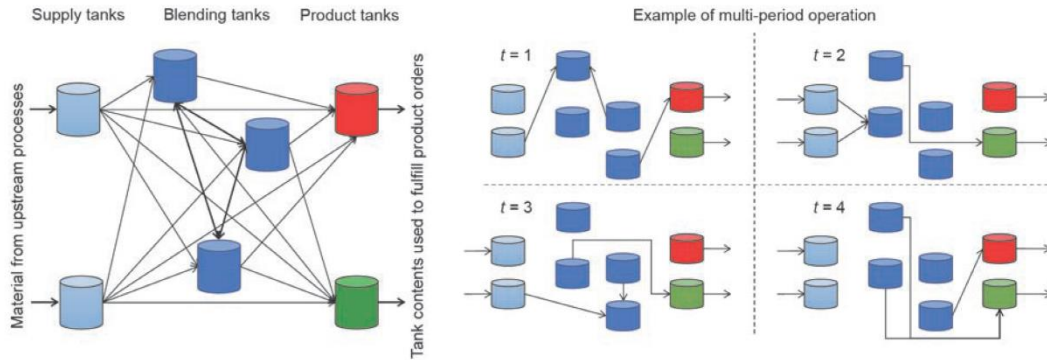


Fig. 2. Batch-blending system. The variable t means time period.

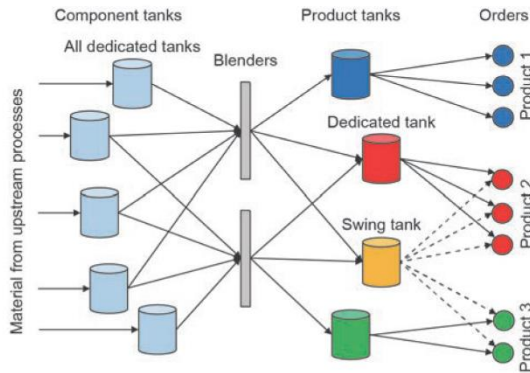


Fig. 3. General scheme of the continuous gasoline-blending system studied in this work.

blenders. Each blender is connected to the sources of blend components. Blended material in some refinery configurations goes to a storage tank, while in other configurations, it can go directly to the pipeline. Since there are several grades of gasoline produced (e.g., regular, medium, premium), the blender switches from blending one grade to another. Each switch requires (partial) realignment of blend feeding lines, which leads to a loss of blending capacity. In addition, switching to a different range of quality properties often requires resetting or recalibration of the analyzers used to measure them.

Some of the most important quality properties of gasoline are research octane number (RON), motor octane number (MON), Reid vapor pressure (RVP), density, sulfur, aromatics (Ar), and olefins (OI) content. RON, MON, and RVP do not blend linearly; thus, considering nonlinear blending rules for such quality properties in the scheduling model can increase the accuracy of the solution and reduce

quality giveaways [4,5].

Since the 1960s, there has been a significant effort to derive so-called “blend indices.” These are nonlinear transformations of the actual quality properties of the blend components, which then, as a linear combination, can predict the quality of the blend. Even with this approach, two issues remain:

- (1) If a product is blended into a tank, there is always some material in the tank (the so-called “tank heel”) left over from previous blends. Any new blend must include the properties of the material in the tank in the calculation of the new blend recipe, which leads to a nonlinear blending model for multi-period scheduling, even when using blend indices.
- (2) Blend properties calculated from these indices are not 100% accurate, and a cumulative annual quality giveaway of, for example, octane can amount to very large cost increments. This forces the use of nonlinear blending models for the calculation of, for example, octane numbers.

Mathematical models for scheduling problems are usually formulated as mixed-integer linear programming (MILP). However, for gasoline blending, nonlinear behavior is intrinsic to the corresponding process and mixed-integer nonlinear programming (MINLP) needs to be employed for the sake of accuracy. Most nonlinear terms are non-convex, making convex optimization techniques ineffective. A global optimization approach is thus required. Before describing the proposed global optimization method, a brief review of previous work is presented in the following paragraphs.

1.1. Literature review on refinery scheduling

Scheduling models can be divided into two main categories based on the treatment of the time domain: discrete- and continuous-time formulations. In discrete-time models, the time horizon is divided into several time periods of known duration with fixed start and end time. In continuous-time models, the time horizon is partitioned into time slots whose duration will be determined by the

optimization. While continuous-time models generate problems with fewer discrete variables than their discrete counterparts, they are more complex to formulate and often feature many “big-M” constraints that, due to their weak relaxations [6], compromise computational performance. More in-depth reviews of scheduling formulations can be found in Refs. [1,7–9].

Gasoline blending has been studied by many researchers due to its commercial importance and non-convex features, which makes it a suitable subject for testing different formulations and algorithmic approaches. Operational constraints found in gasoline blending are related to the presence of multipurpose tanks and non-identical blenders, to different storage-tank policies (e.g., whether the simultaneous receipt and delivery of material is allowed or not), and to practical aspects such as minimum blend sizes and minimum blender running and setup times. Not all of these constraints are considered in published scheduling models. In some cases, blend recipes are assumed to be fixed (i.e., they cannot be optimized). The downstream distribution or shipping problem (i.e., timing delivery tasks to fulfill the demand) is sometimes also part of the blend-scheduling problem.

Méndez et al. [2] presented both a discrete- and continuous-time MILP model to schedule gasoline-blending operations. An iterative method was employed to handle nonlinear blending rules while preserving the linearity of the models. Several key operational constraints were omitted and the distribution problem was not considered.

Jia and Ierapetritou [10] developed a continuous-time MILP model to simultaneously schedule gasoline-blending tasks and distribution operations. The linearity of the model was maintained by using given preferred recipes. Their model was later extended to schedule operations of the main processing units in an oil refinery [11].

Glismann and Gruhn [12] used a two-level approach based on discrete-time models. Blend recipes and production targets were computed first using a nonlinear programming (NLP) model. Then a MILP model was employed to solve the short-term scheduling problem using such recipes and targets. The scheduling model was based on the resource-task-network (RTN) representation and did not consider multipurpose tanks or the delivery-scheduling problem.

Li et al. [13] formulated a continuous-time MILP model featuring a common time grid for all units (i.e., blenders and tanks). Li and Karimi [3] extended and improved this MILP model by using unit-specific time grids and including most of the operational constraints found in practice. Both models optimized blend recipes using blend indices. Based on these two previous works, Li et al. [4] presented a unit-specific continuous-time MINLP formulation where nonlinear terms arise from enforcing constant blending rates.

Castillo and Mahalec [14] developed a three-level decomposition algorithm through which recipe optimization can be done using linear and/or nonlinear blending rules. They considered the distribution problem, blend-size threshold constraints, parallel non-identical blenders, swing tanks, and product-dependent setup times. A discrete-time model was formulated for each level. The first level optimized the blend recipes, the second level approximated the production schedule, and the third level computed a detailed blend-and-delivery schedule. Due to the large size of the scheduling model at the third level for the entire horizon, it was solved in subintervals. Solutions computed by this approach were better and the execution times for large problems were two orders of magnitude shorter than those from previous methods [3,13]. In their subsequent work, Castillo and Mahalec [15] introduced a significantly modified version of the continuous-time scheduling model from Li and Karimi [3] (with a smaller number of binary variables [16]) for dealing with the third level. Case studies with nonlinear blend-scheduling problems were solved very close to global optimality with short execution times.

Lotero et al. [17] proposed another formulation of the multi-period pooling problem. They denominated this discrete-time MINLP formulation as a hybrid of a source-based model (similar to Castro's split-fraction model [18]) and a concentration-based model [19]. Redundant constraints were added to improve the linear relaxation, and the model was solved using a two-stage MILP-NLP approach. The MILP was a relaxation of the original MINLP model. The NLP model was obtained by fixing the integer variables of the original model to the values computed by the MILP. The algorithm adds integer, optimality, or feasibility cuts to the MILP model at each iteration, and stops when the difference between the MILP and NLP solutions is smaller than a pre-specified tolerance.

Cerdá et al. [20] presented a continuous-time MILP formulation that uses floating slots dynamically allocated to time periods while solving the problem. The model included most of the operational constraints found in practice. Cerdá et al. [21] then extended the model to handle nonlinear blending rules, thus formulating a continuous-time MINLP model. An approximate MILP formulation was derived by replacing the nonlinear blending rules with linear blend indices. The values of the binary variables computed by this MILP were fixed in the original MINLP, thus becoming an NLP that was solved to find a near-optimal solution of the original problem.

1.2. Literature review on global optimization

Global optimization of nonlinear non-convex problems has been a subject of extensive research over the last three decades. Even though powerful commercial solvers have been developed [22,23], there has been a continuous stream of advances in the field.

Global optimization algorithms have in common the generation of a convex relaxation of the problem, which provides a lower bound to the value of the objective function, and a way to generate feasible solutions (the upper bound). In addition, they have a method of bringing the lower and upper bounds together, so that the optimality gap can be reduced to ε -tolerance.

Computing a tight lower bound is absolutely critical. This may involve replacing the original formulation with an equivalent that preserves the feasible space but has a stronger relaxation (the different formulations for the pooling problem are a well-known example [24]). Another option is to reorganize the model constraints and add others that, although redundant in the original space, strengthen the relaxation. This procedure is known as the reformulation linearization technique [25]. The disadvantage is that there is no systematic way of knowing where to act in order to move toward a stronger relaxation.

A widely used method that guarantees convergence to the global optimal solution is known as spatial branch-and-bound [26,27]. It is an essential element of commercial solvers such as BARON [22], ANTIGONE [23], Couenne [28], and SCIP [29]. Spatial branch-and-bound works by iteratively reducing the domain of the variables, one by one, which in turn improves the quality of the convex relaxation. Note that if the initial relaxation is weak, due to the presence of many non-convex terms, convergence can be rather slow. It is thus important to have good branching strategies and bound-tightening techniques. Optimality-based bound tightening (OBBT) is an example of the latter. Although typically applied only at the root node or up to a limited depth [28], recent results have shown that applying OBBT in every node may lead to considerably lower optimality gaps [30]. OBBT involves solving one minimization and one maximization problem for each variable (appearing in non-convex terms) in order to generate tighter lower and upper bounds. It can be solved sequentially—which has the advantage of generating tighter variable bounds and the disadvantage of being computationally expensive when dealing with a large number of variables—or in parallel [31].

Bilinear terms are a common source of non-convexities in process

systems engineering. They can be relaxed using the McCormick envelopes, considering either the full domain of the variables [32] or a reduced domain following partitioning [33,34]. Simultaneous domain partitioning involves adding a new set of binary variables to the problem and guarantees global optimality in the limit of an infinite number of partitions. Spatial branch-and-bound can thus be avoided. One critical aspect concerns the scaling of problem size with the number of partitions. Earlier piecewise relaxation techniques [33] scale linearly, while recent formulations scale logarithmically. Examples of the latter are described next.

Misener et al. [35] developed a global optimization algorithm for the standard pooling problem and concluded that the logarithmic scheme is more advantageous for more than eight partitions. Kolodziej et al. [19] proposed a MINLP formulation for the multi-period pooling problem, in which nonlinearities arise from using the dynamic inventories of tanks as blend components. They employed a radix-based discretization technique that partitions one variable in every bilinear term to obtain a MILP relaxation. This discretization technique is known as multiparametric disaggregation [36]. Castro [37] developed the normalized multiparametric disaggregation technique (NMDT) [36], which works by discretizing the range between a variable's lower and upper bounds (belonging to $[0, 1]$). The advantage is that the number of partitions becomes the same for every discretized variable, even if their domain is different (when using a global discretization level parameter). NMDT has been successfully used to solve multi-period blending problems to global optimality, both as a stand-alone approach [18,38,39] or integrated in a spatial branch-and-bound algorithm [30].

Overall, contributions from cited works have enabled optimal solutions of gasoline blend-scheduling problems up to a certain model size. However, when dealing with large-scale problems, the computation and validation of global optimal solutions remain a difficult challenge.

1.3. Contributions of this work

This work presents a novel deterministic global optimization algorithm to solve non-convex MINLP or NLP models with nonlinearities that are strictly due to bilinear and/or quadratic terms. The main features of this algorithm are:

- The use of different linear and piecewise linear relaxation techniques to derive convex relaxations of the original non-convex model;
- The collection of different feasible solutions from the convex relaxation, which provide starting points for a local nonlinear solver to find feasible solutions of the original model;
- A dynamic increase in the number of partitions for piecewise linear relaxations;
- The reduction of the domain of the variables involved in nonlinear terms by means of an OBBT method; and
- The parallelization of the steps regarding computation of feasible solutions and the OBBT method.

The algorithm is tested on different gasoline blend-scheduling examples. For this class of problems, the results show that the proposed algorithm is on par with or better than two leading commercial global solvers.

The rest of this paper is organized as follows: Section 2 describes the problem statement and the assumptions made. Section 3 reviews the scheduling model employed in this work and presents the nonlinear equations used for octane blending. Section 4 briefly explains the piecewise linear relaxations employed to compute the estimates of the global solution. Section 5 describes the OBBT method to reduce the domain of the variables involved in nonlinear terms. Section 6 presents the steps of the global optimization algorithm. Section 7 contains the data describing the test examples. Section 8

shows the results obtained with the proposed algorithm and provides a comparison with other methods. The paper ends with conclusions in Section 9.

2. Problem statement

Given a blending system (i.e., storage tanks, blenders, and their interconnections; see Fig. 3), a scheduling horizon, a set of blend components and their corresponding supply and quality profiles along the horizon, a set of products and their minimum and maximum quality property specifications, a set of delivery orders for each product, and the initial inventory levels, it is required to determine the blend recipes, the production and delivery sequences, and the inventory profiles of all tanks, while minimizing the cost of the blended materials plus the switching costs (i.e., number of blend runs, number of tanks delivering the same order, and product transitions in the swing tanks) and the demurrage costs (i.e., late deliveries).

The following constraints are considered:

- (1) If a blender is to produce a product, it must blend at least a minimum amount.
- (2) A blender can produce at most one product at any time. Once it begins blending, it must operate for some minimum time before it can switch to another product.
- (3) A blender requires a minimum setup time during a product changeover.
- (4) A blender can feed at most one product tank at any time (industrial practice).
- (5) Product tanks can only store one product at any time.
- (6) Product tanks cannot receive and deliver material at the same time.

The assumptions made in this work are:

- (1) The flowrate profile of each component from the upstream process is piecewise constant.
- (2) The component quality profile is piecewise constant.
- (3) Perfect mixing occurs in each blender.
- (4) There is only one tank for a given blend component.
- (5) Only swing tanks can change their product service (i.e., change from storing one product to storing another).
- (6) Changeover times between products are negligible for swing tanks.
- (7) For each blender, changeover times between product blending are product dependent but sequence independent.
- (8) Each order involves only one product (any original order involving different products can be disaggregated into orders of a single product).
- (9) All orders are fulfilled during the scheduling horizon.

In summary, this problem considers the scheduling of blending and delivery operations, recipe determination, and product allocation of swing tanks along the scheduling horizon.

3. Gasoline blend-scheduling model

The scheduling model employed in this work is the one presented by Castillo and Mahalec [16]. It employs a continuous-time formulation, considers nonlinear blending equations, and does not allow simultaneous receipt and delivery by product tanks. This is a non-convex MINLP model, and it will be denoted as model **P** (or problem **P**). The main features of the scheduling model are highlighted in this section.

The scheduling model uses unit-specific time slots of varying length to determine when a specific task needs to be executed in each unit (blenders and tanks in this case). We assign a sufficiently high number of time slots, which will likely be higher than the number of slots required for blending each grade. This ensures that

there are sufficient degrees of freedom (enough available switches) to meet the varying product-delivery schedule.

The start time of a unit slot is equal to the end time of the previous one. The first unit slot starts at the beginning of the scheduling horizon, and the end of the last unit slot matches exactly the end of the horizon. Blending tasks begin at the start of a time slot, but can finish before its end. Delivery tasks from product tanks can start and finish within the corresponding slot. It is assumed that component tanks are continuously receiving material at some specified rate (i.e., the supply profile). Time periods are used to delineate the points where changes occur in the supply rates and/or quality of blend components. Time slots are assigned to these time periods. A time slot must end within its assigned period. However, for component tanks, the last time slot of a period must end exactly at that period's boundary (in order to properly respect the changes in supply rates and/or quality of blend components). Fig. 4 shows a graphical representation of these unit-specific time slots for a blending system with two blend component tanks (CT1, CT2), one blender (B1), and two product tanks (PT1, PT2). Unit slots 1 and 2 are pre-assigned to period 1, while slots 3 and 4 are pre-assigned to period 2. Note that the optimization has determined that slot 3 in the CT2 grid, and slot 4 in the PT1 grid, have zero length.

The objective of the scheduling model is to minimize the blend cost (i.e., materials cost), the switching cost associated with each blend run, product changeovers in the swing tanks, the number of “delivery runs” (i.e., the number of time slots used to deliver a specific order from a given tank), and the demurrage cost. Delivery runs are penalized in order to avoid computing delivery schedules that deliver the same order from several tanks at the same time, and to minimize intermittent deliveries of the same order from the same tank.

Binary variables are employed in the model to determine, at each time slot, the following discrete decisions:

- Which product tank each blender is feeding (one variable for each blender-tank connection);
- What gasoline grade is stored in each product tank (one variable for each grade-tank pair); and
- What demand order each product tank is partially or completely fulfilling (one variable for each tank-order connection).

With these binary variables, other discrete decisions can be modeled with 0–1 continuous variables, such as:

- What gasoline grade each blender is producing;
- The status of a blender (running or idle);
- The transition of a blender from running to being idle, or vice versa;
- When a new blend run starts;
- Product transitions in the blenders; and
- Product changeovers in the swing tanks.

The scheduling model also considers variable blending rates,

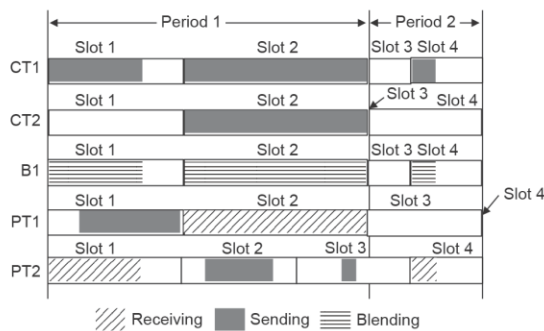


Fig. 4. Representation of unit-specific time slots employed in the scheduling model.

variable delivery rates, blender- and product-specific setup times for the blenders (i.e., idle times for, e.g., cleaning or sensor recalibration purposes), maximum delivery rates from blend component tanks to the blenders, minimum blend size, and minimum running times for each blender and product. Other constraints include the material balances, product composition specifications, product quality specifications, and linear and/or nonlinear blending equations.

The difficulty in solving this scheduling model to global optimality arises from the following factors:

- The significant number of discrete decisions that can be made, which are directly related to the number of time slots, gasoline grades, blenders, product tanks, and demand orders (the combinatorial nature of the problem);
- The inclusion of nonlinear blending equations (the non-convex nature of the problem); and
- All the considered operational constraints.

Castillo and Mahalec [16] found that introducing constraints regarding minimum blend cost and minimum switching cost can improve the quality of the solution and reduce the execution times for small- to medium-size problems. The minimum blend cost is computed using the approach delineated in Castillo et al. [40].

The nonlinear blending equations are presented next, since they were rewritten in such a way that nonlinear terms are only bilinear or quadratic.

Nonlinear blending equations

Eq. (1) to Eq. (19) are the proposed reformulation of the ethyl RT-70 model for octane blending [5,41]. Bilinear terms appear in Eqs. (1), (13), (14), and (18). Quadratic terms appear in Eqs. (15), (16), (17), and (19). Main sets, subscripts, variables, and parameters are described next. Set $\mathbf{I} = \{i\}$ consists of the blend components, $\mathbf{BL} = \{bl\}$ is constituted by the blenders, $\mathbf{N1} = \{n\}$ is the time slots, and set $\mathbf{QN} = \{(\theta, n)\}$ represents the time slots associated with each quality profile θ . Variable $V_{\text{comp}}(i, bl, n)$ indicates the volume of blend component i to blender bl during slot n . Variable $V_{\text{blend}}(bl, n)$ is the volume being processed by blender bl during slot n . The volume fraction of component i going into blender bl during slot n is denoted by variable $r(i, bl, n)$. Parameter $Q_{bc}(i, e, \theta)$ represents the value of quality property e for blend component i and quality profile θ . $sens(i, \theta)$ is a parameter known as the octane number sensitivity; it is the difference between the octane numbers, that is, RON – MON, of blend component i and quality profile θ . The values for the ethyl RT-70 model coefficients are taken from Singh et al. [5] and are as follows: $a_1 = 0.03224$, $a_2 = 0.00101$, $a_3 = 0$, $a_4 = 0.0445$, $a_5 = 0.00081$, and $a_6 = -0.0645 \times 10^{-4}$.

$$V_{\text{comp}}(i, bl, n) = r(i, bl, n)V_{\text{blend}}(bl, n) \quad \forall i, bl, n \in \mathbf{N1} \quad (1)$$

$$r_{\text{avg}}^{\text{RON}}(bl, n) = \sum_i r(i, bl, n)Q_{bc}(i, e, \theta) \quad (2)$$

$\forall e = \text{“RON”}, bl, n \in \mathbf{N1}, \theta: (\theta, n) \in \mathbf{QN}$

$$r_{\text{avg}}^{\text{MON}}(bl, n) = \sum_i r(i, bl, n)Q_{bc}(i, e, \theta) \quad (3)$$

$\forall e = \text{“MON”}, bl, n \in \mathbf{N1}, \theta: (\theta, n) \in \mathbf{QN}$

$$O_{\text{avg}}(bl, n) = \sum_i r(i, bl, n)Q_{bc}(i, e, \theta) \quad (4)$$

$\forall e = \text{“OI”}, bl, n \in \mathbf{N1}, \theta: (\theta, n) \in \mathbf{QN}$

$$A_{r_{\text{avg}}}(bl, n) = \sum_i r(i, bl, n)Q_{bc}(i, e, \theta) \quad (5)$$

$\forall e = \text{“Ar”}, bl, n \in \mathbf{N1}, \theta: (\theta, n) \in \mathbf{QN}$

$$O_{\text{avg}}^{\text{sq}}(bl, n) = \sum_i r(i, bl, n)[Q_{bc}(i, e, \theta)^2] \quad (6)$$

$\forall e = \text{“OI”}, bl, n \in \mathbf{N1}, \theta: (\theta, n) \in \mathbf{QN}$

$$A_{r_{\text{avg}}}^{\text{sq}}(bl, n) = \sum_i r(i, bl, n)[Q_{bc}(i, e, \theta)^2] \quad (7)$$

$\forall e = \text{“Ar”}, bl, n \in \mathbf{N1}, \theta: (\theta, n) \in \mathbf{QN}$

$$sens_{\omega_g}(bl, n) = \sum_i r(t, bl, n) sens(t, \theta) \quad \forall t, bl, n \in \mathbf{NI} \quad (8)$$

$$sens_{\omega_g}^{RON}(bl, n) = \sum_i r(t, bl, n) Q_{\omega_g}(t, e, \theta) sens(t, \theta) \quad (9)$$

$\forall e = \text{"RON"}, bl, n \in \mathbf{NI}, \theta : (\theta, n) \in \mathbf{QN}$

$$sens_{\omega_g}^{MON}(bl, n) = \sum_i r(t, bl, n) Q_{\omega_g}(t, e, \theta) sens(t, \theta) \quad (10)$$

$\forall e = \text{"MON"}, bl, n \in \mathbf{NI}, \theta : (\theta, n) \in \mathbf{QN}$

$$Q_{pr}(bl, e, n) = r_{\omega_g}^{RON}(bl, n) + a_1 [sens_{\omega_g}^{RON}(bl, n) - r_{\omega_g}^{RON}(bl, n)]$$

$$+ a_2 [O_{\omega_g}^{Rq}(bl, n) - O_{\omega_g}(bl, n)] \quad (11)$$

$$+ a_3 [Ar_{\omega_g}^{2q}(bl, n) - 2 \cdot Ar_{\omega_g}^3(bl, n) + Ar_{\omega_g}^4(bl, n)]$$

$\forall e = \text{"RON"}, bl, n \in \mathbf{NI}$

$$Q_{pr}(bl, e, n) = r_{\omega_g}^{MON}(bl, n) + a_4 [sens_{\omega_g}^{MON}(bl, n) - r_{\omega_g}^{MON}(bl, n)]$$

$$+ a_5 [O_{\omega_g}^{Rq}(bl, n) - O_{\omega_g}(bl, n)] \quad (12)$$

$$+ a_6 [Ar_{\omega_g}^{2q}(bl, n) - 2 \cdot Ar_{\omega_g}^3(bl, n) + Ar_{\omega_g}^4(bl, n)]$$

$\forall e = \text{"MON"}, bl, n \in \mathbf{NI}$

$$r_{\omega_g}^{RON}(bl, n) = r_{\omega_g}^{RON}(bl, n) sens_{\omega_g}(bl, n) \quad \forall bl, n \in \mathbf{NI} \quad (13)$$

$$r_{\omega_g}^{MON}(bl, n) = r_{\omega_g}^{MON}(bl, n) sens_{\omega_g}(bl, n) \quad \forall bl, n \in \mathbf{NI} \quad (14)$$

$$Ar_{\omega_g}^2(bl, n) = Ar_{\omega_g}^r(bl, n)^2 \quad \forall bl, n \in \mathbf{NI} \quad (15)$$

$$O_{\omega_g}(bl, n) = O_{\omega_g}^r(bl, n)^2 \quad \forall bl, n \in \mathbf{NI} \quad (16)$$

$$Ar_{\omega_g}^{2q}(bl, n) = Ar_{\omega_g}^{2q}(bl, n)^2 \quad \forall bl, n \in \mathbf{NI} \quad (17)$$

$$Ar_{\omega_g}^3(bl, n) = Ar_{\omega_g}^{2q}(bl, n) Ar_{\omega_g}^2(bl, n) \quad \forall bl, n \in \mathbf{NI} \quad (18)$$

$$Ar_{\omega_g}^4(bl, n) = Ar_{\omega_g}^2(bl, n)^2 \quad \forall bl, n \in \mathbf{NI} \quad (19)$$

4. Piecewise linear relaxations

As mentioned in Section 1, the use of piecewise linear relaxations is becoming more widespread due to the maturity of MILP solvers. Piecewise McCormick relaxation (PMCR) and the NMDT will be employed in this work. These techniques replace each bilinear term in model **P** with a single variable, thus linearizing the corresponding equations. This single variable is then subject to various linear constraints, which add extra continuous and binary variables to the model. If equal to 1, these extra binary variables activate a specific interval of the domain (i.e., partition) of one of the variables in the

bilinear term (denoted as the discretized variable). The number of partitions is denoted as NP , and it is assumed that all discretized variables have the same number of partitions. PMCR has a linear relation between NP and the number of extra binary variables required per discretized variable, while NMDT exhibits a logarithmic relation. For a more detailed explanation of these methods, the reader is encouraged to review Refs. [16,37].

The resulting MILP model is denoted as model **PR** and is a relaxation of problem **P**. This means that the optimal solution of model **PR** is a valid estimate of the global solution of **P** (in the minimization case, this will be a lower bound, LB). Moreover, an estimate of the best possible solution of model **PR** is a valid estimate of the global optimum of **P**. Therefore, even if model **PR** is not solved to optimality by a MILP solver within a given allocated time, a new estimate of the global solution can still be found. The larger the number of partitions, the closer model **PR** is to model **P**; see Fig. 5 for an illustration with an example involving a single discretized variable.

If the relaxation is tight, then its optimal solution will be very close to the original optimum. Hence, a strategy to find a feasible solution to the original problem **P** (in the minimization case, this will be an upper bound, UB) is to initialize **P** with the optimal solution of model **PR**. Since some MILP solvers, such as CPLEX, can store multiple feasible solutions to the MILP problem, potentially leading to different solutions of **P** due to the different starting points, we use a multi-start strategy in parallel fashion. Note that, for practical reasons related to the speed and robustness of commercial solvers, it is more convenient to solve NLP models instead of MINLPs. This is the reason why the values of the binary variables are fixed, converting problem **P** (MINLP) into **PF** (NLP). The compact notations of models **P**, **PR**, and **PF** are as follows:

Model **P**:

$$\min f_0(x, y)$$

$$\text{s.t. } f_m(x, y) \leq 0 \quad \forall m \in \mathbf{M} \setminus \{0\}$$

$$f_m(x, y) = \sum_{(x,y) \in \text{BLT}} a_{im} x_i x_j + B_m x + C_m y + d_m \quad \forall m \in \mathbf{M}$$

$$0 \leq x^L \leq x \leq x^U$$

$$x \in \mathbb{R}^N, y \in \{0, 1\}^D$$

Model **PR**:

$$\min f_0^R(x, y)$$

$$\text{s.t. } f_m^R(x, y) \leq 0 \quad \forall m \in \mathbf{M} \setminus \{0\}$$

$$f_m^R(x, y) = \sum_{(x,y) \in \text{BLT}} a_{im} W_{ij} + B_m x + C_m y + d_m \quad \forall m \in \mathbf{M}$$

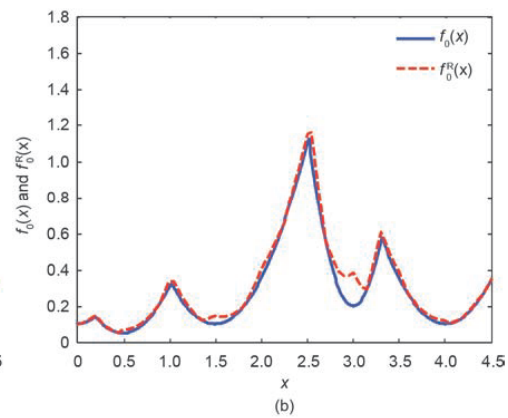
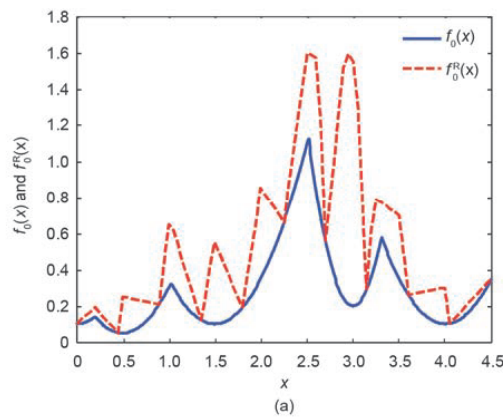


Fig. 5. Accuracy of the relaxation (f_0^R) with respect to exact representation (f_0) of the boundaries of a feasible region increases with the number of partitions (maximization problem). (a) 10 partitions; (b) 100 partitions.

$$\begin{aligned} g_n^R(x, w, v, z) &\leq 0 \quad \forall n \in N \\ g_n^R(x, w, v, z) &= H_n^T x + A_n^T w + B_n^T v + C_n^T z + d_n^T \quad \forall n \in N \\ 0 &\leq x^L \leq x \leq x^U \end{aligned}$$

Model PF:

$$\begin{aligned} \min f_0^R(x) \\ \text{s.t. } f_m^R(x) &\leq 0 \quad \forall m \in M \setminus \{0\} \\ f_m^R(x) &= \sum_{(i,j) \in \text{BLT}} a_{ijm} x_i x_j + B_m x + C_m y + d_m \quad \forall m \in M \\ 0 &\leq x^L \leq x \leq x^U \\ x &\in \mathbb{R}^{lx} \end{aligned}$$

Note that, in this section, set $M = \{m\}$ represents all the original constraints, set $N = \{n\}$ represents all the constraints required by the piecewise linear relaxation technique, and set $\text{BLT} = \{(i, j)\}$ represents all the bilinear terms. Variables x and y are the original continuous and binary variables, respectively, and v and z are the extra continuous and binary variables, respectively, required by the relaxation strategy. Variable w_j is the continuous variable that replaces the bilinear term $x_i x_j$. Scalars lx , ly , lv , lz represent the size of vectors x , y , v , and z , respectively. Parameters x^L and x^U are respectively the lower and upper bounds of the x variables. Note that quadratic terms can be treated as bilinear terms.

5. Tightening bounds on the variables

Model PR becomes tighter (i.e., closer to model P) as the number of partitions of the discretized variables is increased. However, an increase in the number of partitions produces an increment in the size of model PR and, after a certain number of partitions, model PR can become computationally intractable. Therefore, another technique is required in order to avoid the necessity of a large number of partitions. In this work, an OBBT method is employed [34,42]. The idea is to reduce the domain of the variables involved in nonlinear terms by computing new bounds of these variables by solving two optimization problems (a maximization problem and a minimization problem per variable). This is done after a new and better feasible solution to P is computed. After reducing the domain of the variables, model PR becomes closer to P without increasing the number of partitions, as shown in Fig. 6.

The mathematical model used in OBBT is denoted as model PRB, which is constructed as a relaxation of P, but with a different objective function and an extra constraint. To compute a lower bound of variable x_k , that is, x_k^L , the objective function is to minimize this variable. To compute an upper bound of variable x_k , that is, x_k^U , the

objective function is to maximize this variable. In order to compute new bounds, the extra constraint added imposes the condition that the value of the relaxed version of the original objective function, that is, $f_0^R(x, y)$, must be at least as good as the current best feasible solution.

Note that models PR and PRB can use different relaxations. In this work, model PRB employs standard McCormick envelopes [32] and integrality requirements on variables y are dropped, thus reducing PRB to linear programming (LP). The lower and upper bounds of variable x_k are updated with the optimal solutions of the corresponding LP model. Compact notation of model PRB is shown below for a minimization problem.

Model PRB:

$$\begin{aligned} x_k^L &= \min x_k \quad (x_k^U = \max x_k) \\ \text{s.t. } f_0^R(x, y) &\leq UB \\ f_m^R(x, y) &\leq 0 \quad \forall m \in M \setminus \{0\} \\ f_m^R(x, y) &= \sum_{(i,j) \in \text{BLT}} a_{ijm} w_{ij} + B_m x + C_m y + d_m \quad \forall m \in M \\ g_k^{\text{RB}}(x, w) &\leq 0 \quad \forall k \in K \\ g_k^{\text{RB}}(x, w) &= H_k^T x + A_k^T w + d_k^T \quad \forall k \in K \\ 0 &\leq x^L \leq x \leq x^U \\ x &\in \mathbb{R}^{lx}, y \in [0, 1]^{ly}, w \in \mathbb{R}^{lw} \end{aligned}$$

OBBT consists of solving these LP models for all the variables involved in nonlinear terms, in a parallel framework, to reduce execution times. Thus, the bounds will generally be weaker than when solving the problems sequentially. Since the number of instances to solve can be very large, instances are solved in different blocks. These blocks are defined by a maximum number of problems to be solved in parallel. After one block is solved, the corresponding bounds are updated and then the next block is solved. Fig. 7 shows the flowchart of the OBBT method. Note that OBBT is applied only once per variable.

6. Global optimization algorithm

The steps of the proposed global optimization algorithm are presented next for a minimization problem. Fig. 8 shows the corresponding flowchart. Note that the algorithm can be applied to any MINLP problem with nonlinearities exclusive to bilinear or quadratic terms.

- (1) Initialize algorithm parameters. Define the number of partitions to be used $\{NP_1, NP_2, \dots, NP_{\text{last}}\}$ and set $NP = NP_1$. Set the

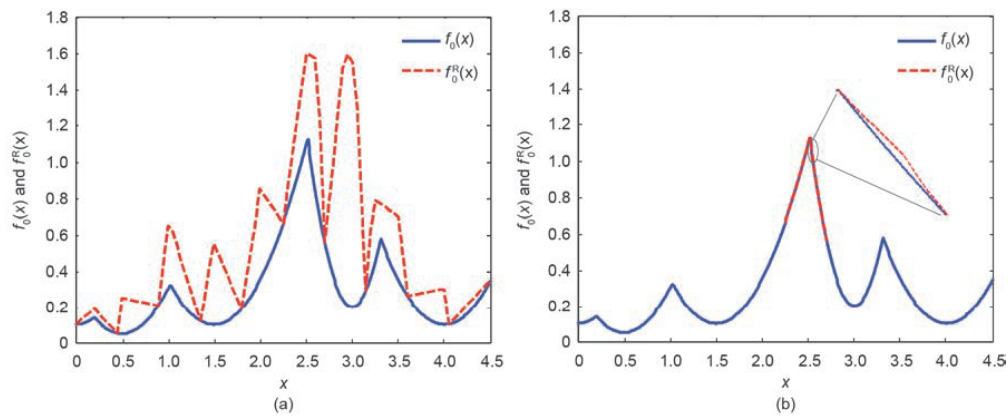


Fig. 6. Accuracy of the relaxation increases when the domain of the variables involved in nonlinear terms is reduced. (a) 10 partitions with $x \in [0, 4.5]$; (b) 10 partitions with $x \in [2.25, 2.7]$.

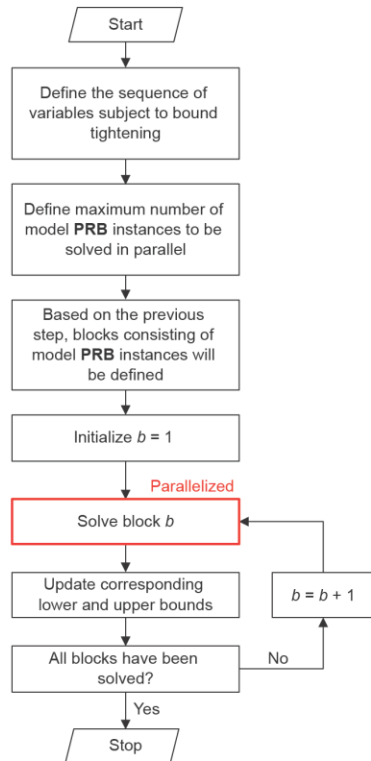


Fig. 7. Flowchart of the OBBT method.

lower bound $LB = -\infty$, upper bound $UB = +\infty$, total number of iterations counter $IT_{total} = 1$, iterations with the same number of partitions $IT_{sameNP} = 1$, maximum number of total iterations IT_{total}^{max} , maximum number of iterations with the same number of partitions IT_{sameNP}^{max} , maximum total time $TIME_{total}^{max}$, and minimum relative tolerance ϵ .

- (2) Lower bound computation. Solve MILP model **PR** using the CPLEX solver with parallel and solution pool options active. Update LB with the best possible solution from CPLEX, if this value is greater than the previous LB .
- (3) Upper bound computation. Use the solutions stored in the CPLEX solution pool as starting points for NLP model **PF**. Solve NLP model **PF** instances in parallel using a local nonlinear solver. Update UB if any of the computed solutions is feasible and has a smaller objective function value than the previous UB .
- (4) Update optimality gap. The following formula is used in this step: $OptGap = (UB - LB)/LB \times 100$.
- (5) Check termination criteria. Stop if $OptGap \leq \epsilon$, if $IT_{total} = IT_{total}^{max}$, if the total execution time is equal to or greater than $TIME_{total}^{max}$, or if the number of partitions has already reached NP_{last} . Otherwise, continue to Step 6.
- (6) If the upper bound UB did not improve in Step 3, or if $IT_{sameNP} = IT_{sameNP}^{max}$, continue to Step 7. Otherwise, reduce the domain of the variables involved in nonlinear terms using the OBBT method described in Section 5; set $IT_{total} = IT_{total} + 1$ and $IT_{sameNP} = IT_{sameNP} + 1$, and then go back to Step 2.
- (7) Increase the number of partitions to the next specified value. Set $IT_{total} = IT_{total} + 1$ and go back to Step 2.

Although the main elements of the algorithm have already been proposed (e.g., PMCR, NMDT, OBBT), the novelty is related to the way in which they are implemented. More specifically: ① the CPLEX solution

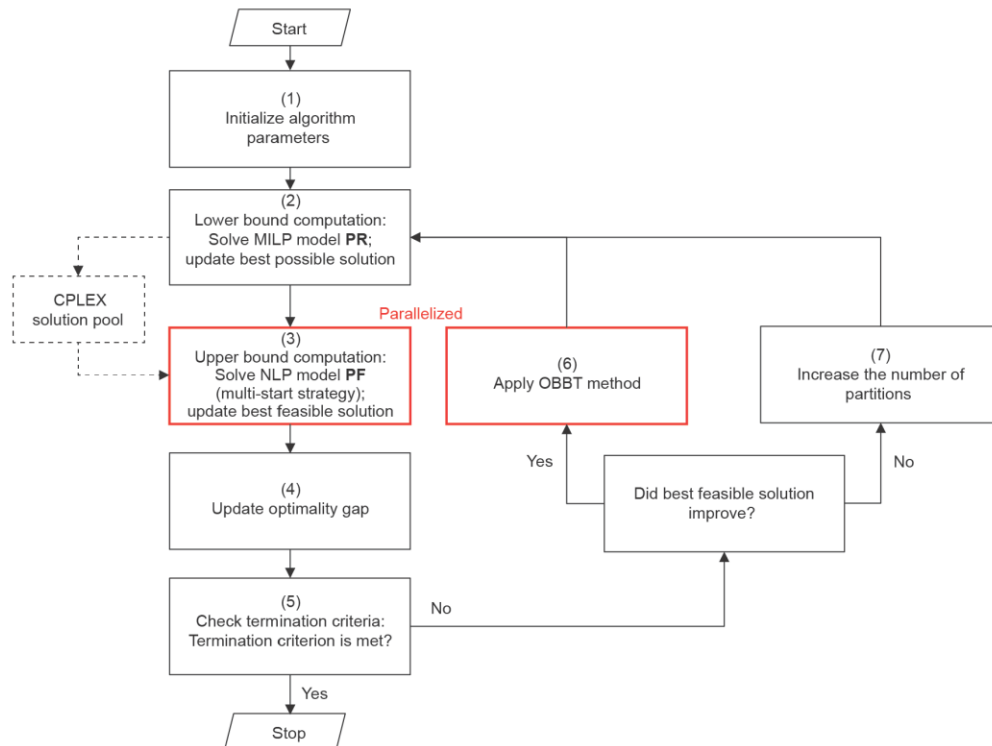


Fig. 8. Flowchart of the global optimization algorithm.

pool is used to store starting points for model PF, ② instances of model PF are solved in parallel, ③ OBBT is applied to blocks of variables and in a parallel framework, and ④ no branching strategy is employed.

7. Case studies

The tests in this paper consist of Examples 4, 8, 12, and 14 from Li and Karimi [3]. The difference in this work is that the ethyl RT-70 models are considered for blending RON and MON properties (as described in Section 3.1) instead of blend indices. RON index correlations from Li et al. [13], shown in Eq. (20) and Eq. (21), were used to compute the actual RON values from the blend indices given by Li and Karimi [3]. RBN denotes the research octane number blend index. MON values were assumed in this work and the corresponding minimum product specifications were set equal to zero; therefore, MON constraints will not be active at the optimal solution. Quality properties of blend components are assumed to be known (recall Section 2); therefore, Eq. (20) and Eq. (21) are only used to

convert the blend components' RBN values to RON values before the optimization runs (i.e., they do not appear in the optimization problems).

$$RBN = RON + 11.5 \quad 0 \leq RON \leq 85 \quad (20)$$

$$RBN = \exp(0.0135RON + 3.422042) \quad RON > 85 \quad (21)$$

Table 1 describes the size of the blending system examples. Information about the periods, their duration, their corresponding time slots, and the orders that can be delivered within such periods is presented in Table 2. RON and MON values and their respective specifications are shown in Table 3. Table 4 presents the statistics regarding the size of model P when not using the constraints on the minimum blend and switching costs. When using such constraints, four equations are added to the model (minimum blend cost, minimum number of delivery runs, minimum number of blend runs, and minimum number of product changeovers in the swing tanks). Note that the size of the blending system and its corresponding scheduling model increase from Example 4 to Example 14.

Table 1
Summary of the blending system examples.

Example ID	Number of blenders	Number of orders	Number of products	Number of product tanks	Number of quality properties
4	1	15	4	11	9
8	2	20	4	11	9
12	2	35	5	11	9
14	3	45	5	11	9

Table 2
Periods, duration, time slots, and orders that can be delivered in each period.

Example ID	Period	Duration (h)	Slots	Orders that can be delivered
4	1	100	1, 2	O1–O7, O12–O15
	2	92	3, 4	O8–O11
8	1	80	1, 2	O1–O7, O12–O19
	2	70	3, 4	O8
	3	42	5, 6	O8–O11, O20
12	1	50	1–3	O1–O7, O12, O13, O15, O19, O33
	2	50	4–6	O14–O18, O27, O28, O33
	3	50	7–9	O8, O21, O24, O29–O32, O34, O35
	4	42	10–12	O8–O11, O20, O22, O23, O25, O26
14	1	50	1–3	O1–O7, O12, O13, O15, O19, O26
	2	50	4–7	O14–O18, O26
	3	50	8–10	O8, O21, O24, O27–O31, O45
	4	42	11–13	O8–O11, O20, O22, O23, O25, O32–O44

Table 3
RON and MON values and specifications.

Property	Blend components									Product specifications [min, max]				
	C1	C2	C3	C4	C5	C6	C7	C8	C9	P1	P2	P3	P4	P5
RON	75	90.3	95.6	97.3	83	100	115	118	81	[95, 200]	[96, 200]	[94, 200]	[90, 200]	[98, 200]
MON	66	80.8	80.5	91.7	74	100	109	100	72	[0, 200]	[0, 200]	[0, 200]	[0, 200]	[0, 200]

Table 4
Statistics of model P.

Example ID	Number of equations	Number of variables	Number of binary variables	Number of bilinear terms
4	6 207	2 503	433	168
8	9 297	3 323	553	336
12	23 087	8 170	1 317	672
14	32 574	10 693	1 628	1 092

8. Results

All the examples were solved on a computing machine Intel® Core™ i7-4710HQ central processing unit (CPU), 2.50 GHz, 12 GB random-access memory (RAM), Windows 10 (8-core). The global optimization algorithm was implemented in Python 2.7. The Python script generates general algebraic modeling system (GAMS) files with the corresponding mathematical models, which are then solved by employing the GAMS-Python application program interface (API). The selected solvers were CPLEX 12.6 for model **PR** and model **PRB**, and CONOPT 3 for model **PF**.

The global optimization algorithm termination criteria were as follows: 0.01% optimality gap or 3600 s (1 h). There was no limit on the total number of iterations, nor on the iterations with the same number of partitions. The numbers of partitions in model **PR** when using PMCR were {2, 4, 8, 16, 32}, and when using NMDT were {10, 100, 1000}.

The termination criteria for the MILP problems (instances of model **PR**) were: an optimality gap of 0.01% or 600 s. The CPLEX parallel option was active (in deterministic mode) with a maximum number of threads equal to 8. In addition, the CPLEX solution pool option was active with a maximum pool capacity of 30 and the replacement option that generates diverse solutions. Thus, a maximum of 30 instances of model **PF** was solved per iteration using the GAMS parallel computing grid facility. CONOPT 3 default termination criteria were used.

The termination criteria for the LP problems (instances of model **PRB**) were: optimality or 60 s. A maximum number of 100 LP problems were solved in parallel using the GAMS parallel computing grid facility.

For comparison purposes, the global commercial solvers BARON 15.9 [33] and ANTIGONE 1.1 [34] were employed to solve the original model **P** using the same termination criteria as the proposed global optimization algorithm.

Section 8.1 presents the results obtained when not including the constraints on the minimum blend cost and minimum switching cost, while Section 8.2 shows the results when such constraints are added to the model. A comparison with previously published heuristic methods is included in Section 8.3.

8.1. Not using constraints on the minimum blend and switching costs

A comparison of the results obtained by the proposed algorithm with those obtained by commercial solvers is presented in Table 5. For simplicity, we refer to our proposed algorithm as GO-PMCR when it uses piecewise McCormick relaxation to construct model **PR** and as GO-NMDT when it employs the NMDT.

ANTIGONE, BARON, and GO-PMCR compute the same solution for Examples 4 and 8. GO-NMDT computes the same solution for Example 4, but the final solution for Example 8 is slightly higher. GO-PMCR computes better solutions than GO-NMDT and ANTIGONE in all examples. In this work, this is because GO-PMCR can use more distinct numbers of partitions (2, 4, 8, 16, 32) than GO-NMDT (10, 100, 1000); thus, it generates more feasible solutions from the MILP relaxation.

BARON does not find a feasible solution for Examples 12 and 14 within 1 h. Based on the log files generated by commercial solvers, it seems that BARON relies more on the branching procedure, while ANTIGONE focuses more on the MILP relaxation and bound-tightening steps (as our proposed algorithm does). Feasible integer solutions for scheduling problems may be found only at deep nodes in the branch-and-bound tree [43], which can be of significant size when the number of binary variables is large.

In Examples 4 and 8, the algorithm and the commercial solvers compute similar optimality gaps. For Examples 12 and 14, BARON cannot compute an optimality gap (no feasible solution was found), and GO-PMCR obtains a lower optimality gap than GO-NMDT and ANTIGONE.

Both commercial solvers and the proposed algorithm did not solve all four examples to the desired tolerance within 1 h. The times reported in Table 5 are the times in which the best solution was found. GO-PMCR and GO-NMDT require shorter times than both commercial solvers. GO-NMDT is significantly faster than GO-PMCR only in the small-sized Example 4. Note that, for the number of partitions selected, the size of the MILP relaxation grows faster with GO-NMDT than with GO-PMCR. Therefore, MILP relaxations are often faster to solve to optimality with GO-PMCR. However, GO-PMCR will require more iterations. Based on the three factors considered (i.e., best solution found, optimality gap, and time to best solution), GO-PMCR shows the best performance. Fig. 9 shows the total number of binary variables in the relaxation of the scheduling model (i.e., model **PR**) when using PMCR and NMDT, at each iteration of the algorithm and for each example. It shows that, for the selected partition values, PMCR requires fewer binary variables in the first 4–5 iterations than NMDT at any iteration. This is expected since the partitions when using PMCR are fewer than 8 in those iterations, and NMDT starts with 10. Note that flat sections in the curves indicate that the OBBT method was applied instead of increasing the number of partitions.

The major differences between the proposed algorithm and BARON are:

- The use of piecewise relaxation methods for bilinear terms instead of standard McCormick envelopes; and
- Dynamically increasing the number of partitions in order to tighten the MILP relaxation, instead of implementing a branching strategy.

These features seem to be adequate for the scheduling problems presented here. We do not claim that our proposed algorithm will always be better than BARON when solving a different type of problem.

Our proposed algorithm and ANTIGONE perform similarly, but differ mainly in the following areas:

- The use of NMDT as a piecewise relaxation technique;
- How the number of partitions is increased in each iteration;
- When and how to apply OBBT; and
- The use of the CPLEX solution pool to store feasible solutions from the MILP relaxations and use them as starting points to solve the NLP problem.

Finally, ANTIGONE and BARON can handle more than just bilinear and quadratic terms; in addition, they apply other mathematical

Table 5
Summary of results (not using constraints on the minimum blend and switching costs).

Example ID	Best solution found (1000 USD)				Optimality gap (%)				Time to best solution (s)			
	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT
4	4 633	4 633	4 633	4 633	6.90	6.37	6.31	6.37	2 350	930	616	120
8	8 203	8 203	8 203	8 223	9.53	9.53	9.19	9.49	1 708	3 273	714	1 391
12	16 650	NF	15 408	15 440	20.51	NA	14.00	14.22	1 631	3 600	1 411	1 438
14	21 360	NF	21 316	31 639	12.50	NA	12.31	40.92	3 600	3 600	2 911	2 904

NF = not found; NA = not available.

techniques capable of improving performance (e.g., reformulation-linearization technique, cutting planes, feasibility-based bound tightening, different branching strategies, etc.).

8.2. Using constraints on the minimum blend and switching costs

For this case, the results computed by the algorithm and commercial solvers are presented in Table 6. The most notable differences with respect to Table 5 are the smaller optimality gaps and the shorter times for Examples 4 and 8.

Both commercial solvers and the algorithm find similar solutions for Examples 4 and 8. BARON still does not find a feasible solution for Examples 12 and 14 within the allocated time. GO-PMCR computes better solutions than GO-NMDT for Examples 12 and 14, which in turn has better solutions than ANTIGONE. Note that the addition of bounds caused an increment in the number of solutions for Example 8. Since the solutions from Section 8.1 are still feasible even with the inclusion of the constraints regarding minimum blend and switching costs, this suggests that such constraints are affecting the solvers. This effect is also observed in ANTIGONE in Examples 12 and 14.

Regarding optimality gaps, most of the same observations as in Section 8.1 can be made. Similar optimality gaps are computed by all methods for Examples 4 and 8. For Examples 12 and 14, GO-PMCR calculates lower optimality gaps than GO-NMDT and ANTIGONE.

Both commercial solvers and the proposed algorithm solve Example 4 to the desired tolerance; BARON is the slowest. For Example 12, the time to the best solution required by GO-PMCR is larger than that required by GO-NMDT; however, it must be considered that GO-PMCR computes a better solution.

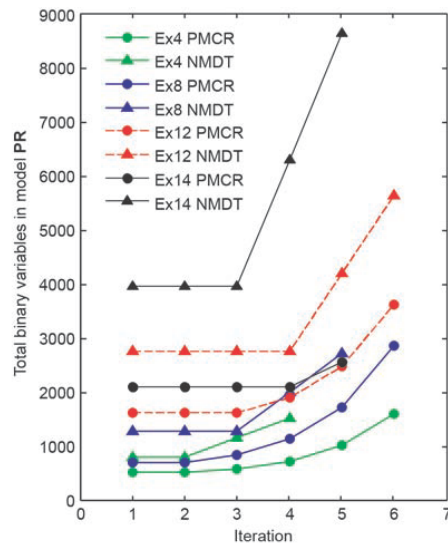


Fig. 9. Number of binary variables in model PR at each iteration of the algorithm.

Table 6
Summary of results (using constraints on the minimum blend and switching costs).

Example ID	Best solution found (1000 USD)				Optimality gap (%)				Time to best solution (s)			
	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT	ANTIGONE	BARON	GO-PMCR	GO-NMDT
4	4 633	4 633	4 633	4 633	0.01	0.01	0.01	0.01	26	296	30	14
8	8 207	8 204	8 206	8 204	0.05	0.02	0.04	0.02	557	1 218	103	140
12	23 590	NF	15 384	15 403	34.80	NA	0.01	0.13	3 333	3 600	2 674	742
14	23 520	NF	21 270	21 360	9.68	NA	0.13	0.55	1 636	3 600	2 574	2 845

NF = not found; NA = not available.

Overall, GO-PMCR shows the best performance once again. For illustration purposes, the blend and delivery schedules computed for Example 14 by the algorithm using PMCR are shown in Fig. 10 and Fig. 11, respectively.

8.3. Comparison with heuristic methods

In this section, the proposed algorithm is compared with previously published heuristic methods [15,21]. Table 7 [15,21] contains the best solution found by those methods and the time required to compute such solutions. Note that heuristic methods do not compute an optimality gap since they aim to find close-to-optimal solutions very rapidly, and they do not spend time estimating and refining the value of the global optimal solution. These heuristic methods are tailored to the examples used in this work. These methods construct the final solution by decomposing the original problem into different levels, each one with different accuracy and complexity. Short execution times are achieved by solving the least complex level first and then, in each subsequent level, fixing the values of the most important variables to those from the previous level's solution.

The objective function of the scheduling model employed in this work is the same as the one used by Castillo and Mahalec [15]. This objective function penalizes each individual blend run, even when the same product is being blended in contiguous blend runs. On the other hand, Cerdá et al. [21] did not penalize the number of individual blend runs, but only penalized the product transitions in the blenders. We show the adjusted values of the solutions reported by Cerdá et al. [21]; that is, individual blend runs are penalized.

All methods find the same solution for Example 4. In general, all the methods compute very similar solutions for the remaining examples. Solutions from Cerdá et al. [21] have higher costs for Examples 8, 12, and 14 because they did not originally penalize individual blend runs. The method from Cerdá et al. [21] might compute similar solutions to those from the other methods if it used the same objective function.

Heuristic methods still require smaller execution times than the proposed global optimization algorithm. This is expected, because those methods do not involve as many steps as global optimization techniques. The proposed global optimization algorithm does not find solutions of the same quality as quickly as the two selected heuristic methods. To compute feasible solutions in each iteration, our proposed algorithm needs to first solve a MILP (i.e., model PR). The solution of the MILP is the most time-consuming step, thus reducing the speed required to compute new feasible solutions. Moreover, the small number of partitions at the beginning of the algorithm may result in weak MILP relaxations, which generate starting points for the NLP models that are far from the global optimum.

These results indicate the need to improve the corresponding step to compute feasible solutions, or to simply integrate heuristic methods into the proposed algorithm.

9. Conclusions

In this work, we presented a global optimization algorithm that

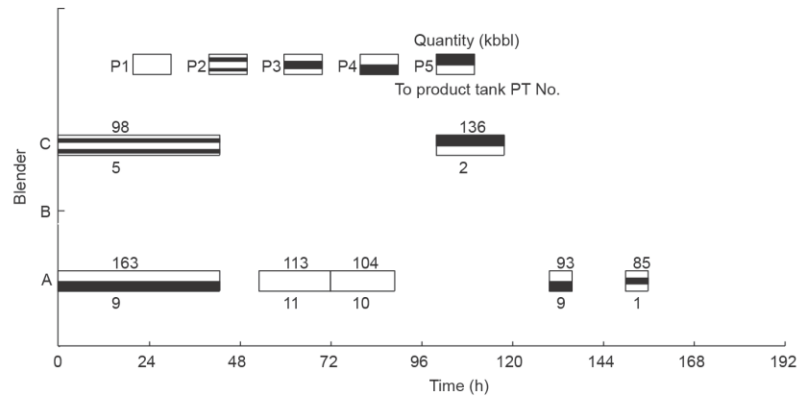


Fig. 10. Blend schedule computed for Example 14 by the proposed algorithm using PMCR. Kbb is short for kilobarrel, 1 kbb = 158.9873 m³.

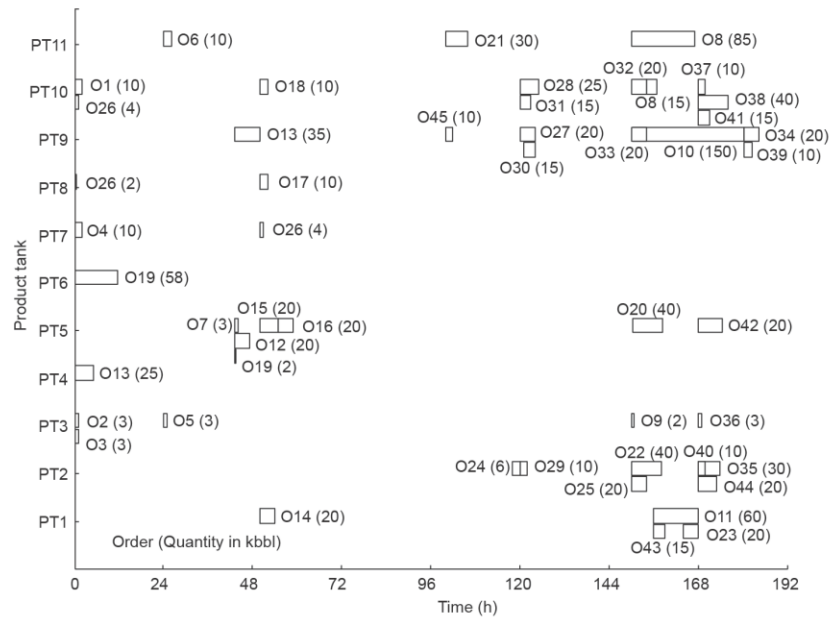


Fig. 11. Delivery schedule computed for Example 14 by the proposed algorithm using PMCR.

Table 7
Comparison with heuristic methods.

Example ID	Best solution found (1000 USD)					Time to best solution (s)			
	Castillo and Mahalec [15]	Cerdá et al. [21]	Cerdá et al. [21] adjusted values	GO-PMCR	GO-NMDT	Castillo and Mahalec [15]	Cerdá et al. [21]	GO-PMCR	GO-NMDT
4	4 633	4 613	4 633	4 633	4 633	3	0.4	30	14
8	8 203	8 163	8 223	8 206	8 204	6	7.5	103	140
12	15 403	15 342	15 442	15 384	15 403	17	31.0	2 674	742
14	21 263	21 181	21 301	21 270	21 360	24	21.0	2 574	2 845

can solve MINLP problems with bilinear and quadratic terms. The algorithm computes estimates of the global solution by constructing and solving MILP problems that are relaxations of the original problem obtained by using either PMCR or NMDT. These methods discretize the domain of one of the variables of a bilinear term into several partitions, and introduce extra binary and continuous variables into the model. To improve the estimates of the global optimum, the

number of partitions is increased during the algorithm.

To avoid a rapid increase in the model size due to a large number of partitions, an OBBT method is used. The MILP relaxation will be closer to the original problem if the number of partitions stays the same but the domain of the variables is reduced. The OBBT method solves several LPs in a parallel setting.

The CPLEX solution pool is active and stores different feasible

solutions found during the branch-and-bound procedure to solve the MILP relaxation. These solutions are then used as starting points for a nonlinear solver to find feasible solutions to the original problem. This step is also parallelized.

We showed that the proposed algorithm can be used to schedule gasoline-blending operations, taking into consideration the distribution problem and the most important operational constraints. We employ a continuous-time MINLP scheduling model [16] where the ethyl RT-70 models are used for octane blending.

The proposed algorithm was compared with two commercial solvers and two heuristic methods. The elements under evaluation were: the best solution found, the corresponding optimality gap, and the time to best solution. The proposed algorithm with PMCR showed a better performance than with NMDT. In our large-sized examples, the proposed algorithm with either PMCR or NMDT performed better than the commercial global solvers. This result shows that further research on this algorithm may be very promising. Both selected heuristic methods provided good solutions in shorter execution times than the global algorithms. This result indicates that the step to compute feasible solutions can still be improved.

We tested the performance of the algorithm by solving the scheduling model for two scenarios: ① not including lower bounds on the blend cost and switching costs, and ② including such bounds. The first problem is harder to solve and is representative of a kind of model one may write without diligently trying to reduce the search space as much as possible. Adding a tight lower bound to the blending cost as a constraint, as well as adding the lower bound to the switching costs, enables algorithms to search smaller spaces and improves their performance. This result also indicates that the relaxations are still not tight enough. Future work will include the derivation and addition of redundant and symmetry-breaking constraints; the testing of a different relaxation scheme for quadratic, cubic, and higher order terms (e.g., outer approximation); and the modification of the bound-tightening method in order to speed up the algorithm.

Acknowledgements

Support by Ontario Research Foundation, McMaster Advanced Control Consortium, and Fundação para a Ciência e Tecnologia (Investigador FCT 2013 program and project UID/MAT/04561/2013) is gratefully acknowledged.

Compliance with ethics guidelines

Pedro A. Castillo Castillo, Pedro M. Castro, and Vladimir Mahalec declare that they have no conflict of interest or financial conflicts to disclose.

Nomenclature

Sets and subscripts

BL = { <i>bl</i> }	Blenders
E = { <i>e</i> }	Quality properties
I = { <i>i</i> }	Blend components and corresponding storage tanks
NI = { <i>n</i> }	Time slots
QN = {{ <i>θ</i> , <i>n</i> }}	Time slot <i>n</i> is associated with the period with quality profile <i>θ</i>

Parameters

a_1, a_2, \dots, a_6	Coefficients for the ethyl RT-70 model
$Q_{bc}(i, e, \theta)$	Value of quality property <i>e</i> of blend component <i>i</i> during quality profile <i>θ</i>
$sens(i, \theta)$	Octane number sensitivity, i.e., octane difference RON – MON for blend component <i>i</i> during quality profile <i>θ</i>

Continuous variables

$Ar_{avg}(bl, n)$	Volumetric average of the aromatics content of the processed material by blender <i>bl</i> during slot <i>n</i>
$Ar_{avg}^2(bl, n)$	Volumetric average of the squared value of the aromatics content of the processed material by blender <i>bl</i> during slot <i>n</i>
$Ar2_{avg}(bl, n)$	Squared value of $Ar_{avg}(bl, n)$
$Ar2_{avg}^2(bl, n)$	Squared value of $Ar_{avg}^2(bl, n)$
$Ar3_{avg}(bl, n)$	Product of $Ar_{avg}^2(bl, n)$ and $Ar2_{avg}(bl, n)$
$Ar4_{avg}(bl, n)$	Squared value of $Ar2_{avg}(bl, n)$
$Ol_{avg}(bl, n)$	Volumetric average of the olefins content of the processed material by blender <i>bl</i> during slot <i>n</i>
$Ol_{avg}^2(bl, n)$	Volumetric average of the squared value of the olefins content of the processed material by blender <i>bl</i> during slot <i>n</i>
$Ol2_{avg}(bl, n)$	Squared value of $Ol_{avg}^2(bl, n)$
$Q_{pr}(bl, e, n)$	Value of quality property <i>e</i> of the processed material by blender <i>bl</i> during slot <i>n</i>
$r(i, bl, n)$	Volume fraction of blend component <i>i</i> going into blender <i>bl</i> during slot <i>n</i>
$r_{avg}^{MON}(bl, n)$	Volumetric average of the motor octane number of the processed material by blender <i>bl</i> during slot <i>n</i>
$r_{avg}^{RON}(bl, n)$	Volumetric average of the research octane number of the processed material by blender <i>bl</i> during slot <i>n</i>
$r_{avg}^{MON}(bl, n)$	Product of $r_{avg}^{MON}(bl, n)$ and $sens_{avg}(bl, n)$
$r_{avg}^{RON}(bl, n)$	Product of $r_{avg}^{RON}(bl, n)$ and $sens_{avg}(bl, n)$
$sens_{avg}(bl, n)$	Volumetric average of the octane number sensitivity of the processed material by blender <i>bl</i> during slot <i>n</i>
$sens_{avg}^{MON}(bl, n)$	Volumetric average of the octane number sensitivity times the motor octane number
$sens_{avg}^{RON}(bl, n)$	Volumetric average of the octane number sensitivity times the research octane number
$V_{blend}(bl, n)$	Volume being processed by blender <i>bl</i> during slot <i>n</i>
$V_{comp}(i, bl, n)$	Volume of blend component <i>i</i> transferred to blender <i>bl</i> during slot <i>n</i>

References

- Harjunkoski I, Maravelias CT, Bongers P, Castro PM, Engell S, Grossmann IE, et al. Scope for industrial applications of production scheduling models and solution methods. *Comput Chem Eng* 2014;62:161–93.
- Méndez CA, Grossmann IE, Harjunkoski I, Kaboré P. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. *Comput Chem Eng* 2006;30(4):614–34.
- Li J, Karimi I. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res* 2011;50(15):9156–74.
- Li J, Xiao X, Floudas CA. Integrated gasoline blending and order delivery operations: Part I. Short-term scheduling and global optimization for single and multi-period operations. *AIChE J* 2016;62(6):2043–70.
- Singh A, Forbes JF, Vermeer PJ, Woo SS. Model-based real-time optimization of automotive gasoline blending operations. *J Process Contr* 2000;10(1):43–58.
- Joly M, Pinto JM. Mixed-integer programming techniques for the scheduling of fuel oil and asphalt production. *Chem Eng Res Des* 2003;81(4):427–47.
- Floudas CA, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: A review. *Comput Chem Eng* 2004;28(11):2109–29.
- Sundaramoorthy A, Maravelias CT. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Ind Eng Chem Res* 2011;50(9):5023–40.
- Maravelias CT. General framework and modeling approach classification for chemical production scheduling. *AIChE J* 2012;58(6):1812–28.
- Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Ind Eng Chem Res* 2003;42(4):825–35.
- Jia Z, Ierapetritou M. Efficient short-term scheduling of refinery operations based on a continuous time formulation. *Comput Chem Eng* 2004;28(6–7):1001–19.
- Glismann K, Gruhn G. Short-term scheduling and recipe optimization of blending processes. *Comput Chem Eng* 2001;25(4–6):627–34.
- Li J, Karimi I, Srinivasan R. Recipe determination and scheduling of gasoline blending operations. *AIChE J* 2010;56(2):441–65.
- Castillo PAC, Mahalec V. Inventory pinch based, multiscale models for integrated planning and scheduling—Part II: Gasoline blend scheduling. *AIChE J* 2014;60(7):2475–97.
- Castillo PAC, Mahalec V. Inventory pinch gasoline blend scheduling algo-

- rithm combining discrete- and continuous-time models. *Comput Chem Eng* 2016;84:611–26.
- [16] Castillo PAC, Mahalec V. Improved continuous-time model for gasoline blend scheduling. *Comput Chem Eng* 2016;84:627–46.
- [17] Lotero I, Trespalacios F, Grossmann IE, Papageorgiou DJ, Cheon MS. An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem. *Comput Chem Eng* 2016;87:13–35.
- [18] Castro PM. New MINLP formulation for the multiperiod pooling problem. *AIChE J* 2015;61(11):3728–38.
- [19] Kolodziej SP, Grossmann IE, Furman KC, Sawaya NW. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. *Comput Chem Eng* 2013;53:122–42.
- [20] Cerdá J, Pautasso PC, Cafaro DC. A cost-effective model for the gasoline blend optimization problem. *AIChE J* 2016;62(9):3002–19.
- [21] Cerdá J, Pautasso PC, Cafaro DC. Optimizing gasoline recipes and blending operations using nonlinear blend models. *Ind Eng Chem Res* 2016;55(28):7782–800.
- [22] Tawarmalani M, Sahinidis NV. A polyhedral branch-and-cut approach to global optimization. *Math Program* 2005;103(2):225–49.
- [23] Misener R, Floudas CA. ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. *J Glob Optim* 2014;59(2):503–26.
- [24] Boland N, Kalinowski T, Rigtering F. New multi-commodity flow formulations for the pooling problem. *J Glob Optim* 2016;66(4):669–710.
- [25] Sherali HD, Alameddine A. A new reformulation-linearization technique for bilinear programming problems. *J Glob Optim* 1992;2(4):379–410.
- [26] Ryou HS, Sahinidis NV. A branch-and-reduce approach for global optimization. *J Glob Optim* 1996;8(2):107–38.
- [27] Smith EMB, Pantelides CC. Global optimization of nonconvex MINLPs. *Comput Chem Eng* 1997;21(Suppl):S791–6.
- [28] Belotti P, Lee J, Liberti L, Margot F, Wächter A. Branching and bounds tightening techniques for non-convex MINLP. *Optim Methods Softw* 2009;24(4–5):597–634.
- [29] Achterberg T. SCIP: Solving constraint integer programs. *Math Program Comput* 2009;1(1):1–41.
- [30] Castro PM. Spatial branch-and-bound algorithm for MIQCPs featuring multiparametric disaggregation. *Optim Methods Softw*. Epub 2016 Dec 13.
- [31] Castillo PC, Castro PM, Mahalec V. Global optimization algorithm for large-scale refinery planning models with bilinear terms. *Ind Eng Chem Res* 2017;56(2):530–48.
- [32] McCormick GP. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Math Program* 1976;10(1):147–75.
- [33] Karuppiah R, Grossmann IE. Global optimization for the synthesis of integrated water systems in chemical processes. *Comput Chem Eng* 2006;30(4):650–73.
- [34] Castro PM. Tightening piecewise McCormick relaxations for bilinear problems. *Comput Chem Eng* 2015;72:300–11.
- [35] Misener R, Thompson JR, Floudas CA. APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Comput Chem Eng* 2011;35(5):876–92.
- [36] Kolodziej S, Castro PM, Grossmann IE. Global optimization of bilinear programs with a multiparametric disaggregation technique. *J Glob Optim* 2013;57(4):1039–63.
- [37] Castro PM. Normalized multiparametric disaggregation: An efficient relaxation for mixed-integer bilinear problems. *J Glob Optim* 2016;64(4):765–84.
- [38] Castro PM, Grossmann IE. Global optimal scheduling of crude oil blending operations with RTN continuous-time and multiparametric disaggregation. *Ind Eng Chem Res* 2014;53(39):15127–45.
- [39] Castro PM. Source-based discrete and continuous-time formulations for the crude oil pooling problem. *Comput Chem Eng* 2016;93:382–401.
- [40] Castillo PAC, Mahalec V, Kelly JD. Inventory pinch algorithm for gasoline blend planning. *AIChE J* 2013;59(10):3748–66.
- [41] Healy WC, Maassen CW, Peterson RT. A new approach to blending octanes. In: *Proceedings of the 24th Midyear Meeting of American Petroleum Institute's Division of Refining*; 1959 May 27; New York, US; 1959. p. 132–136.
- [42] Castro PM, Grossmann IE. Optimality-based bound contraction with multiparametric disaggregation for the global optimization of mixed-integer bilinear problems. *J Glob Optim* 2014;59(2):277–306.
- [43] Kallrath J. Planning and scheduling in the process industry. *OR Spectrum* 2002;24(1):219–250.

Chapter 9: Global Optimization of MIQCPs with Dynamic Piecewise Relaxations


This chapter has been published online in the Journal of Global Optimization. Complete citation:

Castillo Castillo, P. A., Castro, P. M., & Mahalec, V. (2017). Global optimization of MIQCPs with dynamic piecewise relaxations. *Journal of Global Optimization*. doi: 10.1007/s10898-018-0612-7.

In Chapter 9, an improved version of the algorithm described in Chapter 7 is presented. Compared to the method detailed in Chapter 7, this new version of the algorithm uses optimality-based bound tightening not only when a new upper bound is found, but whenever the domain of the variables is significantly reduced. In addition, the algorithm also increases or decreases the number of partitions depending on the last iteration performance, which is defined by the required execution time, optimality gap improvement, and average domain reduction. Finally, the algorithm can switch from piecewise McCormick to Normalized Multiparametric Disaggregation when the number of partitions is greater or equal to 10.

The test examples include the refinery planning problems from Chapter 6, and 3 scheduling problems of a hydro energy system. The results show that this version of the algorithm is superior to that from Chapter 6.

Global optimization of MIQCPs with dynamic piecewise relaxations

Pedro A. Castillo Castillo¹ · Pedro M. Castro²  · Vladimir Mahalec¹

Received: 2 June 2017 / Accepted: 23 January 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract We propose a new deterministic global optimization algorithm for solving mixed-integer bilinear programs. It relies on a two-stage decomposition strategy featuring mixed-integer linear programming relaxations to compute estimates of the global optimum, and constrained non-linear versions of the original non-convex mixed-integer nonlinear program to find feasible solutions. As an alternative to spatial branch-and-bound with bilinear envelopes, we use extensively piecewise relaxations for computing estimates and reducing variable domain through optimality-based bound tightening. The novelty is that the number of partitions, a critical tuning parameter affecting the quality of the relaxation and computational time, increases and decreases dynamically based on the computational requirements of the previous iteration. Specifically, the algorithm alternates between piecewise McCormick and normalized multiparametric disaggregation. When solving ten benchmark problems from the literature, we obtain the same or better optimality gaps than two commercial global optimization solvers.

Keywords Mixed-integer nonlinear programming · Global optimization of quadratic programs with bilinear terms · Piecewise linear relaxations · Optimality-based bound tightening

1 Introduction

We aim to solve a special class of nonconvex mixed-integer nonlinear programming (MINLP) problems to ε -global optimality, where ε is a non-zero tolerance. Problem **P** is a mixed-integer quadratically constrained problem (MIQCP) where nonlinearities are due to bilinear terms

✉ Pedro M. Castro
pmcastro@fc.ul.pt

¹ Department of Chemical Engineering, McMaster University, Hamilton, ON L8S 4A7, Canada

² Centro de Matemática Aplicações Fundamentais e Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisbon, Portugal

$x_i x_j$ of continuous variables x with finite lower x^L and upper x^U bounds, and binary variables y appear linearly in the constraints:

$$\begin{aligned}
 & f_{\mathbf{P}}^* = \min f_0(x, y) \\
 & \text{s.t. } f_q(x, y) \leq 0 \quad \forall q \in \mathbf{Q} / \{0\} \\
 & f_q(x, y) = \sum_{(i,j) \in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y + d_q \quad \forall q \in \mathbf{Q} \\
 & w_{ij} = x_i x_j \quad \forall (i, j) \in \mathbf{BL} \\
 & 0 \leq x^L \leq x \leq x^U \\
 & x \in \mathbb{R}^{lx}, \quad y \in \{0, 1\}^{ly}, \quad w \in \mathbb{R}^{|\mathbf{BL}|}
 \end{aligned} \tag{P}$$

\mathbf{BL} is an (i, j) -index set defining all bilinear terms, \mathbf{Q} represents the set of all functions appearing in the constraints and objective function ($q = 0$), which excludes auxiliary equations defining new sets of bilinear variables w . The total number of original continuous variables is lx , while the number of original binary variables is ly . We assume that \mathbf{P} is feasible with global optimal solution $f_{\mathbf{P}}^*$.

Many relevant engineering problems can be formulated as \mathbf{P} . Others, closely resemble \mathbf{P} , the difference being the presence of constraints with exponential terms for estimating the capital cost. Examples can be found in pooling problems [1–4], synthesis of general multi-component process networks [5,6], design of water networks [7–11], short-term planning of oil refineries [12,13], scheduling of crude-oil blending operations [14–16] and hydro energy systems [17].

Nonconvex optimization problem \mathbf{P} can present multiple local and global optima. Gradient-based methods cannot guarantee finding a global solution and they do not tell, at termination, how far the best feasible solution is from the best possible solution (i.e., the best estimate of the global optimum) [18]. Deterministic global optimization algorithms are required for such purposes, with their development being a very active research area.

Deterministic global optimization algorithms rely on a relaxation of \mathbf{P} to compute estimates of the global solution, on various techniques to iteratively improve such estimates, and on methods to compute feasible solutions. They aim to reduce the relative difference between the best feasible and best possible solutions below ε . The quality of the best possible solution depends on the tightness of the relaxation, which in turn depends on the size of the domain of the variables (i.e., $x^U - x^L$). The smaller the domain, the closer the relaxation is to the original nonconvex function.

Spatial branch-and-bound (SBB) is the most common method to systematically reduce the domain of the variables. In SBB, branching is applied on discrete variables, as well as on continuous variables involved in nonlinear terms [19,20]. Branching occurs one variable at a time and it generates two child nodes, each with a smaller domain than the parent node, leading to potentially tighter relaxations. Whenever the best possible solution at a node becomes worse than the best feasible solution, the node is fathomed (pruned). Cutting planes and bound-tightening techniques have been incorporated in SBB algorithms to improve the relaxation and reduce the number of nodes to explore. Although SBB guarantees convergence to an ε -global solution, computational time can grow exponentially with problem size.

The tightest linear relaxation for a bilinear term is given by McCormick envelopes [21]. They are generated by four inequalities that have a low computational cost. Many deterministic global optimization algorithms employ McCormick envelopes as their only relaxation technique for bilinear terms [5, 12, 22, 23]. However, McCormick envelopes usually provide a weak relaxation when at least one of the variables involved in a bilinear term has a significantly large domain. This led to the development of piecewise linear relaxation techniques

that improve the quality of the relaxation by partitioning the variables domain (the larger the number of partitions, the tighter the relaxation). However, since piecewise linear relaxations introduce additional binary and continuous variables, there exists a trade-off between tightness and the computational effort required to solve the MILP to optimality.

The piecewise McCormick relaxation [24–27] partitions the domain of one of the variables involved in a bilinear term and constructs McCormick envelopes for each partition. The major drawback of piecewise McCormick is that the number of additional binary variables increases linearly with the number of partitions. This important issue fostered the development of relaxation techniques where the number of binary variables increases logarithmically with respect to the number of partitions [2, 28, 29], with one example being normalized multiparametric disaggregation. Piecewise linear relaxations have been used in SBB algorithms [2, 30, 31], but they can also be deployed as an alternative to the SBB framework [13, 28, 32–34], as well as in decomposition methods [26].

The semidefinite programming (SDP) relaxation of MIQCPs has also been extensively studied. Problem **(P)** is a lifted reformulation, with extra variables w_{ij} and non-convex constraints $w_{ij} = x_i x_j$. It can be relaxed as a pair of inequalities, $W - xx^T \succeq 0$ (convex) and $xx^T - W \succeq 0$ (non-convex). In order to produce strong convex relaxations, Saxena et al. [35] use the convex SDP inequality to derive convex quadratic cuts and exploit the non-convex inequality to derive disjunctive cuts. Their cutting plane algorithm also relies on McCormick envelopes to strengthen the initial relaxation of the MIQCP, later removing all non-binding (at the solution of the convex relaxation) RLT inequalities when generating disjunctive cuts. In the companion paper [36], Saxena et al. study methods that capture the strength of such extended SDP relaxations but are defined only in the space of the x variables. By replacing the RLT convexification of **(P)** with an alternative that splits matrix A , defining bilinear terms $x^T A x$, as a difference of positive semidefinite and symmetric matrices, they show how to project the extended RLT formulation in the original space by solving linear programs (LPs). A similar procedure is performed when adding the convex inequality $W - xx^T \succeq 0$ to the extended RLT formulation, leading to the solution of SDPs rather than LPs. For the GLOBALlib instances, relaxations from projected formulations are almost as strong as those from [35], with the advantage of being solved two orders of magnitude faster.

In this work, we present a deterministic global optimization algorithm to solve MINLP problems of type **P**. The main novelty is the use of a dynamic partitioning scheme for piecewise relaxations, not only to compute the lower bound, but also for performing optimality-based bound tightening (OBBT) for all variables appearing in bilinear terms [33]. The extensive use of OBBT contrasts with commercial global optimization solvers [37–39], which apply some restricted version of it, always featuring the simplest bilinear envelopes [40]. Dynamic partitioning refers to changing the number of partitions between iterations. Although the same term has been applied in [34], there are major differences between the two algorithms, as can be seen in Fig. 1.

Nagarajan et al. [34] assume that a local solution (x^*, y^*, w^*) to **(P)** is given and divide their global optimization algorithm in two parts. In part one, a sequence of OBBT iterations is performed to reduce the domain of all x variables. The procedure stops when bound improvement in consecutive iterations falls below a specified tolerance. Part two involves the solution of relaxation problems **(PR)** derived from piecewise McCormick envelopes. The domain of bilinearly appearing variables x_i and x_j is partitioned in a non-uniform way. Partitions are dynamically added around the current solution (local solution x^* in the first iteration and optimal solution from the relaxation problem x^R in subsequent iterations) until the normalized improvement on the lower bound LB from **(PR)** is less than a given tolerance,

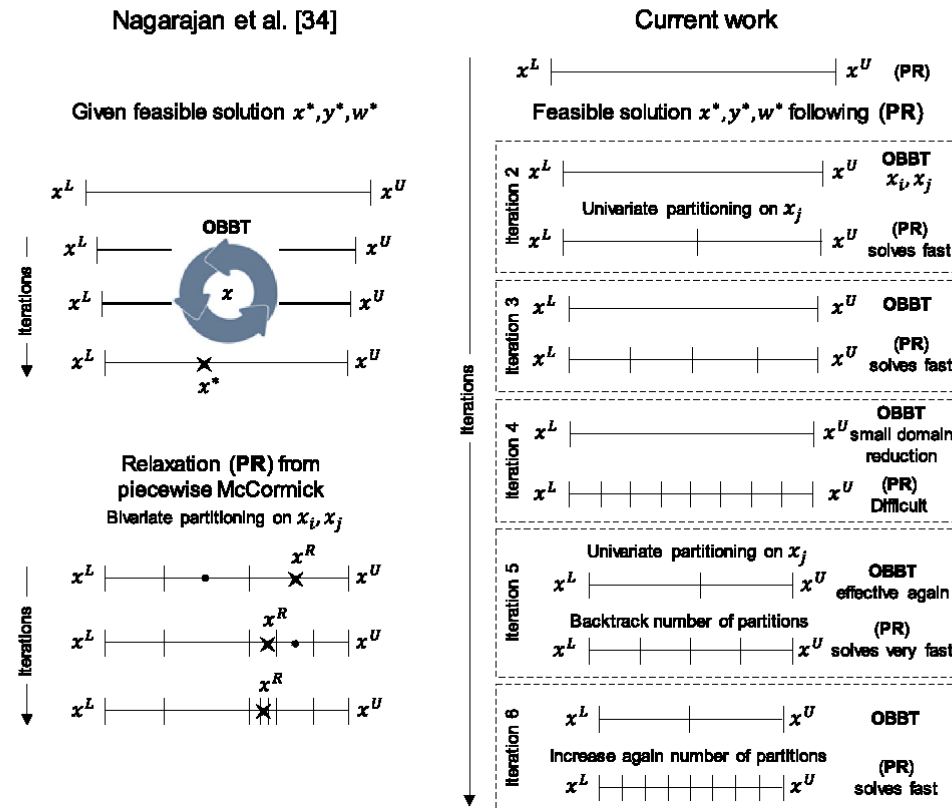


Fig. 1 Comparison of dynamic partitioning schemes

x^R variables remain in the same partitions and the size of such partitions is already very small, or the computation hits a time limit.

The algorithm proposed in this work does not assume a feasible solution is given. In the first iteration, it solves a simple relaxation problem (McCormick envelopes) to try to find one very quickly. This process is repeated in subsequent iterations to improve the upper bound UB, and consequently the bounds from OBBT (step omitted from Fig. 1 since the focus is on comparing the lower bounding procedure). The integration of OBBT and (PR) steps is the first major difference compared to [34]. The second difference is that our algorithm relies on univariate and uniform partitioning. Uniform partitioning, by giving the same importance to all regions of the domain, may protect us against frequent x^R movements from narrower to wider partitions, meaning potentially fewer iterations at the expense of more partitions (larger problems) per iteration. The next partitioning level is decided based on (PR)'s computational requirements. Figure 1 assumes (PR) solves fast before reaching $N = 8$ partitions in iteration 4, coinciding with OBBT becoming ineffective. To further improve domain reduction, it is thus worth to try piecewise relaxation strategies for OBBT, not considered in [34], by selecting $N = 2$. Iteration 5 also backtracks to $N = 4$ for (PR), illustrating that dynamic partitioning can go in both directions. Finally, and to benefit from the better scaling of problem size with the number of partitions when reaching $N = 10$, the algorithm will change the relaxation technique from piecewise McCormick to normalized multiparametric disaggregation.

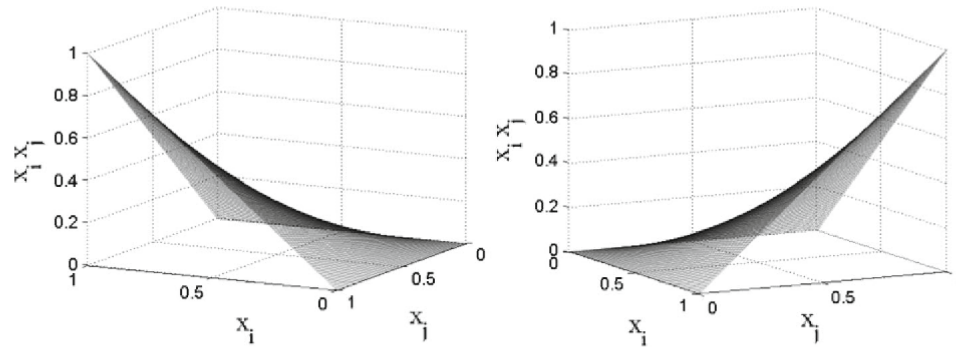


Fig. 2 Bilinear function $x_i x_j$ in $[0, 1]^2$

2 Computing lower bounds

If y variables remain binary and all original constraints $q \in \mathbf{Q} \setminus \{0\}$ are kept, the simplest relaxation of \mathbf{P} is obtained by removing equations $w_{ij} = x_i x_j$. However, it is also the weakest. Narrowing the domain of variables w_{ij} to regions \mathbf{WR}_{ij} will potentially lead to a tighter relaxation. Since \mathbf{P} is feasible, so is its relaxation \mathbf{PR} . If $f_{\mathbf{PR}}^R$ is the global optimal solution of \mathbf{PR} , then $f_{\mathbf{PR}}^R \leq f_{\mathbf{P}}^*$, representing a lower bound on the global optimal solution of \mathbf{P} .

$$\begin{aligned}
 & f_{\mathbf{PR}}^R = \min f_0(x, y) \\
 & \text{s.t. } f_q(x, y) \leq 0 \quad \forall q \in \mathbf{Q} \setminus \{0\} \\
 & f_q(x, y) = \sum_{(i,j) \in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y + d_q \quad \forall q \in \mathbf{Q} \\
 & w_{ij} \in \mathbf{WR}_{ij} \quad \forall (i, j) \in \mathbf{BL} \\
 & 0 \leq x^L \leq x \leq x^U \\
 & x \in \mathbb{R}^{lx}, \quad y \in \{0, 1\}^{ly}, \quad w \in \mathbb{R}^{|\mathbf{BL}|}
 \end{aligned} \tag{PR}$$

Three alternative ways of defining regions \mathbf{WR}_{ij} for relaxation problem \mathbf{PR} will be discussed next.

2.1 McCormick relaxation (SMCR)

The standard McCormick relaxation for bilinear function $w_{ij} = x_i x_j$ represented in Fig. 2, is given by Eqs. (1–4). These equations define regions \mathbf{WR}_{ij} that form the convex hull for $x_i x_j$, see Fig. 3.

$$w_{ij} \geq x_i x_j^L + x_j x_i^L - x_i^L x_j^L \quad \forall (i, j) \in \mathbf{BL} \tag{1}$$

$$w_{ij} \geq x_i x_j^U + x_j x_i^U - x_i^U x_j^U \quad \forall (i, j) \in \mathbf{BL} \tag{2}$$

$$w_{ij} \leq x_i x_j^L + x_j x_i^U - x_i^U x_j^L \quad \forall (i, j) \in \mathbf{BL} \tag{3}$$

$$w_{ij} \leq x_i x_j^U + x_j x_i^L - x_i^L x_j^U \quad \forall (i, j) \in \mathbf{BL} \tag{4}$$

2.2 Piecewise linear relaxations

Piecewise McCormick and normalized multiparametric disaggregation are well-known examples of piecewise relaxation techniques that typically use the same number of parti-

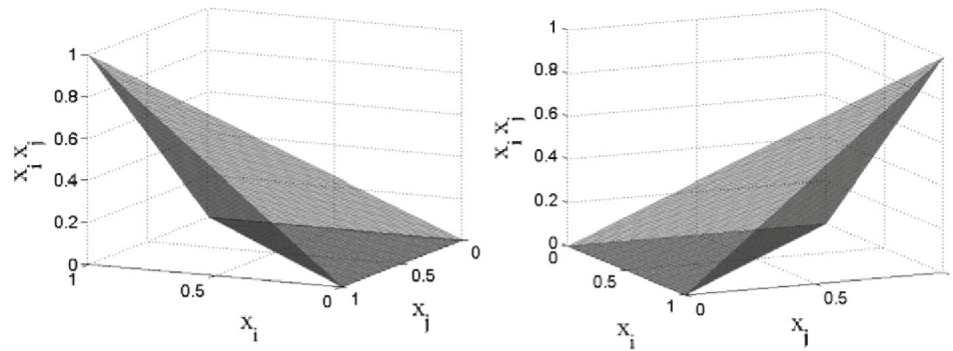


Fig. 3 Feasible region from McCormick envelopes for bilinear function $x_i x_j$ in $[0, 1]^2$

tions N for every partitioned variable x_j . Both introduce additional binary variables into the problem, creating non-convex regions \mathbf{WR}_{ij} .

2.2.1 Piecewise McCormick relaxation (PMCR)

Piecewise McCormick uses binary variable z_{jn} to identify the active (n) partition for variable x_j . The McCormick envelopes in Eqs. (1–4) can then benefit from tighter bounds $x_j^L \leq x_{jn}^L$ and $x_{jn}^U \leq x_j^U$, computed by Eqs. (5–6). The mixed-integer linear relaxation can be formulated as a disjunction and convex-hull reformulated [41] into Eqs. (7–15). Notice the new continuous disaggregated variables \hat{x}_{jn} and \hat{x}_{ijn} . The feasible region associated to PMCR using 4 partitions is illustrated in Fig. 4. Notice that it is closer to the original bilinear function (Fig. 2) than SMCR (Fig. 3).

$$x_{jn}^L = x_j^L + \frac{(x_j^U - x_j^L)(n-1)}{N} \quad \forall j : (i, j) \in \mathbf{BL}, n \in \{1, \dots, N\} \tag{5}$$

$$x_{jn}^U = x_j^L + \frac{(x_j^U - x_j^L)(n)}{N} \quad \forall j : (i, j) \in \mathbf{BL}, n \in \{1, \dots, N\} \tag{6}$$

$$w_{ij} \geq \sum_{n=1}^N (\hat{x}_{ijn} x_{jn}^L + \hat{x}_{jn} x_i^L - z_{jn} x_i^L x_{jn}^L) \quad \forall (i, j) \in \mathbf{BL} \tag{7}$$

$$w_{ij} \geq \sum_{n=1}^N (\hat{x}_{ijn} x_{jn}^U + \hat{x}_{jn} x_i^U - z_{jn} x_i^U x_{jn}^U) \quad \forall (i, j) \in \mathbf{BL} \tag{8}$$

$$w_{ij} \leq \sum_{n=1}^N (\hat{x}_{ijn} x_{jn}^L + \hat{x}_{jn} x_i^U - z_{jn} x_i^U x_{jn}^L) \quad \forall (i, j) \in \mathbf{BL} \tag{9}$$

$$w_{ij} \leq \sum_{n=1}^N (\hat{x}_{ijn} x_{jn}^U + \hat{x}_{jn} x_i^L - z_{jn} x_i^L x_{jn}^U) \quad \forall (i, j) \in \mathbf{BL} \tag{10}$$

$$x_i = \sum_{n=1}^N \hat{x}_{ijn} \quad \forall (i, j) \in \mathbf{BL} \tag{11}$$

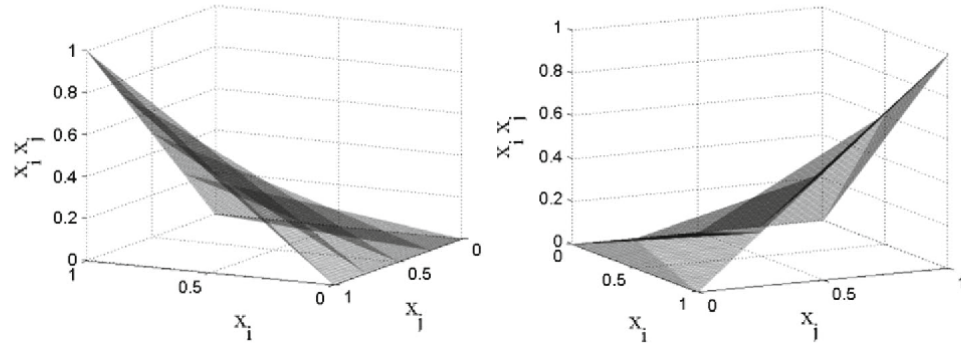


Fig. 4 Feasible region from piecewise McCormick relaxation with 4 partitions for bilinear term $x_i x_j$ in $[0, 1]^2$

$$x_j = \sum_{n=1}^N \hat{x}_{jn} \quad \forall j : (i, j) \in \mathbf{BL} \tag{12}$$

$$\sum_{n=1}^N z_{jn} = 1 \quad \forall j : (i, j) \in \mathbf{BL} \tag{13}$$

$$x_i^L z_{jn} \leq \hat{x}_{ijn} \leq x_i^U z_{jn} \quad \forall (i, j) \in \mathbf{BL}, n \in \{1, \dots, N\} \tag{14}$$

$$x_{jn}^L z_{jn} \leq \hat{x}_{jn} \leq x_{jn}^U z_{jn} \quad \forall j : (i, j) \in \mathbf{BL}, n \in \{1, \dots, N\} \tag{15}$$

2.2.2 Normalized multiparametric disaggregation (NMDT)

Normalized multiparametric disaggregation provides an equivalent relaxation to PMCR but can be orders of magnitude more efficient computationally. However, the number of partitions is restricted to powers of ten, i.e. $N = 10^{-p}$, with $p \in \mathbb{Z}^-$ being the accuracy parameter chosen by the user. The normalized $[0, 1]$ domain of variable x_j is discretized considering all digits $k \in \{0, \dots, 9\}$ of the decimal representation system and positions $l \in \{p, \dots, -1\}$. It is then linked to the real domain of x_j through continuous variable λ_j and global bounds x_j^L and x_j^U . Note that continuous variable $\Delta\lambda_j$ allows λ_j to take continuous values between discrete points. The active partition for x_j is identified by the non-zero values of $(-p)$ binary variables z_{jkl} , one per position l . The number of binary variables per variable is thus $10 \log_{10} N$ versus N when using PMCR. Equations (16–26) provide the NMDT relaxation that also requires continuous variables v_{ij} , Δv_{ij} , and \hat{x}_{ijkl} . It is illustrated in Fig. 5 for $p = -1$ ($N = 10$), which is already very similar to Fig. 2.

$$w_{ij} = x_i x_j^L + v_{ij} (x_j^U - x_j^L) \quad \forall (i, j) \in \mathbf{BL} \tag{16}$$

$$x_j = x_j^L + \lambda_j (x_j^U - x_j^L) \quad \forall j : (i, j) \in \mathbf{BL} \tag{17}$$

$$\lambda_j = \Delta\lambda_j + \sum_{l=p}^{-1} \sum_{k=0}^9 10^l \cdot k \cdot z_{jkl} \quad \forall j : (i, j) \in \mathbf{BL} \tag{18}$$

$$0 \leq \Delta\lambda_j \leq 10^p \quad \forall j : (i, j) \in \mathbf{BL} \tag{19}$$

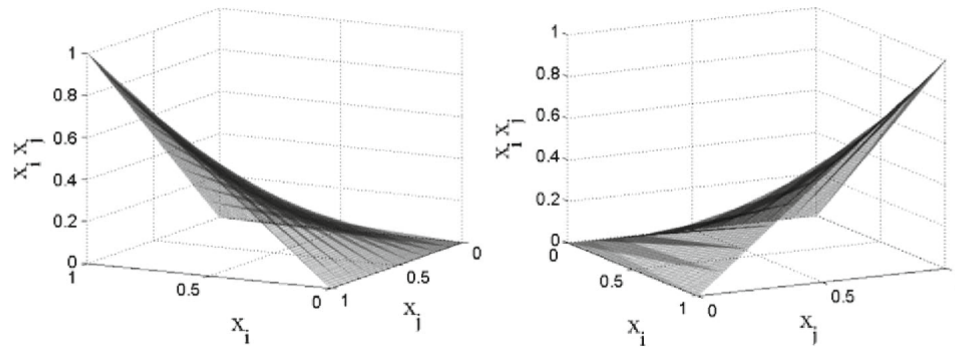


Fig. 5 Feasible region from normalized multiparametric disaggregation with $p = -1$ for bilinear term $x_i x_j$ in $[0, 1]^2$

$$v_{ij} = \sum_{l=p}^{-1} \sum_{k=0}^9 10^l \cdot k \cdot \hat{x}_{ijkl} + \Delta v_{ij} \quad \forall (i, j) \in \mathbf{BL} \tag{20}$$

$$x_i^L \Delta \lambda_j \leq \Delta v_{ij} \leq x_i^U \Delta \lambda_j \quad \forall (i, j) \in \mathbf{BL} \tag{21}$$

$$\Delta v_{ij} \leq 10^p (x_i - x_i^L) + x_i^L \Delta \lambda_j \quad \forall (i, j) \in \mathbf{BL} \tag{22}$$

$$\Delta v_{ij} \geq 10^p (x_i - x_i^U) + x_i^U \Delta \lambda_j \quad \forall (i, j) \in \mathbf{BL} \tag{23}$$

$$x_i = \sum_{k=0}^9 \hat{x}_{ijkl} \quad \forall (i, j) \in \mathbf{BL}, l \in \{p, \dots, -1\} \tag{24}$$

$$\sum_{k=0}^9 z_{jkl} = 1 \quad \forall j : (i, j) \in \mathbf{BL}, l \in \{p, \dots, -1\} \tag{25}$$

$$x_i^L z_{jkl} \leq \hat{x}_{ijkl} \leq x_i^U z_{jkl} \quad \forall (i, j) \in \mathbf{BL}, l \in \{p, \dots, -1\}, k \in \{0, \dots, 9\} \tag{26}$$

3 Optimality-based bound tightening (OBBT)

For all three relaxation techniques described in Sect. 2, the volume of region \mathbf{WR}_{ij} depends on bounds x_i^L, x_i^U, x_j^L and x_j^U . It is thus desirable to strengthen such bounds (raise x_i^L and x_j^L , and decrease x_i^U and x_j^U) to obtain a tighter relaxation (higher $f_{\mathbf{PR}}^R$). One way to do it, is through optimality-based bound tightening (OBBT). For each variable $h \in \mathbf{BLV} = \{h | (i, j) \in \mathbf{BL} \wedge (h = i \vee h = j)\}$ involved in a bilinear term, lower and upper bounds are computed by solving one minimization and one maximization problem, respectively. These problems, denoted as \mathbf{PRB} , are like relaxation problem \mathbf{PR} but with a different objective function (now the variable to minimize/maximize) and an additional constraint, which imposes the value of the objective function in \mathbf{P} , $f_0(x, y)$, to be less or equal than the current upper bound UB .

$$\begin{aligned}
 & x_h^L = \min x_h \quad (x_h^U = \max x_h) \\
 & \text{s.t. } f_0(x, y) \leq UB \\
 & f_q(x, y) \leq 0 \quad \forall q \in \mathbf{Q} / \{0\} \\
 & f_q(x, y) = \sum_{(i,j) \in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y + d_q \quad \forall q \in \mathbf{Q} \\
 & w_{ij} \in \mathbf{WR}_{ij} \quad \forall (i, j) \in \mathbf{BL} \\
 & 0 \leq x^L \leq x \leq x^U \\
 & x \in \mathbb{R}^{lx}, y \in \mathcal{Y}, w \in \mathbb{R}^{|\mathbf{BL}|}
 \end{aligned} \tag{PRB}$$

Remark 1 Given that many problems may need to be solved, the complexity of problems **PRB** should be manageable. Region \mathbf{WR}_{ij} will be generated from either the standard or piecewise McCormick envelopes with a low number of partitions ($N < 10$). With the former, binary variables are further relaxed, $\mathcal{Y} \in [0, 1]^{ly}$, to work with linear problems (LPs) instead of MILPs ($\mathcal{Y} \in \{0, 1\}^{ly}$).

Other types of probing methods can be found in the literature that also solve bounded relaxations of the problem to extract further information on the variables, e.g. to identify conflicts between binary variables y [42]. They are not part of this work.

4 Generating upper bounds

Previous work has shown that an effective way to compute a good feasible solution to non-convex MINLP problem **P**, is to rely on a two-stage MILP/NLP strategy. Any feasible solution to MILP problem **PR** can be used to extract the values x^R, y^R and w^R of variables x, y and w . Parameters y^R will then replace binary variables y in **P**, reducing it to NLP problem **PF**. **PF** will be solved by a local NLP solver, after initializing variables x and w with parameters x^R and w^R , to facilitate convergence. Note that **PF** is a restricted version of **P**, and so it is not necessarily feasible. If feasible, the optimal solution (x^*, y^*, w^*) of **PF** is an upper bound UB on the global solution of **P**, i.e. $f_{\mathbf{PF}}^* \geq f_{\mathbf{P}}^*$.

$$\begin{aligned}
 & f_{\mathbf{PF}}^* = \min f_0(x) \\
 & \text{s.t. } f_q(x) \leq 0 \quad \forall q \in \mathbf{Q} / \{0\} \\
 & f_q(x) = \sum_{(i,j) \in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y^R + d_q \quad \forall q \in \mathbf{Q} \\
 & w_{ij} = x_i x_j \quad \forall (i, j) \in \mathbf{BL} \\
 & 0 \leq x^L \leq x \leq x^U \\
 & x \in \mathbb{R}^{lx}, w \in \mathbb{R}^{|\mathbf{BL}|}
 \end{aligned} \tag{PF}$$

5 Global optimization algorithm

We now propose a global optimization algorithm for the solution of any mixed-integer non-linear program that can be written as problem **P**. It is summarized in Fig. 6 and detailed in Tables 1 and 2.

Assumed given are the selection of partitioned variables x_j in every bilinear term, variable bounds x^L and x^U , and a variety of tuning parameters. Problem-specific settings include the pre-specified values that the number of partitions can take when solving **PR** (N_{PR}) and **PRB** (N_{PRB}), maximum computational time and relative optimality tolerance (e.g. $\text{TIME}_{PR}^{max}, \varepsilon_{PR}$). The other parameters will be named while describing the algorithm.

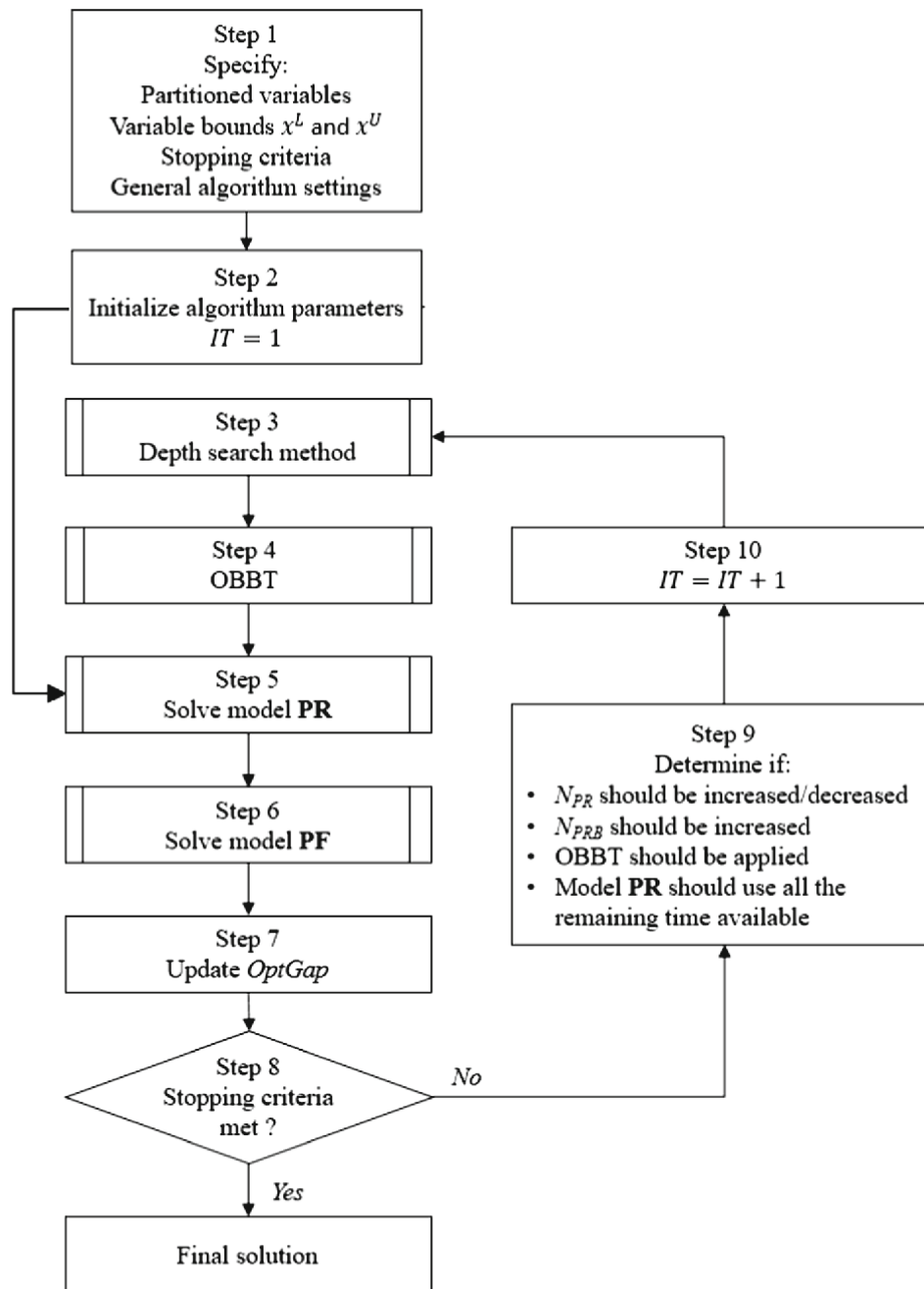


Fig. 6 Flowchart of the proposed global optimization algorithm

Following the initialization step, the algorithm computes the lower bound LB using the simplest McCormick relaxation. In subsequent iterations, step 5 will typically involve piecewise linear relaxations. Note that once OBBT loses efficiency (flag $LAST_{PR} = 1$), the maximum computational time $TIME_{PR}^{max}$ will be reset to the remaining time to run the algorithm. Step 5 solves one MILP problem of type **PR**, gathering a maximum of n_{pool} solutions in a pool. If the optimal solution f_{PR}^R is higher than the lower bound, the latter is updated.

Remark 2 Region WR_{ij} in problem **PR** is computed using piecewise McCormick envelopes whenever $N_{PR} \in \{1, 2, \dots, 9\}$. Normalized multiparametric disaggregation is used instead for $N_{PR} \in \{10, 100, 1000, \dots\}$ ($p \in \{\dots, -3, -2, -1\}$).

Remark 3 The lower bound is updated using the best possible solution at termination for problem **PR** and not the best-found feasible solution. The same is true for problem **PRB**, when it is an MILP.

In step 6, we use the values (x^R, y^R, w^R) of the variables in the previous solutions to help computing upper bounds. A total of n_{pool} problems of type **PF** are solved in parallel using n_{par} threads. Amongst those that are feasible, the one with the lowest objective function f_{PF}^* can set the upper bound UB . Note that **PF** is solved by a local NLP solver and so this step is much faster than steps 4–5. It is the reason why no execution-time constraints are enforced.

With the lower and upper bound, step 7 computes the relative optimality gap $OptGap$. Step 8 stops the algorithm if the termination criteria is met, either a relative tolerance below ε or a computational time ($TIME$) above maximum value $TIME^{max}$. Decisions related to the dynamic partitioning scheme are taken in step 9.

The details of step 9 can be found in Table 2. Two flags are used: $NN_{PR}^{nec} = 1$ indicates that we have the necessary conditions for increasing the number of partitions in problem **PR**; $NN_{PR} = 1$ gives the sufficient condition for selecting the next setting in $\{N_{PR,first}, \dots, N_{PR,last}\}$, see 9c. These are the initial values for the first entry in 9a, which checks the time spent solving **PR** ($TIME_{PR}$).

If greater or equal to $TIME_{PR}^{max}$, it means that we should try to backtrack and reduce the number of partitions in the next iteration to reduce the complexity of **PR**, unless we are already in the coarsest setting $N_{PR,first}$ or have previously backtracked to N_{PR} ; either way, we will definitely not increase N_{PR} , i.e. $NN_{PR} = 0$. The same is true if $TIME_{PR}$ is within $TIME_{PR}^{max}$ and the maximum time ratio tr_{PR}^{max} , and $LAST_{PR} = 0$. We also set $NN_{PR}^{nec} = 0$ to later decide how to improve the lower bound.

If the number of partitions did not increase in the previous iteration ($NN_{PR} = 0$) and **PR** was solved rather fast (below minimum time ratio tr_{PR}^{min}), we will try to generate a better lower bound by rising N_{PR} in the next iteration. This concludes step 9a.

Step 9b takes measures when the average domain reduction in OBBT is below the minimum target of ADR^{min} . This is not an issue if **PR** problems can be solved rather fast ($NN_{PR} = 1$), we simply avoid spending time in the next iteration with an inefficient OBBT by making $DO_{OBBT} = 0$. On the other hand, if we do not meet the necessary condition to increase N_{PR} ($NN_{PR}^{nec} = 0$), we may need to move towards termination of the algorithm.

If the next possible value of N_{PRB} is lower than N_{PR} , then we might still be able to get a good domain reduction by increasing N_{PRB} . Counters of LP (C_{PRB}^{LP}) and MILP (C_{PRB}^{MILP}) problems solved are then reset. Else, we increase the appropriate counter by one. We then proceed to the last if-then-else. If we have already solved at least one MILP in OBBT and found that $ADR < ADR^{min}$, then the most reasonable thing to do is to give all remaining time to **PR** by making $LAST_{PR} = 1$. If the domain reduction was low but we have been solving

LPs in OBBT, then we also move towards the end while allowing one more OBBT run, now solving MILPs, after selecting the next value of N_{PRB} .

We then proceed to the next iteration in step 10. Steps 3 and 4 are the first procedures of iteration IT but do not occur in the first iteration to quickly compute an optimality gap.

Step 3 involves a depth search and is detailed in Sect. 5.1.

Step 4 executes optimality-based bound tightening to reduce the variables domain. It is triggered by $DO_{OBBT} = 1$ and involves solving two **PRB** problems per variable, after setting the number of partitions N to N_{PRB} . Since the number of variables involved in bilinear terms can be significantly large, it is much more efficient to solve the multiple instances of problem **PRB** in parallel rather than sequentially (see results in Sect. 7.6). This procedure is repeated until OBBT has been applied on all x_h variables involved in bilinear terms. We then compute the average domain reduction ADR (%) using Eq. (27).

$$ADR = \frac{1}{|\mathbf{BLV}|} \sum_{h \in \mathbf{BLV}} \left[\frac{\left(x_h^{U, previous} - x_h^{L, previous} \right) - \left(x_h^{U, updated} - x_h^{L, updated} \right)}{\left(x_h^{U, previous} - x_h^{L, previous} \right)} \times 100 \right] \tag{27}$$

Remark 4 $N_{PRB} = 0$ triggers the computation of relaxed region \mathbf{WR}_{ij} of bilinear function $w_{ij} = x_i x_j$ from the McCormick envelopes with binary variables relaxed (recall Remark 1).

5.1 Depth search method

MIQCPs have two sources of complexity: (1) a combinatorial source from binary variables; (2) a non-convexity source from bilinear terms. A stronger combinatorial component is associated to a higher difficulty finding the global optimal solution and can be addressed by generating a larger number of feasible solutions for \mathbf{P} . This should be done preferably in the earlier stages of the global optimization algorithm, since a better upper bound (UB) helps to improve the bounds computed by problem **PRB**. It is activated in the second iteration ($IT = 2$) or when $ADR < ADR^{min}$, if general setting $DO_{DS} = 1$.

The depth search method works by dynamically increasing the number of partitions in **PR** from the current N_{PR} value. Note that it is not needed to solve **PR** to optimality since the focus here is not on the lower bound. Because the MILP solver normally finds multiple feasible solutions in the early nodes of the search tree, we stop at time $TIME_{PR}^{max}$. Solutions obtained after solving **PR** with more partitions are potentially better (higher f_{PR}^R), leading to values of the model variables that are closer to the feasible region of \mathbf{P} . As explained in Sect. 4, these values are then used to initialize **PF**, potentially leading to a better UB . The depth search method stops after IT_{DS}^{max} increments in the number of partitions, resetting N_{PR} to its original value.

Overall, depth search is very similar to the search performed by the main algorithm. However, it does not use OBBT and it always increases the number of partitions from one iteration to the next.

6 Benchmark problems

Two different sets of MIQCP benchmark problems from the literature are used to evaluate the performance of the proposed global optimization algorithm.

Table 1 Global optimization algorithm

1. Given
 Selection of non-partitioned x_i and partitioned variables x_j for every bilinear term in \mathbf{P}
 Variable bounds: x^L and x^U
 Settings for solving PR: $N_{PR} \in \{N_{PR,first}, \dots, N_{PR,last}\}$, $TIME_{PR}^{max}$, ε_{PR} , n_{pool}
 Settings for solving PRB: $N_{PRB} \in \{N_{PRB,first}, \dots, N_{PRB,last}\}$, $TIME_{PRB}^{max}$, ε_{PRB} , n_{par}
 General settings: $TIME^{max}$, ε , ADR^{min} , DO_{DS} , tr_{PR}^{min} , tr_{PR}^{max} , IT_{DS}^{max}

2. Initialization
 $LB = -\infty$, $UB = +\infty$, $IT = 1$, $IT_{DS} = 1$, $TIME = 0$, $ADR = ADR^{min} + 0.1$, $DO_{OBBT} = 1$,
 $LAST_{PR} = 0$, $N_{PR} = N_{PR,first}$, $N_{PRB} = N_{PRB,first}$, $NN_{PR}^{nec} = 1$, $NN_{PR} = 1$, $C_{PRB}^{LP} = 0$, $C_{PRB}^{MILP} = 0$.

3. Depth search method (see Section 5.1)
 If $DO_{DS} = 1$ and $IT > 1$,
 If $IT = 2$ or $ADR < ADR^{min}$,
 Generate up to IT_{DS}^{max} feasible solutions; if best is better than UB , update UB ; $update(TIME)$

4. Optimality-based bound tightening (OBBT)
 If $IT > 1$ and $DO_{OBBT} = 1$,
 $N = N_{PRB}$
 For every bilinear appearing variable x_h and using n_{par} threads
 Solve PRB minimizing x_h up to $TIME_{PRB}^{max}$ to obtain \underline{x}_h ; $x_h^L = \max(x_h^L, \underline{x}_h)$
 Solve PRB maximizing x_h , up to $TIME_{PRB}^{max}$ to obtain \bar{x}_h ; $x_h^U = \min(x_h^U, \bar{x}_h)$
 Compute average domain reduction ADR ; $update(TIME)$

5. Lower bound computation
 If $LAST_{PR} = 1$,
 $TIME_{PR}^{max} = TIME^{max} - TIME$
 $N = N_{PR}$
 If $N_{PR} \geq 10$,
 Solve problem PR with WR_{ij} from NMDT up to $TIME_{PR}^{max}$, storing n_{pool} feasible solutions
 Else,
 Solve problem PR with WR_{ij} from PMCR up to $TIME_{PR}^{max}$, storing n_{pool} feasible solutions
 $LB = \max(LB, f_{PR}^R)$; $update(TIME_{PR})$; $update(TIME)$.

6. Upper bound computation
 For every (x^R, y^R, w^R) solution in n_{pool} and using n_{par} threads,
 Initialize (x, w) with (x^R, w^R) .
 Solve problem PF.
 If feasible and $f_{PF}^* < UB$, $UB = f_{PF}^*$, update optimal solution (x^*, y^*, w^*)
 $update(TIME)$

7. Optimality gap computation
 $OptGap = \lceil \frac{UB-LB}{UB} \rceil \times 100$.

8. Check termination criteria
 Stop if $OptGap \leq \varepsilon$ or $TIME \geq TIME^{max}$

9. Modify number of partitions
 See details in Table 2

10. Continue to next iteration
 $IT = IT + 1$; go to Step 3

Table 2 Global optimization algorithm—dynamic partitioning scheme

9. Modify number of partitions
 $NN_{PR} = NN_{PR}^{nec}$
a. Check time spent solving problem PR
 If $NN_{PR} = 1$,
 If $TIME_{PR} \geq TIME_{PR}^{max}$,
 If $N_{PR} \neq N_{PR,first}$ and $N_{PR} \notin N_{PR}^{back}$,
 $N_{PR}^{back} = N_{PR}^{back} \cup N_{PR}$
 $N_{PR} = previous(N_{PR})$
 $NN_{PR}^{nec} = 0; NN_{PR} = 0$.
 ElseIf $TIME_{PR} \geq t_{PR}^{max} \cdot TIME_{PR}^{max}$,
 If $LAST_{PR} = 0$,
 $NN_{PR}^{nec} = 0; NN_{PR} = 0$.
 ElseIf $NN_{PR} = 0$,
 If $TIME_{PR} \leq t_{PR}^{min} \cdot TIME_{PR}^{max}$,
 $NN_{PR}^{nec} = 1; NN_{PR} = 1$.
b. Check average domain reduction
 If $NN_{PR} = 1$ and $ADR < ADR^{min}$,
 $DO_{OBBT} = 0$
 If $NN_{PR}^{nec} = 0$,
 If $next(N_{PRB}) < N_{PR}$,
 $N_{PRB} = next(N_{PRB})$
 $C_{PRB}^{LP} = 0; C_{PRB}^{MILP} = 0$.
 Else,
 If $N_{PRB} = 0$,
 $C_{PRB}^{LP} = C_{PRB}^{LP} + 1$
 Else,
 $C_{PRB}^{MILP} = C_{PRB}^{MILP} + 1$
 If $N_{PRB} \geq 1$,
 If $C_{PRB}^{MILP} \geq 1$,
 If $ADR < ADR^{min}$,
 $NN_{PR}^{nec} = 1, NN_{PR} = 0, LAST_{PR} = 1$, and $DO_{OBBT} = 0$
 Else,
 If $C_{PRB}^{LP} \geq 2$,
 If $ADR < ADR^{min}$,
 $NN_{PR}^{nec} = 1, NN_{PR} = 0, LAST_{PR} = 1$, and $DO_{OBBT} = 1$
 $N_{PRB} = next(N_{PRB})$
c. Partitions of problem PR
 If $NN_{PR} = 1$,
 $N_{PR} = next(N_{PR})$

The first set deals with the short-term scheduling of a hydroelectric system [17], where the aim is to maximize the daily profit considering hourly changing electricity prices and start-up costs for the power plants. Power generation is modelled as a bilinear function of discharge flowrate and head change, with binary variables identifying if a plant is producing energy on a given hour (needed to enforce lower and upper bounds on power production and discharge flowrate) and startups. Like in our previous global optimization studies [31,33], we consider the original problem with 7 reservoirs (HYD7) and simpler versions with 2 (HYD2) and 4 reservoirs (HYD4).

Table 3 MIQCP model statistics

Benchmark problem	HYD2	HYD4	HYD7	SC1TP1–SC3TP1	SC1TP3–SC4TP3
Equations	573	1145	2003	1504	4526
Binary variables	96	192	336	12	36
Total variables	433	865	1513	1234	3716
Variables in bilinear terms	118	260	473	342	1132
Bilinear terms	192	384	672	476	1608

The second set consists of planning problems from a petroleum refinery [13]. The objective is to minimize the total operating cost of the system that includes processing units with alternative operating modes and storage tanks. Binary variables identify active modes and products being blended. Bilinear terms appear as the product of volumetric flows and quality properties in the material balances. We solve seven problems with different crude-oil supply and product demand data. Three involve a single period of operation (SC1TP1-SC3TP1), while in the others, the weekly time horizon is divided in three periods of fixed length (SC1TP3-SC4TP3).

The model statistics in Table 3 show that the ratio between the number of binary variables and bilinear terms varies significantly between the two sets of problems (1:2 vs. 1:50). For the hydro problems, a stronger combinatorial component is associated to a higher difficulty finding the global optimal solution and can be addressed by generating a larger number of feasible solutions of \mathbf{P} . We thus activate the depth search method ($DO_{DS} = 1$), with a maximum of five increments in the number of partitions ($IT_{DS}^{max} = 5$). The refinery problems do not benefit from the time-consuming depth search method and so $DO_{DS} = 0$.

6.1 Tuning parameters

The optimization algorithm presented in Sect. 5 has a few parameters affecting its performance. Most of the values selected were independent of problem type, while one was tuned to adjust to instance size. It is beyond the scope of this paper to present a thorough computational study involving such parameters.

MILP problems \mathbf{PR} were solved for a number of partitions $N_{PR} \in \{1, 2, 4, 8, 10, 100, 1000\}$. The termination criteria were either a relative optimality tolerance $\varepsilon_{PR} = 0.0001\%$ or a maximum time $TIME_{PR}^{max}$ equal to: 400s while OBBT is effective; or the remaining time available, otherwise. The number of partitions N_{PR} will increase in the next iteration if the time solving PR divided by $TIME_{PR}^{max}$ is less or equal than $tr_{PR}^{min} = 0.05$. On the other hand, if the time ratio is greater or equal than $tr_{PR}^{max} = 0.75$, N_{PR} will not change. The solution pool option of the MILP solver was active, with a pool capacity of $n_{pool} = 60$, thus leading to a maximum of 60 instances of \mathbf{PF} solved in parallel per iteration.

The OBBT step involves solving LPs, $N_{PRB} = 0$, and MILP problems, $N_{PRB} \in \{2, 3, 4, 5, 6, 7\}$ (recall Remark 4). In the latter case, the relative tolerance for problems \mathbf{PRB} is $\varepsilon_{PRB} = 0.0001\%$, while the maximum time $TIME_{PRB}^{max}$ is instance dependent: 130, 135, 145, 45 and 70s for problems HYD2, HYD4, HYD7, SC#TP1 and SC#TP3, respectively. A maximum of $n_{par} = 80$ instances were solved in parallel and the minimum average domain reduction to consider OBBT effective was $ADR^{min} = 5\%$.

For the hydro problems, the algorithm terminates when the optimality gap $\varepsilon \leq 0.0001\%$ or upon reaching a wall time $TIME^{max} = 18,000$ s. For the refinery problems, the values

are 0.01%, and 3600/10,800 s when dealing with one/three periods. The partitioned variables in the hydro problems are the discharge flowrates. In the refinery problems, the partitioned variables are the stream flowrates, the inventory levels in the storage tanks, and the quality variables associated with the specific gravity.

7 Numerical results

All mathematical models and the global optimization algorithm were implemented in GAMS 24.7.3, taking advantage of its parallel computing grid facility. The MILP problems were solved by CPLEX 12.6.3, running in parallel deterministic mode and using up to 8 threads. CONOPT 3.17A solved the NLP problems. The MINLP benchmark problems were also solved by commercial global optimization solvers BARON 16.5 [37] and ANTIGONE 1.1 [39] using the same termination criteria. The former is centered around spatial branch-and-bound, while the latter focuses more on solving piecewise linear relaxations, applying bound tightening techniques, and generating different types of cutting planes. The hardware consisted of a server with an AMD Opteron™ Processor 6386 SE (2.79 GHz), 32 available cores, 64 GB RAM, and running Windows Server 2008 R2 Enterprise.

7.1 Comparison to our previous algorithms

The global optimization algorithms in our previous work have used piecewise relaxations in a different manner, see details in Table 4. They are responsible for the literature results in Table 5.

Results in [31] for the hydro problems came from a spatial branch-and-bound algorithm using the NMDT relaxation with $N_{PR} = 10$ partitions. OBBT was called in every node of the tree, prior to solving the relaxation problem (as in the current work), and involved solving a sequence of LPs ($N_{PRB} = 0$).

The algorithm solving the refinery problems in [13] used dynamic partitioning in the relaxation step as a replacement to spatial B&B, similarly to the one proposed in this work. The difference is that the number of partitions only increased, until reaching the computational time limit. Now, we enforce timing constraints per iteration to use the available time more efficiently, backtracking on the number of partitions whenever the relaxation problem cannot be solved to optimality. The two algorithms also share the parallel solution strategy for the bound contracting problems. However, the current algorithm calls OBBT more often, once per iteration and while domain reduction remains effective, instead of following the finding of a better solution. More importantly, our new algorithm adjusts to problem complexity by dynamically switching between McCormick and piecewise McCormick relaxations. In the former case, binary variables y are relaxed, leading to LPs instead of the MILPs ($N_{PRB} = 1$) in [13].

7.2 Performance overview

Table 5 shows the optimality gap and computational time required by the different algorithms, and results from the literature. The highlight is that the new algorithm always returns the lowest optimality gap. It can solve four problems to the given tolerance, compared to three problems by ANTIGONE and one by BARON. Our previous attempts with algorithms featuring piecewise relaxation methods and optimality-based bound tightening solved none of these benchmark problems to optimality. An ability to find the global optimal solution is

Table 4 Features of current and previous algorithms

Reference/feature	Spatial B&B	Partitioning relaxation step	OBBT calls	OBBT strategy	OBBT type of problems	Partitioning OBBT step
Hydro problems [31]	Yes	Static ($N = 10$)	In every node of the tree	Sequential	LPs	N.A.
Refinery problems [13]	No	Dynamic (up)	Upon finding better solution	Parallel	MILPs (McCormick)	N.A.
Current work	No	Dynamic (up/down)	While domain reduction is good	Parallel	LPs or MILPs (piecewise McCormick)	Dynamic (up, $N \leq 7$)

Table 5 Comparison between proposed algorithm, commercial global optimization solvers and literature results

Problem	Optimum	Optimality gap (%)			Wall time (s)			Algorithm	
		Literature ^a	BARON	ANTIGONE	Algorithm	Literature ^a	BARON		ANTIGONE
HYD2	209,721.0	0.023	GO	0.536	GO	WTL	17,417	WTL	5430
HYD4	371,811.8	0.585 ^b	1.359 ^b	1.323 ^b	0.465	WTL	WTL	WTL	WTL
HYD7	744,963.7	1.369 ^b	2.285 ^b	1.816 ^b	1.260 ^b	WTL	WTL	WTL	WTL
SC1TP1	55,568.1 ^c	0.12	0.42	GO	GO	WTL	WTL	614	2973
SC2TP1	49,799.3 ^c	0.06	0.21	GO	GO	WTL	WTL	169	2062
SC3TP1	53,798.8 ^c	0.18	0.43	GO	GO	WTL	WTL	376	2900
SC1TP3	55,561.8 ^c	0.79	1.96	0.93	0.60 ^b	WTL	WTL	WTL	WTL
SC2TP3	49,878.5 ^c	0.27	1.36 ^b	0.30	0.26 ^b	WTL	WTL	WTL	WTL
SC3TP3	53,785.8 ^c	0.97	2.35	1.11 ^b	0.80 ^b	WTL	WTL	WTL	WTL
SC4TP3	55,323.3 ^c	0.66	2.39	0.79	0.54	WTL	WTL	WTL	WTL

WTL wall time limit ($TIME^{max}$), GO global optimal solution (optimality gap $\leq \epsilon$)

^aResults for algorithms described in Sect. 7.1

^bBest-found solution is suboptimal

^cSlightly different values compared to [13] (costs within 0.04% except for SC2TP1: 0.17%) due to scaling up of sulfur content values to avoid numerical issues after four iterations with OBBT

Table 6 Detailed information about the performance of the new algorithm

Problem	HYD2	HYD4	HYD7	SC1TP1	SC2TP1	SC3TP1	SC1TP3	SC2TP3	SC3TP3	SC4TP3
Iterations	14	8	6	5	4	5	7	7	6	7
Time solving PR problems (s)	1140	12,907	12,386	571	424	553	940	864	900	938
Time solving PF problems (s)	155	86	69	63	46	58	143	108	107	130
Time in OB BT stage (s)	3902	3706	4418	2289	1554	2243	9665	9776	9750	9687
Time in depth search (s)	213	1301	1126	-	-	-	-	-	-	-
Instances of PF solved	203	111	134	160	107	116	256	166	218	273
Instances of PRB solved	2698	3610	3748	2630	1988	2630	13,104	13,094	10,936	13,104
ADR 1st iteration OB BT (%)	33.6	33.5	27.0	84.0	86.0	83.5	66.2	73.0	67.3	70.3
ADR final versus initial bounds (%)	99.5	67.0	32.9	97.8	95.2	97.4	80.0	85.9	78.1	81.8
Final N_{PR}	1000	4	4	4	8	4	4	4	4	4
Final N_{PRB}	7	3	2	2	0	2	2	2	2	2

ADR average domain reduction

also an important performance metric. The commercial global optimization solvers are doing better in this respect, returning suboptimal solutions in three problems compared to our new algorithms' four. It is an indication that there is still room for improving the upper bounding procedure.

BARON solves HYD2 three times slower, returning considerable larger gaps for the other problems. The poorer performance in the refinery problems might be due to the large number of variables involved in bilinear terms (see Table 3), and thus the potentially large number of nodes to explore in spatial branch-and-bound. ANTIGONE is an overall better performer than BARON and is considerable faster in the single period refinery problems. One possible explanation for the latter behavior is that cutting planes, or other techniques, are more efficient at reducing the domain of model variables than OBBT, when the problem size is small.

7.3 More detailed performance information

To understand how the algorithm is solving the benchmark problems, we show in Table 6 information related to: the total number of iterations; total time spent solving problems **PR** (step 5) and **PF** (step 6); executing OBBT (step 4) and depth search procedures (step 3); number of **PF** and **PRB** instances solved; average domain reduction in first and last OBBT call, with respect to the initial bounds; and number of partitions used in **PR** and **PRB** (final setting).

Piecewise relaxations are explored further in HYD2, with the algorithm reaching the maximum defined number of partitions for **PR** ($N_{PR} = 1000$) and **PRB** ($N_{PRB} = 7$). This is not surprising, considering that HYD2 has the fewest bilinear terms and variables appearing in bilinear terms (recall Table 3). As a consequence, we obtain the largest domain reduction (99.5%). Notice that HYD2 is the only problem taking advantage of the relaxation from multiparametric disaggregation.

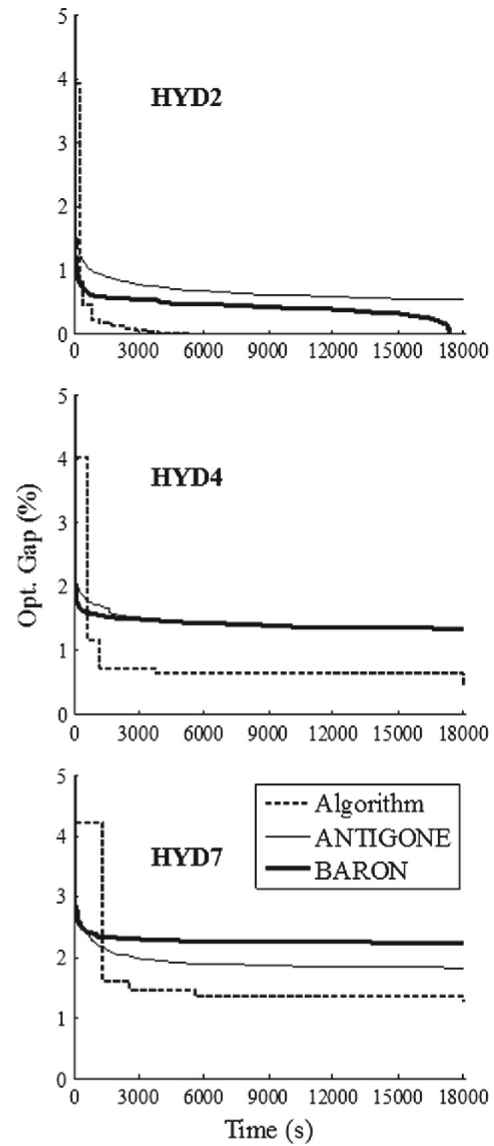
The final OBBT domain reduction is strongly dependent on problem size, decreasing to 67 and 32.9% when the number of reservoirs in the hydro problems increases from 2 to 4 and 7, and from above 95% to below 86% when switching from the single to the three-period refinery problems.

The time spent performing optimality-based bound tightening typically far exceeds the time spent solving relaxation problems. The two exceptions are HYD4 and HYD7, for which domain reduction became ineffective for $N_{PRB} = 3$ and 2 (while reaching the $TIME_{PRB}^{max}$ limit), and all remaining time was allocated to the final **PR** problems with $N_{PR} = 4$ partitions. Refinery problems SC#TP3 exhibited a similar behavior, with the larger number of **PRB** instances solved explaining the longer OBBT time. Options to improve the algorithm performance for such problems involve extending the time limit and reducing the number of **PRB** instances to be solved in parallel.

7.4 Closing the gap

The extensive use of time-consuming yet very efficient piecewise relaxation techniques by our algorithm, is clearly visible when plotting the optimality gap as a function of wall time, see Figs. 7 and 8. Recall from Fig. 6 that the optimality gap is only updated after a sequence of procedures: OBBT (tightens the variable bounds); solving problem **PR** (computes the lower bound, which may only improve with respect to the *LB* incumbent in the last moments of solving the MILP to optimality); solving NLP problems **PF** (compute the upper bound). The consequence is a stepwise profile with major drops in optimality gap compared to a smoother profile from the commercial solvers. Notice that there is still some progress towards the end

Fig. 7 Optimality gap versus wall time for hydro problems



of the search (SC#TP3 problems in Fig. 8), when the solvers have already plateaued. One disadvantage is that it may take a few hundred seconds to go below the gaps of ANTIGONE and BARON.

7.5 Removing the effect of OBBT

The four problems that were solved by the proposed algorithm to global optimality have in common the reduction of the domain of the variables involved in bilinear terms to less than 95% of the initial ranges, on average. We now test the performance of the commercial solvers after setting the variables domain to the final range obtained by our algorithm. The influence of the upper bound is also removed by initializing with the optimum.

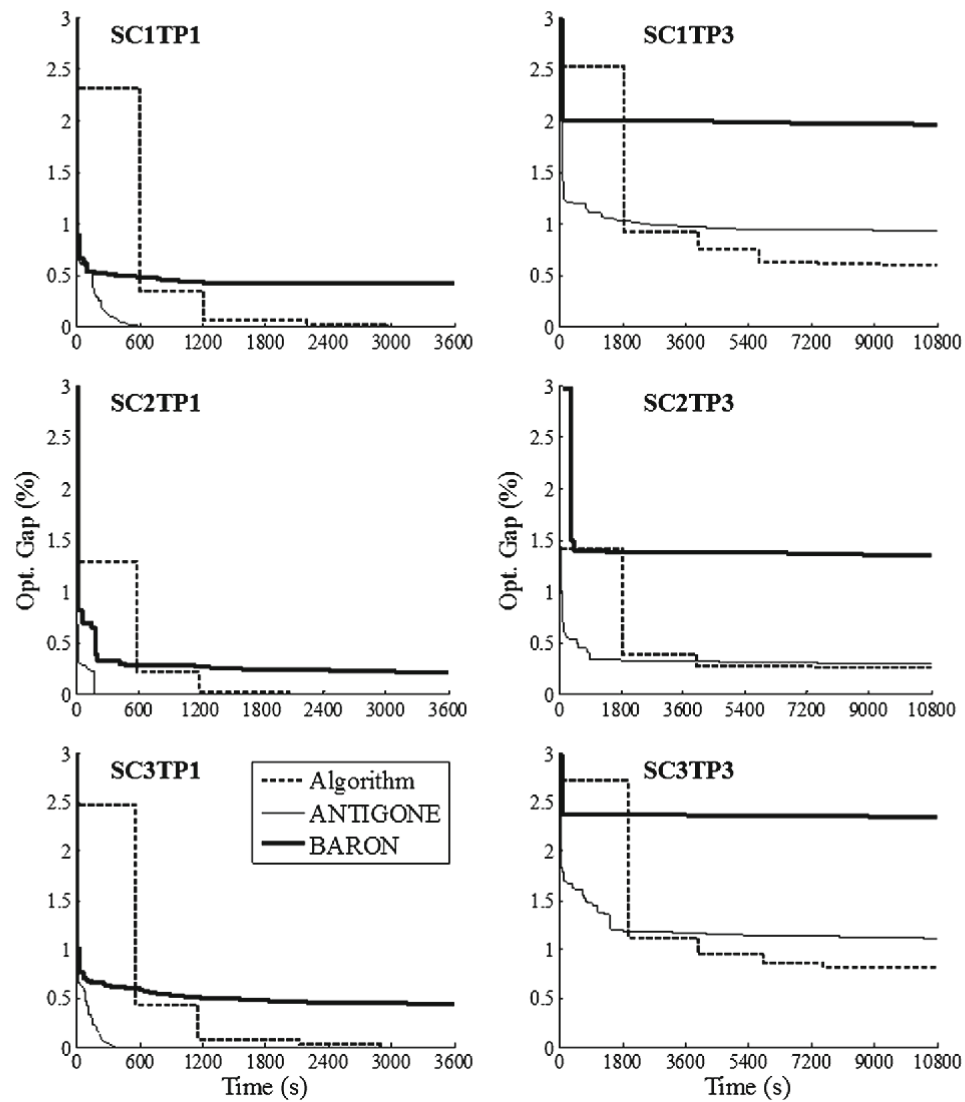


Fig. 8 Optimality gap versus wall time for refinery problems

The results in Table 7 show that the warm start helps ANTIGONE and BARON to solve such four problems in less than a minute. Improvements for HYD4, HYD7 and the refinery problems (with ANTIGONE) are minor. For the latter, BARON reduces the gap to less than half the values in Table 5. Neither solver can reach optimality gaps as low as the proposed algorithm, highlighting the importance of piecewise relaxations.

7.6 Sequential versus parallel OBTT

It remains to explain our choice for a parallel rather than a sequential implementation of optimality-based bound tightening. Figure 9 shows the optimality gap versus time profiles for refinery problems SC1TP1 and SC1TP3 and a fixed number of partitions in problems

Table 7 Performance of commercial solvers after warm start

Problem	Optimality gap (%)		Wall time (s)	
	ANTIGONE	BARON	ANTIGONE	BARON
HYD2	GO	GO	49	3
HYD4	1.241	1.157	WTL	WTL
HYD7	1.805	2.092	WTL	WTL
SC1TP1	GO	GO	3	3
SC2TP1	GO	GO	3	4
SC3TP1	GO	GO	3	8
SC1TP3	0.84	0.85	WTL	WTL
SC2TP3	0.28	0.28	WTL	WTL
SC3TP3	1.03	1.10	WTL	WTL
SC4TP3	0.64	0.72	WTL	WTL

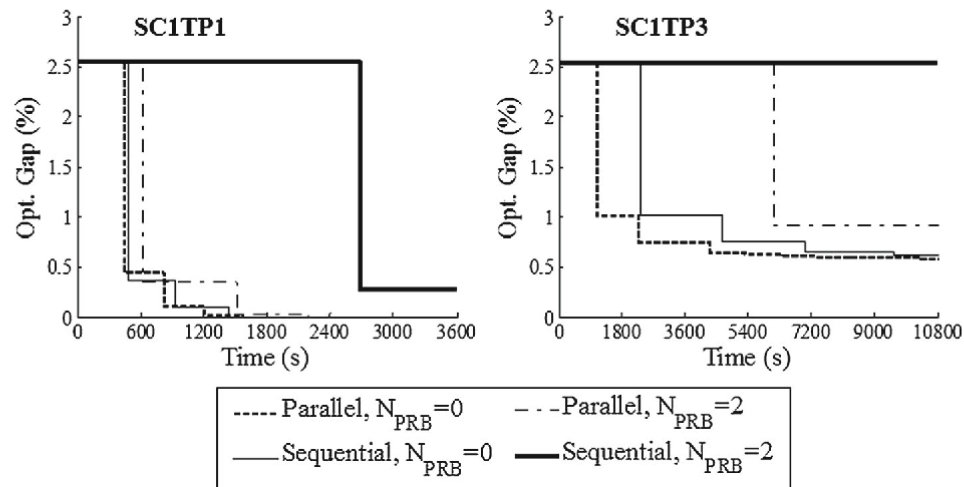


Fig. 9 Optimality gap profiles for sequential versus parallel OBBT

PR ($N_{PR} = 2$) and **PRB**, leading to the solution of LP ($N_{PRB} = 0$) or MILP problems ($N_{PRB} = 2$).

Results for the easiest SC1TP1 problem show that there are no major differences between the sequential and parallel implementations when solving LP problems. The optimality gaps are better when optimizing the bounds for one variable after the other (as expected) and not much time is lost compared to the parallel approach, for which the overhead of exchanging information between the threads is high. After the second iteration, the gaps become very similar and the lower computational time starts to be noticeable. Switching to MILP problems improves the relaxation quality and makes the parallel implementation far more competitive, with three iterations of OBBT taking less time and returning significantly smaller gaps than one iteration with the sequential approach.

Sequential OBBT with a piecewise relaxation ($N_{PRB} = 2$) is no longer an option for the larger SC1TP3, i.e. three hours are not enough to complete one iteration. We can still tackle one iteration with the parallel approach, but it is far more efficient to rely on the standard McCormick relaxation. Overall, the benefits from a parallel implementation of OBBT become

increasingly more important with the increase in the number of variables in bilinear terms and the number of partitions in **PRB**.

8 Conclusions

This paper has presented a new global optimization algorithm for mixed-integer quadratically constrained problems that does not employ spatial branch-and-bound. The novel aspect is the use of dynamic partitioning in piecewise relaxations, not only to compute lower bounds for the problem being minimized, but also to reduce the domain of the variables involved in bilinear terms. Relaxations range from the simplest bilinear envelopes at the start, to univariate piecewise McCormick, up to normalized multiparametric disaggregation, which is computationally more efficient for 10 partitions and beyond. The first type provides a quick lower bound, with the algorithm then switching to piecewise relaxations to refine such estimate. The number of partitions keeps increasing while the relaxation problem remains solvable to the given tolerance within the specified time. In case of severe increase in complexity, the algorithm backtracks to the previous setting, focusing more on optimality-based bound tightening (OBBT). Heuristic rules are used to decide when to increase/decrease the number of partitions in OBBT.

The algorithm has been designed to take advantage of parallel computing when doing OBBT and computing upper bounds. Rather than reducing one at a time the domain of the many variables that appear in bilinear terms, which leads to the tightest bounds, multiple variables are handled simultaneously to reduce the computational wall time. A solution pool is activated when solving the MILP relaxation problems, to generate alternative initialization points for solving restricted NLPs of the original non-convex problem that, if feasible, provide upper bounds. These are also solved in parallel.

The algorithm has been tested on ten industrially relevant benchmark problems, three hydroelectric scheduling problems with more discrete decisions and petroleum refinery planning problems with a larger number of bilinear terms. The computational results have shown that more problems can be solved to ε -global optimality. For the other six problems, the final optimality gaps were better than the values reported in the literature and lower than the ones from state-of-the-art commercial global optimization solvers ANTIGONE and BARON. The latter remained above our algorithm even when starting from a reduced variable range (from our last OBBT iteration). It shows that as problem size increases, piecewise relaxations with just 2 and 4 partitions can already provide tighter lower bounds than spatial branch-and-bound. Commercial solvers should thus use them to a greater extent.

Acknowledgements Support by Ontario Research Foundation, McMaster Advanced Control Consortium, and Fundação para a Ciência e Tecnologia (Projects IF/00781/2013 and UID/MAT/04561/2013), is gratefully appreciated.

References

1. Meyer, C.A., Floudas, C.A.: Global optimization of a combinatorially complex generalized pooling problem. *AIChE J.* **52**, 1027–1037 (2006)
2. Misener, R., Thompson, J.P., Floudas, C.A.: APOGEE: global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Comput. Chem. Eng.* **35**, 876–892 (2011)

3. Castro, P.M.: New MINLP formulation for the multiperiod pooling problem. *AIChE J.* **61**, 3728–3738 (2015)
4. Lotero, I., Trespalacios, F., Grossmann, I.E., Papageorgiou, D.J., Cheon, M.-S.: An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem. *Comput. Chem. Eng.* **87**, 13–35 (2016)
5. Quesada, I., Grossmann, I.E.: Global optimization of bilinear process networks with multicomponent flows. *Comput. Chem. Eng.* **19**, 1219–1242 (1995)
6. Lee, S., Grossmann, I.E.: Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints: applications to process networks. *Comput. Chem. Eng.* **27**, 1557–1575 (2003)
7. Faria, D.C., Bagajewicz, M.J.: Novel bound contraction procedure for global optimization of bilinear MINLP problems with applications to water management problems. *Comput. Chem. Eng.* **35**, 446–455 (2011)
8. Rubio-Castro, E., Ponce-Ortega, J.M., Serna-González, M., El-Halwagi, M.M., Pham, V.: Global optimization in property-based interplant water integration. *AIChE J.* **59**, 813–833 (2013)
9. Alnouri, S., Linke, P., El-Halwagi, M.M.: Spatially constrained interplant water network synthesis with watertreatment options. In: Eden, M.R., Stirolo, J.D.S., Towler, G.P. (eds.) *Proceedings of the 8th International Conference on Foundations of Computer-Aided Process Design*, pp. 237–242. Elsevier, Amsterdam (2014)
10. Teles, J.P., Castro, P.M., Matos, H.A.: Global optimization of water networks design using multiparametric disaggregation. *Comput. Chem. Eng.* **40**, 132–147 (2012)
11. Koleva, M.N., Styan, C.A., Papageorgiou, L.G.: Optimisation approaches for the synthesis of water treatment plants. *Comput. Chem. Eng.* (2017)
12. Andrade, T., Ribas, G., Oliveira, F.: A strategy based on convex relaxation for solving the oil refinery operations planning problem. *Ind. Eng. Chem. Res.* **55**, 144–155 (2016)
13. Castillo Castillo, P., Castro, P.M., Mahalec, V.: Global optimization algorithm for large-scale refinery planning models with bilinear terms. *Ind. Eng. Chem. Res.* **56**, 530–548 (2017)
14. Castro, P.M., Grossmann, I.E.: Global optimal scheduling of crude oil blending operations with RTN continuous-time and multiparametric disaggregation. *Ind. Eng. Chem. Res.* **53**, 15127–15145 (2014)
15. Cerdá, J., Pautasso, P.C., Cafaro, D.C.: Efficient approach for scheduling crude oil operations in marine-access refineries. *Ind. Eng. Chem. Res.* **54**, 8219–8238 (2015)
16. Zhao, Y., Wu, N., Li, Z., Qu, T.: A novel solution approach to a priority-slot-based continuous-time mixed integer nonlinear programming formulation for a crude-oil scheduling problem. *Ind. Eng. Chem. Res.* **55**, 10955–10967 (2016)
17. Catalão, J.P.S., Pousinho, H.M.I., Mendes, V.M.F.: Hydro energy systems management in Portugal: profit-based evaluation of a mixed-integer nonlinear approach. *Energy* **36**, 500–507 (2011)
18. Horst, R., Tuy, H.: *Global Optimization: Deterministic Approaches*. Springer, Berlin (2013)
19. Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. *J. Glob. Optim.* **8**, 107–138 (1996)
20. Smith, E.M.B., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. *Comput. Chem. Eng.* **21**, S791–S796 (1997)
21. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Math. Program.* **10**, 147–175 (1976)
22. Karuppiah, R., Grossmann, I.E.: Global optimization for the synthesis of integrated water systems in chemical processes. *Comput. Chem. Eng.* **30**, 650–673 (2006)
23. Alfaki, M., Haugland, D.: A multi-commodity flow formulation for the generalized pooling problem. *J. Glob. Optim.* **56**, 917–937 (2013)
24. Bergamini, M.L., Aguirre, P., Grossmann, I.: Logic-based outer approximation for globally optimal synthesis of process networks. *Comput. Chem. Eng.* **29**, 1914–1933 (2005)
25. Wicaksono, D.S., Karimi, I.A.: Piecewise MILP under- and overestimators for global optimization of bilinear programs. *AIChE J.* **54**, 991–1008 (2008)
26. Li, X., Chen, Y., Barton, P.I.: Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems. *Ind. Eng. Chem. Res.* **51**, 7287–7299 (2012)
27. Castro, P.M.: Tightening piecewise McCormick relaxations for bilinear problems. *Comput. Chem. Eng.* **72**, 300–311 (2015)
28. Kolodziej, S., Castro, P.M., Grossmann, I.E.: Global optimization of bilinear programs with a multiparametric disaggregation technique. *J. Glob. Optim.* **57**, 1039–1063 (2013)
29. Castro, P.M.: Normalized multiparametric disaggregation: an efficient relaxation for mixed-integer bilinear problems. *J. Glob. Optim.* **64**, 765–784 (2016)

30. Faria, D.C., Bagajewicz, M.J.: A new approach for global optimization of a class of MINLP problems with applications to water management and pooling problems. *AIChE J.* **58**, 2320–2335 (2012)
31. Castro, P.M.: Spatial branch-and-bound algorithm for MIQCPs featuring multiparametric disaggregation. *Optim. Methods Softw.* **32**, 719–737 (2017)
32. Castro, P.M., Teles, J.P.: Comparison of global optimization algorithms for the design of water-using networks. *Comput. Chem. Eng.* **52**, 249–261 (2013)
33. Castro, P.M., Grossmann, I.E.: Optimality-based bound contraction with multiparametric disaggregation for the global optimization of mixed-integer bilinear problems. *J. Glob. Optim.* **59**, 277–306 (2014)
34. Nagarajan, H., Lu, M., Yamangil, E., Bent, R.: Tightening McCormick relaxations for nonlinear programs via dynamic multivariate partitioning. In: Rueher, M. (ed.) *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5–9, 2016, Proceedings*, pp. 369–387. Springer, Cham (2016)
35. Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Math. Program.* **124**, 383–411 (2010)
36. Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Math. Program.* **130**, 359–413 (2011)
37. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. *Math. Program.* **103**, 225–249 (2005)
38. Misener, R., Floudas, C.A.: GloMIQO: global mixed-integer quadratic optimizer. *J. Glob. Optim.* **57**, 3–50 (2013)
39. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *J. Glob. Optim.* **59**, 503–526 (2014)
40. Gleixner, A.M., Berthold, T., Müller, B., Weltge, S.: Three enhancements for optimization-based bound tightening. *J. Glob. Optim.* **67**, 731–757 (2017)
41. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J. Algebr. Discrete Methods* **6**, 466–486 (1985)
42. Atamturk, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Conflict graphs in solving integer programming problems. *Eur. J. Oper. Res.* **121**, 40–55 (2000)

Chapter 10: Concluding Remarks

This thesis has focused on the development of efficient algorithms to solve production planning and scheduling problems. Two approaches were considered: i) a heuristic algorithm based on the inventory pinch concept to compute near-optimal solutions in short execution times, and ii) a rigorous deterministic global optimization algorithm based on increasing number of partitions of piecewise linear relaxations. The main case studies included gasoline blend planning and scheduling, and refinery planning.

The inventory pinch algorithm decomposes the problem into three levels: 1) optimization of operating conditions and blend recipes, 2) computation of an approximate schedule, and 3) detailed scheduling. At the first level, a discrete-time NLP model is formulated, where periods are delineated by the inventory pinch points for various product pools (e.g. gasoline and diesel). This reduces drastically the number of periods and enables use of nonlinear, more accurate refinery models. The second level is solved via discrete-time MILP model where periods are delineated by scheduler based on the demand and supply data, and the minimum time requirements to complete major tasks (i.e. blend runs, product tank service). The third level uses a discrete-time MILP scheduling model (MPIP algorithm) or a continuous-time MILP scheduling model (MPIP-C) to determine the exact times to carry out the necessary tasks. The second and third levels are linear models since the nonlinear constraints are handled at the first level, and the optimal conditions found at such level are fixed in the other levels. The algorithm minimizes the total cost which is defined as the cost of raw materials, switching cost, and demurrage cost. The algorithm eliminates infeasibilities by iteratively re-optimizing operating conditions and blend recipes at the first level.

The deterministic global optimization algorithm relies on discretizing the bilinear or quadratic terms dynamically using either piecewise McCormick (PMCR) or normalized multiparametric disaggregation (NMDT). The resulting MILP model is solved using CPLEX and several feasible solutions are stored in CPLEX's solution pool and employed as starting points for a local nonlinear solver (e.g. CONOPT). These nonlinear models are solved in parallel. Then, the estimate of the global solution and the best feasible solution are updated. If the relative difference between these two (i.e. the optimality gap) is smaller than the tolerance, then the algorithm stops; otherwise, it continues by reducing the range of the variables or increasing the number of partitions for the next iteration. The domain of the variables involved in nonlinear terms is reduced using an optimality-based bound tightening (OBBT) method. This OBBT method consists in solving two optimization problems for each variable: a maximization and a minimization of the range

of the variable subject to the MILP relaxation constraints. Parallelization of this step is required to avoid long execution times.

10.1. Key Findings and Contributions

The research objectives presented in Chapter 1 have been achieved, and the key contributions of this work include:

- 10.1.1. The development of a heuristic technique for blend planning and scheduling problems: the multiperiod inventory pinch algorithm MPIP. This method computes blend plans and schedules with reduced number of different blend recipes by reducing the number of time periods using the inventory pinch points. The inventory pinch points are defined by the cumulative total demand along the planning/scheduling horizon. MPIP employs discrete-time uniform-grid MILP scheduling model. Results in Chapter 2 and 3 show that MPIP computes the same or better solutions than three commercial solvers trying to solve the original full-space model. In Chapter 4, MPIP is used to solve a refinery planning problem.
- 10.1.2. In Chapter 2, results indicate that the solutions computed by the MPIP planning algorithm are optimal when the objective function of the second level contains only variables that are aggregated at the first level; and they are near-optimal when the objective function of the second level contains a penalty term associated with variables that are not aggregated at the first level, and this penalty term is significantly smaller than the cost of raw materials.
- 10.1.3. The formulation of a continuous-time unit-specific slot-based MILP scheduling model with reduced number of binary variables for gasoline blending operations. In Chapter 5, it is shown that the addition of a lower bound on the blend cost reduces the execution times required to solve blend scheduling problems to optimality.
- 10.1.4. The development of the multiperiod inventory pinch algorithm MPIP-C for scheduling problems. MPIP-C has all the features of MPIP but it employs a continuous-time unit-specific slot-based MILP scheduling model. As shown in Chapter 6 and 8, MPIP-C computes solutions in shorter execution times than three commercial solvers, and around the same times as another published heuristic strategy.
- 10.1.5. The development of a deterministic global optimization algorithm for MINLP problems where nonlinearities are strictly bilinear and/or quadratic terms. The algorithm is based on dynamic partitioning of piecewise linear relaxations

(PMCR and NMDT) and optimality-based bound tightening. Chapter 7, 8, and 9 show that the algorithm performs on par with two commercial global solvers, and even better in some examples.

10.2. Future Work Outlook

The MPIP and MPIP-C algorithms have shown promising results for short-term planning and scheduling problems where 1) the cost associated with the raw materials is bigger than the cost associated with switching tasks, and 2) the problem can be decomposed into 2 or 3 decision levels. However, the performance of these heuristic algorithms depends on the ability of the modeler to define the constraints that will be included at each level. Therefore, it is necessary to develop a systematic approach to generate the mathematical models for each level based on the original problem formulation and with minimal additional input from the planner/scheduler. Such development will simplify the application and implementation of these two inventory pinch-based algorithms to a wider variety of planning and scheduling problems, as well as its integration with global optimization algorithms (to find feasible solutions).

A possible next step for the MPIP method is to employ it for solving and linking long- and medium-term planning problems. The questions to be answered include:

1. What granularity of the product demand data to use? Different data granularities (e.g., daily and hourly data) could yield different inventory pinch points.
2. What are the best linking decisions between the long- and medium-term plans? These will depend on the selected case study. For example, for an oil refinery, these can be the total amount of crude oil to purchase, the crude distillation unit throughput, or the final inventory levels.

The deterministic global optimization algorithm from Chapter 9 can be further enhanced. One of the major issues of the current implementation is when the optimality-based bound tightening (OBBT) method is not run and the number of partitions in the relaxed model is increased. In this situation, the MILP solver might explore many of the nodes that were fathomed in the previous iteration. To avoid this unnecessary calculations, it is necessary to retrieve the branch-and-bound tree information from the MILP solver.

Another issue of the deterministic global optimization algorithm is that there is no specific rule to select the variable of a bilinear term to be partitioned. The current rule is to pick the variables that will lead to the smallest MILP relaxation. The proposed method is to make this a dynamic selection during the algorithm run. Let's consider the bilinear

term x_1x_2 , where x_2 is the partitioned variable at the beginning of the algorithm. Once the domain of x_2 cannot be reduced by OBBT, and there is no significant improvement in the best possible solution, x_1 becomes the partitioned variable.

In the dynamic partitioning scheme employed by the deterministic global optimization algorithm, the number of partitions of all partitioned variables increase by the same factor. A topic that can be investigated is if this factor can be different for each partitioned variable, and how to determine it. This can lead to smaller MILP relaxations.

One possible approach to decrease the time required for solving the MILP relaxations is to employ a mathematical decomposition strategy. Either Benders or Lagrangean decomposition methods could prove to be useful given the block structure of the constraints associated with Piecewise McCormick and Normalized Multiparametric Disaggregation.

Appendix A: Supporting Information for Chapters 2 and 3

Table A.1. Components data (properties, cost, supply rates and inventory limits)

Components	ALK	BUT	HCL	HCN	LCN	LNP	RFT
ARO (% vol aromatics)	0	0	0	25	18	2.974	74.9
BEN (% vol benzene)	0	0	0	0.5	1	0.595	7.5
MON	93.7	90	79.8	75.8	81.6	66	90.8
OLF (% vol olefin)	0	0	0	14	27	0	0
RON	95	93.8	82.3	86.7	93.2	67.8	103
RVP (psi)	5.15	138	22.335	2.378	13.876	19.904	3.622
SPG	0.703	0.584	0.695	0.791	0.744	0.677	0.818
SUL (% vol sulfur)	0	0	0	0.485	0.078	0.013	0
Cost (\$/bbl)	29.2	11.5	20	22	25	19.7	24.5
Minimum Inventory ($\times 10^3$ bbl)	5	5	5	5	5	5	5
Maximum Inventory ($\times 10^3$ bbl)	150	75	50	50	150	100	150
Initial Inventory ($\times 10^3$ bbl) Cases 1 – 14	30	20	20	10	30	20	50
Supply Rate ($\times 10^3$ bbl/day) Cases 1 – 7	18	5	3	5	25	20	44

Table A.2. Supply rate of components along planning horizon, cases 8 – 14

Component	ALK	BUT	HCL	HCN	LCN	LNP	RFT
<i>L2-period</i>	$\times 10^3$ bbl/day						
1	25	7	0	3	27	20	45
2	25	7	0	3	27	20	45
3	25	7	0	3	27	20	45
4	20	5	3	5	25	18	40
5	15	3	7	9	20	22	35
6	15	3	7	9	20	22	35
7	15	3	7	9	20	22	35
8	20	5	3	5	25	18	40
9	20	5	3	5	25	18	40
10	25	7	0	3	27	22	45
11	25	7	0	3	27	22	45
12	25	7	0	3	27	22	45
13	20	5	3	5	25	18	40
14	20	5	3	5	25	18	40

Table A.3. Minimum and maximum quality specifications of the products

Specification	Minimum			Maximum		
Product	U87	U91	U93	U87	U91	U93
ARO (% vol aromatics)	0	0	0	60	60	60
BEN (% vol benzene)	0	0	0	5.9	5.9	5.9
MON	81.5	85.7	87.5	-	-	-
OLF (% vol olefin)	0	0	0	24.2	24.2	24.2
RON	91.4	94.5	97.5	-	-	-
RVP (psi)	0	0	0	15.6	15.6	15.6
SPG	0.73	0.73	0.73	0.81	0.81	0.81
SUL (% vol sulfur)	0	0	0	0.1	0.1	0.1

Table A.4. Product storage tank data

Product tank	Storable products	Product transition penalty ($\times 10^3$ \$)	Minimum hold up ($\times 10^3$ bbl)	Maximum hold up ($\times 10^3$ bbl)	Maximum delivery rate ($\times 10^3$ bbl/h)	Initial inventory ($\times 10^3$ bbl)	Initial product
Tk-101	U87	-	10	70	10	40	U87
Tk-102	U91	-	10	70	10	70	U91
Tk-103	U93	-	10	70	10	30	U93
Tk-104	U87, U91, U93	14.5	0	40	10	30	U87
Tk-105	U87, U91, U93	14.5	0	40	10	40	U91
Tk-106	U87, U91, U93	14.5	0	40	10	30	U91

Table A.5. Demand profiles ($\times 10^3$ bbl) and cost coefficient profile for the product inventory slack variables (2nd level MILP model)

Demand profile	Product	L2-period													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	U87	60	50	50	80	50	60	60	50	75	50	50	50	80	100
	U91	50	80	70	30	50	0	40	30	30	50	40	40	30	50
	U93	30	30	0	0	40	40	0	35	30	0	0	40	30	40
2	U87	80	80	60	80	80	100	90	0	0	50	50	30	60	100
	U91	50	50	50	30	30	50	50	30	30	50	0	50	60	50
	U93	30	30	35	30	35	0	30	35	30	0	30	40	30	0
3	U87	70	70	50	70	70	60	60	60	50	70	120	0	50	70
	U91	50	50	50	30	30	50	50	30	30	50	50	30	30	50
	U93	30	30	45	30	40	0	0	35	30	0	30	35	0	30
4	U87	70	50	50	120	100	30	30	50	75	110	50	50	50	90
	U91	50	80	70	30	50	0	0	30	50	50	0	40	30	0
	U93	30	30	45	0	40	40	0	35	30	30	30	0	30	30
5	U87	60	50	50	70	90	80	130	50	0	30	50	50	50	80
	U91	50	80	70	50	50	30	30	30	30	30	0	40	30	0
	U93	30	30	45	0	30	40	30	30	30	30	30	0	30	40
6	U87	100	70	80	100	40	30	40	110	0	50	70	100	0	50
	U91	50	80	70	50	30	30	30	50	30	30	30	35	30	30
	U93	30	30	45	30	0	30	30	30	30	0	0	30	30	30
Cost coefficients for product slack variables															
U87, U91, U93	1.8 $\times 10^6$	1.7 $\times 10^6$	1.6 $\times 10^6$	1.5 $\times 10^5$	1.4 $\times 10^5$	1.3 $\times 10^5$	1.2 $\times 10^5$	1.1 $\times 10^4$	1 $\times 10^3$	9 $\times 10^2$	8 $\times 10^2$	5 $\times 10^2$	1 $\times 10^2$	5 $\times 10^1$	

Appendix B: Supporting Information for Chapters 5, 6, and 8

Table B.1. Demand data

Example	Product		Demand (kbbbl)						Maximum Delivery Rate D_{Order}^{max} (kbbbl/h)					
	12	The rest	3	4	7-8	9	12	14	3	4	7-8	9	12	14
Order														
O1	P1	P1	10	10	10	10	10	10	5	5	5	5	5	5
O2	P2	P2	3	3	3	3	3	3	3	3	3	3	3	3
O3	P2	P2	3	3	3	3	3	3	3	3	3	3	3	3
O4	P1	P1	10	10	10	10	10	10	5	5	5	5	5	5
O5	P2	P2	3	3	3	3	3	3	3	3	3	3	3	3
O6	P1	P1	10	10	10	10	10	10	5	5	5	5	5	5
O7	P2	P2	3	3	3	3	3	3	3	3	3	3	3	3
O8	P1	P1	100	100	100	100	100	100	5	5	5	5	5	5
O9	P2	P2	3	3	3	3	3	3	3	3	3	3	3	3
O10	P4	P4	150	150	150	150	100	150	5	5	5	5	5	5
O11	P3	P3	20	20	60	60	60	60	5	5	5	5	5	5
O12	P2	P2	30	30	20	20	20	20	5	5	5	5	5	5
O13	P4	P4	-	60	60	60	60	60	-	5	5	5	5	5
O14	P3	P3	-	10	15	20	15	20	-	5	5	5	5	5
O15	P2	P2	-	20	20	20	20	20	-	4	4	4	4	4
O16	P2	P2	-	-	20	20	20	20	-	-	5	5	5	5
O17	P1	P1	-	-	10	10	10	10	-	-	5	5	5	5
O18	P1	P1	-	-	10	10	10	10	-	-	5	5	5	5
O19	P2	P2	-	-	60	60	60	60	-	-	5	5	5	5
O20	P2	P2	-	-	40	40	40	40	-	-	5	5	5	5
O21	P5	P1	-	-	-	30	30	30	-	-	-	5	5	5
O22	P5	P5	-	-	-	40	40	40	-	-	-	5	5	5
O23	P3	P3	-	-	-	20	20	20	-	-	-	5	5	5
O24	P5	P5	-	-	-	-	6	6	-	-	-	-	3	3
O25	P5	P5	-	-	-	-	20	20	-	-	-	-	5	5
O26	P3	P1	-	-	-	-	30	10	-	-	-	-	4	4
O27	P3	P4	-	-	-	-	20	20	-	-	-	-	4	5
O28	P4	P1	-	-	-	-	3	25	-	-	-	-	3	5
O29	P4	P5	-	-	-	-	15	10	-	-	-	-	3	5
O30	P1	P4	-	-	-	-	15	15	-	-	-	-	3	5
O31	P2	P1	-	-	-	-	15	15	-	-	-	-	5	5
O32	P5	P1	-	-	-	-	20	20	-	-	-	-	2	5
O33	P1	P4	-	-	-	-	20	20	-	-	-	-	5	5
O34	P3	P4	-	-	-	-	20	20	-	-	-	-	5	5
O35	P3	P5	-	-	-	-	30	30	-	-	-	-	5	5
O36	-	P2	-	-	-	-	-	3	-	-	-	-	-	3
O37	-	P1	-	-	-	-	-	10	-	-	-	-	-	5
O38	-	P1	-	-	-	-	-	40	-	-	-	-	-	5
O39	-	P4	-	-	-	-	-	10	-	-	-	-	-	5
O40	-	P5	-	-	-	-	-	10	-	-	-	-	-	5
O41	-	P1	-	-	-	-	-	15	-	-	-	-	-	5
O42	-	P2	-	-	-	-	-	20	-	-	-	-	-	3
O43	-	P3	-	-	-	-	-	15	-	-	-	-	-	5
O44	-	P5	-	-	-	-	-	20	-	-	-	-	-	4
O45	-	P4	-	-	-	-	-	10	-	-	-	-	-	5

Table B.2. Delivery windows

Example	Delivery Window [TO^{start} , TO^{end}] (h)					
	3	4	7-8	9	12	14
Order						
O1	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
O2	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
O3	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
O4	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]	[0,24]
O5	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
O6	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
O7	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
O8	[118,190]	[118,190]	[118,190]	[118,190]	[118,190]	[118,190]
O9	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
O10	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]	[150.5,185.5]
O11	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]	[144,168]
O12	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]	[24,48]
O13	-	[0,56]	[0,56]	[0,56]	[0,56]	[0,56]
O14	-	[48,72]	[48,72]	[48,72]	[48,72]	[48,72]
O15	-	[0,72]	[0,72]	[0,72]	[0,72]	[0,72]
O16	-	-	[48,72]	[48,72]	[48,72]	[48,72]
O17	-	-	[48,72]	[48,72]	[48,72]	[48,72]
O18	-	-	[48,72]	[48,72]	[48,72]	[48,72]
O19	-	-	[0,50]	[0,50]	[0,50]	[0,50]
O20	-	-	[144, 168]	[144,168]	[144,168]	[144,168]
O21	-	-	-	[96,120]	[96,120]	[96,120]
O22	-	-	-	[144,168]	[144,168]	[144,168]
O23	-	-	-	[144,168]	[144,168]	[144,168]
O24	-	-	-	-	[96,120]	[96,120]
O25	-	-	-	-	[144,168]	[144,168]
O26	-	-	-	-	[144,168]	[0,76]
O27	-	-	-	-	[72,96]	[120,144]
O28	-	-	-	-	[72,96]	[120,144]
O29	-	-	-	-	[96,120]	[120,144]
O30	-	-	-	-	[96,120]	[120,144]
O31	-	-	-	-	[96,120]	[120,144]
O32	-	-	-	-	[96,120]	[144,168]
O33	-	-	-	-	[0,76]	[144,168]
O34	-	-	-	-	[120,144]	[168,192]
O35	-	-	-	-	[120,144]	[168,192]
O36	-	-	-	-	-	[168,192]
O37	-	-	-	-	-	[168,192]
O38	-	-	-	-	-	[168,192]
O39	-	-	-	-	-	[168,192]
O40	-	-	-	-	-	[168,192]
O41	-	-	-	-	-	[168,192]
O42	-	-	-	-	-	[168,192]
O43	-	-	-	-	-	[144,168]
O44	-	-	-	-	-	[168,192]
O45	-	-	-	-	-	[96,120]

Table B.3. Product and component tank data

Product or Component Tank	Initial Product	Initial Stock V^{ini} (kbbbl)	Max. Capacity V^{max} (kbbbl)	Storable Products (Set JP)			Max. Delivery Rate D_{pr}^{max} (kbbbl/h)				
				3, 4, 7, 8	9, 14	12	3, 4	7-8	9	12	14
Example				3, 4, 7, 8	9, 14	12	3, 4	7-8	9	12	14
Tk1	P3	30.00	150	P2, P3	P2, P3, P5	P2, P3, P5	20	20	30	30	30
Tk2	P3	0.00	150	P2, P3	P2, P3, P5	P2, P3, P5	20	20	30	30	30
Tk3	P2	14.08	150	P2, P3	P2, P3, P5	P2, P3, P5	20	20	30	30	30
Tk4	P4	25.00	200	P2- P4	P2- P4	P2- P5	20	20	30	30	30
Tk5	P2	28.49	200	P2, P3	P2, P5	P2, P3, P5	20	20	30	30	30
Tk6	P2	57.59	150	P2, P3	P2, P5	P2, P3, P5	20	20	30	30	30
Tk7	P1	13.79	200	P1, P4	P1, P4	P1, P4	20	20	30	30	30
Tk8	P1	12.36	150	P1, P4	P1, P4	P1, P4	20	20	30	30	30
Tk9	P4	23.96	200	P1, P4	P1, P4	P1, P4	20	20	30	30	30
Tk10	P1	60.00	150	P1, P4	P1, P4	P1, P4	20	20	30	30	30
Tk11	P1	12.36	150	P1, P4	P1, P4	P1, P4	20	20	30	30	30
C1	C1	26.46	250	-	-	-	-	-	-	-	-
C2	C2	67.90	300	-	-	-	-	-	-	-	-
C3	C3	59.44	300	-	-	-	-	-	-	-	-
C4	C4	44.44	300	-	-	-	-	-	-	-	-
C5	C5	10.59	200	-	-	-	-	-	-	-	-
C6	C6	19.53	250	-	-	-	-	-	-	-	-
C7	C7	46.91	250	-	-	-	-	-	-	-	-
C8	C8	49.47	250	-	-	-	-	-	-	-	-
C9	C9	44.58	250	-	-	-	-	-	-	-	-

Table B.4. Product and component property specification (ON, RVPI, SI)

Comp./ Product	Quality Example	ON		RVPI		SI	
		4, 7, 8	9, 12, 14	4, 7, 8	9, 12, 14	4, 7, 8	9, 12, 14
C1	Q_{bc}	86.50	86.50	140.47	140.47	80.00	80.00
C2		103.66	103.66	68.92	68.92	40.00	40.00
C3		111.35	111.35	87.68	87.68	0.00	0.00
C4		113.93	113.93	51.47	51.47	5.00	5.00
C5		94.50	94.50	175.59	175.59	0.00	0.00
C6		118.16	118.16	19.91	19.91	0.08	0.08
C7		144.68	144.68	12.55	12.55	7.50	7.50
C8		150.66	150.66	110.59	110.59	2.00	2.00
C9		92.50	92.50	436.34	436.34	30.00	30.00
P1	$[Q_{pr}^{min}, Q_{pr}^{max}]$	[110.45, +∞]	[110.45, +∞]	[15, 170]	[15, 170]	[0, 45]	[0, 45]
P2		[111.95, +∞]	[111.95, +∞]	[15, 170]	[15, 170]	[0, 50]	[0, 50]
P3		[108.97, +∞]	[108.97, +∞]	[15, 170]	[15, 170]	[0, 44]	[0, 44]
P4		[103.24, +∞]	[103.24, +∞]	[15, 170]	[15, 170]	[0, 50]	[0, 50]
P5		-	[115.01, +∞]	-	[15, 170]	-	[0, 48]

Table B.5. Product and component property specification (BI, AI, OI)

Comp./ Product	Quality Example	BI		AI		OI		
		4, 7, 8	9, 12, 14	4, 7, 8	9, 12, 14	4	7, 8	9, 12, 14
C1	Q_{bc}	0.78	0.78	25.00	25.00	1.00	1.00	1.00
C2		0.98	0.98	31.70	31.70	23.80	23.80	23.80
C3		1.20	1.20	48.00	48.00	0.85	0.85	0.85
C4		0.00	0.00	0.00	0.00	0.00	0.00	0.00
C5		0.10	0.10	0.00	0.00	0.40	0.40	0.40
C6		0.01	0.01	0.00	0.00	0.72	0.72	0.72
C7		0.01	0.01	0.05	0.05	0.00	0.00	0.00
C8		0.25	0.25	19.20	19.20	0.15	0.15	0.15
C9		0.09	0.09	24.00	24.00	0.06	0.06	0.06
P1	$[Q_{pr}^{min}, Q_{pr}^{max}]$	[0, 0.86]	[0, 0.86]	[0, 35.00]	[0, 35.00]	[0, 20.00]	[0, 20.00]	[0, 20.00]
P2		[0, 0.92]	[0, 0.92]	[0, 36.00]	[0, 36.00]	[0, 18.00]	[0, 18.00]	[0, 18.00]
P3		[0, 0.94]	[0, 0.94]	[0, 42.00]	[0, 42.00]	[0, 20.00]	[0, 20.00]	[0, 20.00]
P4		[0, 0.90]	[0, 0.90]	[0, 40.00]	[0, 40.00]	[0, 18.00]	[0, 18.00]	[0, 18.00]
P5		-	[0, 0.93]	-	[0, 40.00]	-	-	[0, 20.00]

Table B.6. Product and component property specification (BI, AI, OI)

Comp. / Product	Quality Example	SGI		FI		OXI			
		4, 7, 8	9, 12, 14	4, 7, 8	9, 12, 14	4	7, 8	9	12, 14
C1	Q_{bc}	1.49	1.49	3.45	3.45	0.25	0.25	0.25	0.25
C2		1.33	1.33	6.25	6.25	0.75	0.75	0.75	0.75
C3		1.22	1.22	2.36	2.36	2.00	2.00	2.00	2.00
C4		1.58	1.58	3.56	3.56	1.25	1.25	1.25	1.25
C5		1.50	1.50	1.96	1.96	0.08	0.08	0.08	0.08
C6		1.44	1.44	3.65	3.65	0.00	0.00	0.00	0.00
C7		1.15	1.15	2.96	2.96	0.00	0.00	0.00	0.00
C8		1.35	1.35	5.46	5.46	18.20	18.20	18.20	18.20
C9		1.61	1.61	7.95	7.95	0.85	0.85	0.85	0.85
P1	$[Q_{pr}^{min}, Q_{pr}^{max}]$	[1.19, 1.67]	[1.19, 1.67]	[1.4, 7.60]	[1.4, 7.60]	[0, 1.85]	[0, 2.80]	[0, 2.80]	[0, 2.80]
		[1.20, 1.67]	[1.20, 1.67]	[1.4, 7.25]	[1.4, 7.25]	[0, 1.90]	[0, 2.75]	[4, 7.25]	[0, 2.75]
P2		[1.18, 1.67]	[1.18, 1.67]	[1.4, 7.20]	[1.4, 7.20]	[0, 2.10]	[0, 2.90]	[0, 2.90]	[0, 2.90]
P3		[1.19, 1.67]	[1.19, 1.67]	[1.4, 7.50]	[1.4, 7.50]	[0, 2.00]	[0, 2.70]	[0, 2.70]	[0, 2.70]
		P4	-	[1.20, 1.67]	-	[1.4, 7.40]	-	-	[0, 3.00]
P5		-	-	-	-	-	-	3.00]	3.00]

Table B.7. Composition constraints (components C1, C2, C3)

Component Product	Example	C1		C2		C3	
		3, 4, 7, 8	9, 12, 14	3, 4, 7, 8	9, 12, 14	3, 4, 7, 8	9, 12, 14
P1	$[r^{min}, r^{max}]$	[0, 0.22]	[0, 0.22]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]
P2		[0, 0.24]	[0, 0.24]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]
P3		[0, 0.25]	[0, 0.25]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]
P4		[0, 0.24]	[0, 0.24]	[0.10, 1]	[0.10, 1]	[0, 1]	[0, 1]
P5		-	[0, 0.30]	-	[0.15, 1]	-	[0, 1]

Table B.8. Composition constraints (components C4, C5, C6)

Component Product	Example	C4				C5		C6	
		3, 4, 7	8	9, 12	14	3, 4, 7, 8	9, 12, 14	3, 4, 7, 8	9, 12, 14
P1	$[r^{min}, r^{max}]$	[0, 0.40]	[0, 0.40]	[0, 0.40]	[0, 0.40]	[0, 0.25]	[0, 0.25]	[0, 0.20]	[0, 0.20]
P2		[0, 0.45]	[0, 0.45]	[0, 0.45]	[0, 0.45]	[0, 0.25]	[0, 0.25]	[0, 0.22]	[0, 0.22]
P3		[0, 0.43]	[0, 0.43]	[0, 0.43]	[0, 0.43]	[0, 0.25]	[0, 0.25]	[0, 0.18]	[0, 0.18]
P4		[0, 0.44]	[0, 0.44]	[0, 0.44]	[0, 0.44]	[0, 0.25]	[0, 0.25]	[0, 0.20]	[0, 0.20]
P5		-	-	[0, 0.40]	[0, 0.40]	-	[0, 0.25]	-	[0, 0.20]

Table B.9. Composition constraints (components C7, C8, C9)

Component Product	Example	C7		C8		C9	
		3, 4, 7, 8	9, 12, 14	3, 4, 7, 8	9, 12, 14	3, 4, 7, 8	9, 12, 14
P1	$[r^{min}, r^{max}]$	[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 0.30]	[0, 0.15]	[0, 0.15]
P2		[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 0.30]	[0, 0.18]	[0, 0.18]
P3		[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 0.30]	[0, 0.20]	[0, 0.20]
P4		[0, 0.25]	[0, 0.25]	[0, 0.30]	[0, 0.30]	[0, 0.16]	[0, 0.16]
P5		-	[0, 0.25]	-	[0, 0.30]	-	[0, 0.17]

Table B.10. Blender data

		ct_{blend} and CV_{blend} at time 0 (kbbbl)			Minimum & Maximum Blending Rate, F_{blend}^{min} and F_{blend}^{max} (kbbbl/h)					Allowable Product (set BP)			
Blender	Example	3, 4, 7	8, 9, 12	14	3, 4	7	8, 9	12	14	3, 4, 7	8	9, 12	14
A		0	0	0	1.5-20	1.5-25	1.5-25	1.5-30	1.5-30	P1-P4	P1-P4	P1-P5	P1-P5
B		-	0	0	-	-	1.5-25	1.5-30	1.5-30	-	P4	P5	P5
C		-	-	0	-	-	-	-	1.5-25	-	-	-	P1-P5

		Minimum Blend Run Length ct_{blend}^{min} (h)													
Blender	Example	P1			P2			P3			P4			P5	
		3, 4, 7	8, 9, 12	14	3, 4, 7	8, 9, 12	14	3, 4, 7	8, 9, 12	14	3, 4, 7	8, 9, 12	14	9, 12	14
A		6	6	6	6	6	6	6	6	6	6	6	6	5	5
B		-	6	6	-	6	6	-	6	6	-	6	6	5	5
C		-	-	6	-	-	6	-	-	6	-	-	6	-	5

Table B.11. Supply profiles of blend components

Example	Supply profile α	Duration (h)	End time FT_{bc}^{end} (h)	Feed Flow Rate to Component Tank F_{bc} (kbbbl/h)								
				C1	C2	C3	C4	C5	C6	C7	C8	C9
3, 4	1	100	100	1.2	0.8	1.2	1.2	0.5	0.8	0.0	0.0	1.0
	2	92	192	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
7	1	80	80	1.2	0.8	1.2	1.2	0.7	0.8	0.0	0.0	1.0
	2	70	150	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	42	192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	1	80	80	1.2	0.8	1.2	1.2	0.5	0.8	0.0	0.0	1.0
	2	70	150	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	42	192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	1	80	80	1.0	0.5	1.0	1.0	0.5	0.5	0.0	0.0	1.0
	2	70	150	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	42	192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	1	50	50	1.0	0.5	1.0	1.0	0.8	0.5	0.0	0.0	1.0
	2	50	100	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	50	150	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.5
	4	42	192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	1	50	50	1.0	0.5	1.0	1.0	0.7	0.5	0.5	0.5	1.0
	2	50	100	0.8	0.6	0.6	0.8	0.5	0.6	0.5	0.5	0.0
	3	50	150	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.5
	4	42	192	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table B.12. Economic data

Component	C1	C2	C3	C4	C5	C6	C7	C8	C9	Scheduling Horizon H (h)	
Cost c_1 (\$/bbl)	20	24	30	25	22	27	50	50	22.5	192	
Swing tank	Tk1	Tk2	Tk3	Tk4	Tk5	Tk6	Tk7	Tk8	Tk9	Tk10	Tk11
Transition Cost c_3 (k\$/instance)	14.5	14.5	14.5	19	19	14.5	19	14.5	19	14.5	14.5
Transition Cost in blender c_2 (k\$/instance)	Penalty coefficients for slack variables										
20	$c_6(n) = \{ [(N - n) / N]^2 \} \cdot (1000 - 100) + 100$										
Demurrage Cost c_5 (k\$/h)	$c_7(n) = 0.5 \cdot c_6(n)$										
2.5	$c_8 = 1000$										

Appendix C: Supporting Information for Chapter 7 and 9

Table C.1. Supply and demand data for scenario #1 (kbbbl)

Day	1	2	3	4	5	6	7
RG	40	40	40	50	40	80	80
PG	30	30	40	20	20	20	20
K1	10	10	10	10	10	15	10
D1	10	10	10	10	10	10	10
D2	10	30	30	30	20	10	20
CO1	30	30	0	0	0	0	0
CO2	0	50	70	0	80	0	70
CO3	40	0	40	50	0	80	70
CO4	0	30	0	0	30	0	0
CO5	0	0	0	30	0	30	0

Table C.2. Supply and demand data for scenario #2 (kbbbl)

Day	1	2	3	4	5	6	7
RG	40	40	0	50	40	80	80
PG	30	30	40	20	20	0	20
K1	10	0	10	10	10	15	10
D1	10	10	10	0	10	10	10
D2	0	30	30	30	20	0	20
CO1	0	30	0	30	20	0	0
CO2	40	50	50	0	70	40	0
CO3	40	0	40	50	0	80	40
CO4	0	30	0	0	30	0	20
CO5	0	0	0	40	0	0	0

Table C.3. Supply and demand data for scenario #3 (kbbbl)

Day	1	2	3	4	5	6	7
RG	40	40	50	80	0	30	90
PG	0	30	60	50	0	30	50
K1	0	10	10	10	10	0	20
D1	10	10	10	0	0	20	20
D2	20	0	30	30	0	30	20
CO1	30	0	30	0	20	0	0
CO2	50	60	70	0	0	60	20
CO3	60	40	80	20	0	40	0
CO4	30	30	0	0	0	0	0
CO5	30	0	0	0	0	0	0

Table C.4. Quality data of crude oils CO1 and CO2 (parameter $qco(s1,qp,s)$)

Crude oil	Quality property	CDU outlet streams (cuts)							
		cdu_pf _ln	cdu_atm _hn	cdu_atm_ kero	cdu_atm _ds	cdu_atm _ago	cdu_vcm _lgo	cdu_vcm _hgo	cdu_vcm _rsd
CO1	sg	0.64	0.75	0.84	0.90	0.93	0.96	1.02	1.07
CO1	sul	0.00	0.09	0.68	1.93	2.61	3.29	4.69	6.08
CO1	ron	71.20	44.80	0.00	0.00	0.00	0.00	0.00	0.00
CO1	mon	69.70	43.10	0.00	0.00	0.00	0.00	0.00	0.00
CO1	arom	0.00	11.51	12.87	0.00	0.00	0.00	0.00	0.00
CO1	rvp	5.80	5.80	0.00	0.00	0.00	0.00	0.00	0.00
CO1	cin	0.00	0.00	34.80	37.80	35.75	33.70	21.95	10.20
CO1	pour	256.00	332.00	345.00	409.00	451.50	494.00	539.00	584.00
CO2	sg	0.67	0.76	0.81	0.85	0.88	0.91	0.94	0.98
CO2	sul	0.00	0.00	0.02	0.22	0.42	0.62	0.96	1.29
CO2	ron	71.80	44.68	0.00	0.00	0.00	0.00	0.00	0.00
CO2	mon	70.30	43.08	0.00	0.00	0.00	0.00	0.00	0.00
CO2	arom	0.00	16.38	20.22	0.00	0.00	0.00	0.00	0.00
CO2	rvp	3.50	3.50	0.00	0.00	0.00	0.00	0.00	0.00
CO2	cin	0.00	0.00	43.70	54.00	55.35	56.70	51.35	46.00
CO2	pour	256.00	332.00	398.00	477.00	520.00	563.00	563.00	563.00

Units: sul (% wt.), rvp (psig), arom (% vol.), pour (°R)

Table C.5. Quality data of crude oils CO3 and CO4 (parameter $qco(s1,qp,s)$)

Crude oil	Quality property	CDU outlet streams (cuts)							
		cdu_pf _ln	cdu_atm _hn	cdu_atm_ kero	cdu_atm _ds	cdu_atm _ago	cdu_vcm _lgo	cdu_vcm _hgo	cdu_vcm _rsd
CO3	sg	0.67	0.76	0.82	0.86	0.88	0.91	0.95	0.99
CO3	sul	0.00	0.00	0.02	0.22	0.43	0.63	0.99	1.35
CO3	ron	72.00	44.90	0.00	0.00	0.00	0.00	0.00	0.00
CO3	mon	70.40	43.10	0.00	0.00	0.00	0.00	0.00	0.00
CO3	arom	0.00	7.86	15.56	0.00	0.00	0.00	0.00	0.00
CO3	rvp	4.20	4.20	0.00	0.00	0.00	0.00	0.00	0.00
CO3	cin	0.00	0.00	40.50	53.40	54.90	56.40	48.95	41.50
CO3	pour	256.00	332.00	393.00	473.00	518.50	564.00	558.50	553.00
CO4	sg	0.66	0.75	0.82	0.89	0.93	0.97	1.01	1.04
CO4	sul	0.02	0.07	0.33	1.45	2.40	3.34	4.58	5.81
CO4	ron	69.50	46.80	0.00	0.00	0.00	0.00	0.00	0.00
CO4	mon	68.00	45.30	0.00	0.00	0.00	0.00	0.00	0.00
CO4	arom	0.63	14.21	10.84	24.94	28.06	31.18	31.38	31.58
CO4	rvp	3.70	3.70	0.00	0.00	0.00	0.00	0.00	0.00
CO4	cin	0.00	0.00	38.40	40.20	20.10	0.00	0.00	0.00
CO4	pour	256.00	332.00	389.00	413.00	456.50	500.00	552.50	605.00

Units: sul (% wt.), rvp (psig), arom (% vol.), pour (°R)

Table C.6. Quality data of crude oil CO5 (parameter $qco(s1,qp,s)$)

Crude oil	Quality property	CDU outlet streams (cuts)							
		cdu_pf _ln	cdu_atm _hn	cdu_atm_ kero	cdu_atm _ds	cdu_atm _ago	cdu_vcm _lgo	cdu_vcm _hgo	cdu_vcm _rsd
CO5	sg	0.65	0.75	0.81	0.86	0.90	0.94	1.01	1.07
CO5	sul	0.03	0.21	0.93	2.32	3.16	4.00	5.87	7.74
CO5	ron	70.20	47.50	0.00	0.00	0.00	0.00	0.00	0.00
CO5	mon	69.40	46.00	0.00	0.00	0.00	0.00	0.00	0.00
CO5	arom	0.00	14.60	25.80	0.00	0.00	0.00	0.00	0.00
CO5	rvp	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00
CO5	cin	0.00	0.00	0.00	54.00	55.35	56.70	51.35	46.00
CO5	pour	256.00	332.00	415.00	471.00	554.50	638.00	671.00	704.00

Units: sul (% wt.), rvp (psig), arom (% vol.), pour (°R)

Table C.7. Fixed values for quality properties – part 1 (parameter $q^{\text{fix}}(s,qp,n)$ for all n)

Stream	Quality property							
	sg	ron	mon	rvp	arom	sul	cin	Pour
Alkylate	0.7030	95.0	91.7	6.6	0.0	0.0000	0	0
All n-butane streams	0.5840	93.8	90.0	138.0	0.0	0.0000	0	0
nht_hn	C	39.8	39.5	0.8	13.1	C		
dht_n	0.7732	55.0	54.0	1.3	22.0	0.0120		
dht_ds	C					C	54	458
goht_hc_n	0.7732	55.0	54.0	1.3	22.0	0.0221		
goht_hc_ds	0.8473					0.0520	54	450
hc_feed	C					C		550
goht_fcc_n	0.7732	55.0	54.0	1.3	22.0	0.0221		
goht_fcc_ds	0.8473					0.0520	54	450
fcc_feed	C					C		550
rht_n	0.7732	55.0	54.0	1.3	22.0	0.0471		
rht_ds	0.8473					0.1108	54	450
FuelOil	C					C		510
reformateA	0.8180	102.0	90.3	6.6	40.0	0.0000		
reformateB	0.8180	93.0	83.4	4.4	40.0	0.0000		
hcgm_ln	0.6601	82.4	79.5	13.0	2.0	0.0005		
hcgm_hn	0.7658	53.5	53.1	0.5	10.0	0.0010		
hckm_ln	0.6641	84.0	80.8	13.0	2.0	0.0005		
hckm_hn	0.7345	61.9	61.7	1.0	7.0	0.0010		
hckm_kero	0.8144				18.5	C		394
hcdm_ln	0.6673	85.3	81.8	13.0	2.0	0.0005		
hcdm_hn	0.7644	65.4	64.9	0.8	6.0	0.0010		
hcdm_diesel	0.8360					C	51	405

C = Computed within the model

Units: sul (% wt.), rvp (psig), arom (% vol.), pour (°R)

Table C.8. Fixed values for quality properties – part 2 (parameter $q^{\text{fix}}(s,qp,n)$ for all n)

Stream	Quality property							
	sg	ron	mon	rvp	arom	Sul	cin	pour
fccA_n	0.7440	91.5	80.7	6.4	0.0	C		
fccA_lco	0.9240					C	51	460
fccA_hco	0.9710					C	51	480
fccB_n	0.7450	92.3	81.3	6.4	0.0	C		
fccB_lco	0.9350					C	51	430
fccB_hco	1.0450					C	51	450

C = Computed within the model

Units: sul (% wt.), rvp (psig), arom (% vol.), pour (°R)

Table C.9. Fixed yields for hydrotreaters (parameter $Yield_{HTU}(u,s)$)

Unit	Outlet stream	Yield (% vol.)
nht	nht_nbut	0.08
	nht_hn	100.01
dht	dht_nbut	0.02
	dht_n	0.08
	dht_ds	99.90
goht_hc	goht_hc_nbut	0.09
	goht_hc_n	0.88
	goht_hc_ds	7.19
	hc_feed	92.31
goht_fcc	goht_fcc_nbut	0.09
	goht_fcc_n	0.88
	goht_fcc_ds	7.19
	fcc_feed	92.31
rht	rht_nbut	0.50
	rht_n	2.46
	rht_ds	7.34
	FuelOil	88.68

Table C.10. Fixed yields for processing units (parameter $Yield_{PV}(u,s)$)

Unit	Outlet stream	Yield (% vol.)
reformerA	refA_nbut	7.42
	reformateA	70.69
reformerB	refB_nbut	4.44
	reformateB	80.99
hc_gm	hcgm_nbut	6.83
	hcgm_ln	33.30
	hcgm_hn	70.76
hc_km	hckm_nbut	4.30
	hckm_ln	19.76
	hckm_hn	35.02
	hckm_kero	54.79
hc_dm	hcdm_nbut	2.78
	hcdm_ln	10.85
	hcdm_hn	29.37
	hcdm_diesel	70.88
fccA	fccA_nbut	2.19
	fccA_n	58.03
	fccA_lco	17.39
	fccA_hco	7.62
	fccA_coke	5.00
fccB	fccB_nbut	2.37
	fccB_n	62.75
	fccB_lco	10.43
	fccB_hco	4.57
	fccB_coke	6.63

Table C.11. Sulfur removal factor (parameter $SRF^{\text{fix}}(u,s)$)

Unit	Outlet stream	$SRF^{\text{fix}}(u,s)$
hc_km	hckm_kero	0.008
hc_dm	hcdm_diesel	0.020
fccA	fccA_n	0.130
	fccA_lco	0.500
	fccA_hco	0.750
fccB	fccB_n	0.100
	fccB_lco	0.750
	fccB_hco	0.900

Table C.12. Initial quality of storage tanks (parameter $q^{\text{ini}}(s,qp)$)

Tank	Outlet stream	Quality property							
		sg	ron	mon	rvp	arom	sul	cin	pour
tank_srh	srhn	0.750	65	63	1	7	0		
tank_hcln	hcln	0.667	85.3	81.8	13	2	0		
tank_hchn	hchn	0.764	65.4	64.9	0.8	6	0.0001		
tank_srln	srln	0.670	78	68	5	20	0		
tank_fccnA	fccnA	0.744	90	80	3	20	0.0001		
tank_fccnB	fccnB	0.745	92	82	3	20	0.0001		
tank_refA	refA	0.818	102	90.3	6.6	40	0		
tank_refB	refB	0.818	93	83.4	4.4	40	0		
tank_srk	srk	0.814				20	0.3	47	400
tank_hck	hck	0.814				20	0.3	47	400
tank_ds	ds1	0.828				20	0.001	47	470
tank_hcds	hcds	0.836				20	0.001	47	470
tank_lcoA	lcoA	0.924				20	0.001	47	470
tank_hcoA	hcoA	0.971				20	0.001	47	470
tank_lcoB	lcoB	0.935				20	0.001	47	470
tank_hcoB	hcoB	1.045				20	0.001	47	470
tank_srds	srds	0.830				20	0.002	47	470

Units: sul (% wt.), rvp (psig), arom (% vol.), pour (°R)

Table C.13. Minimum and maximum feed flow rates to the units (kbbbl/day)

Unit	$VF^{\text{min}}(u)$	$VF^{\text{max}}(u)$
cdu	72	120
nht	1	40
dht	1	40
goht_hc	1	40
goht_fcc	1	40
rht	0	60
reformerA	4	40
reformerB	4	40
fccA	4	40
fccB	4	40
hc_gm	4	40
hc_km	4	40
hc_dm	4	40

For all mixers and splitters: $VF^{\text{min}}(u) = 0$, $VF^{\text{max}}(u) = 40$ kbbbl/day.

Table C.14. Initial, minimum, and maximum inventory levels (kbbbl)

Tank	$V^{\text{ini}}(t)$	$V^{\text{min}}(t)$	$V^{\text{max}}(t)$
tank_CO1	100	10	200
tank_CO2	50	10	200
tank_CO3	80	10	200
tank_CO4	30	10	200
tank_CO5	10	10	200
tank_rgas	100	20	200
tank_pgas	20	20	200
tank_kero	20	20	200
tank_D1	40	20	200
tank_D2	40	20	200
tank_srhcn	5	5	50
tank_hcln	68	5	100
tank_nbut	25	5	100
tank_hchn	74	5	100
tank_srln	16	5	50
tank_refA	27	5	100
tank_refB	24	5	100
tank_fccnA	35	5	100
tank_fccnB	26	5	100
tank_ds	35	5	100
tank_lcoA	32	5	100
tank_hcoA	19	5	100
tank_lcoB	26	5	100
tank_hcoB	30	5	100
tank_hcds	5	5	100
tank_srk	19	5	100
tank_hck	20	5	100
tank_srds	20	5	50

Table C.15. Tank, mixer, and unit subsets

ID	Tanks	Description
T1	tank_CO1, tank_CO2, tank_CO3, tank_CO4, tank_CO5	No quality computation
T2	tank_rgas, tank_pgas, tank_kero, tank_D1, tank_D2, tank_nbut, tank_hcln, tank_hchn, tank_refA, tank_refB	Quality of the outlet stream is equal to the quality of the inlet stream
T3	tank_srln, tank_srhcn, tank_srk, tank_srds	Quality properties are computed with blending equations
T4	tank_ds, tank_hcds, tank_lcoA, tank_hcoA, tank_lcoB, tank_hcoB, tank_srds, tank_hck, tank_fccA, tank_fccB	Only sulfur content is computed with blending equations, all the other properties of the outlet stream are equal to those of the inlet stream
ID	Mixers	Description
MX1	mixer6_coke	No quality computation
MX2	mixer1_nbut	Quality of the outlet stream is set equal to a specified value
MX3	mixer2_naphtha, mixer3_diesel, mixer4_hcln, mixer5_hchn	Quality of the outlet stream is equal to the quality of the main inlet stream
MX4	mixer_nht, mixer_ds_ago, mixer_tgo_hc, mixer_tgo_fcc	Quality properties are computed with blending equations
ID	Units	Description
CDU	cdu	Crude distillation units
HTU	nht, dht, goth_fcc, goth_hc, rht	Hydrotreating units
MU	reformerA, reformerB, fccA, fccB, hc_gm, hc_km, hc_dm	Units representing an operating mode
PU	reformerA, reformerB, fccA, fccB, hc_gm, hc_km, hc_dm	Processing units (reformer, fluid catalytic cracker, hydrocracker)
RU	REFORMER, HYDROCRACKER, FLUID_CATALYTIC_CRACKER	Physical unit with different operating modes

Values used for remaining parameters

Parameters $HTS^{\min}(u)$ and $HTS^{\max}(u)$ are equal to 0.8 and 0.998, respectively, for all hydrotreaters (i.e. $u \in \mathbf{HTU}$).

Parameter $RSR^{\max}(u)$ is equal to 50 ton/day for all hydrotreaters (i.e. $u \in \mathbf{HTU}$).

Parameters $VR^{\min}(t)$ and $VR^{\max}(t)$ are equal to 0 and 300 kbbl/day, respectively, for all storage tanks.

Parameters $VFTR^{\min}(ru)$ and $VFTR^{\max}(ru)$ are equal to 4 and 40 kbbl/day, respectively, for all units $ru \in \mathbf{RU}$. Parameter $VF^{\min}(u)$ is equal to 4 kbbl for all units $u \in \mathbf{PU}$.

Parameters $VBR^{\min}(b)$ and $VBR^{\max}(b)$ are equal to 10 and 120 kbbl/day, respectively, for all blenders. Parameter $Vblend^{\min}(s)$ is equal to 3 kbbl for all products $s:(b,s) \in \mathbf{BO}$.

Quality blending equations for storage tanks

The equations in this subsection are the actual form of eq. 31 shown in the paper for each type of tank. There are four classes of tanks considered. There are tanks that only require the volumetric balance (i.e. mathematical model given by eqs. 27-30 from the paper) since it is assumed that 1) the tank has a single inlet, 2) the quality of the inlet stream is known and it does not change with time, 3) the quality of the initial material in the tank is the same as that of the inlet stream, and 4) the quality of the outlet stream is used in the next unit, thus it is not necessary to include it here. These type of tanks are assigned to set **T1**.

Set **T2** includes the tanks for which we assume that the quality of the outlet stream is equal to the quality of the inlet stream. Therefore, eq. 31 for a tank from set **T2** is replaced by eq. C1.

$$q(s, qp, n) = q(s1, qp, n) \quad \forall t \in \mathbf{T2}, n, s1 : (t, s1) \in \mathbf{T1}, s : (t, s) \in \mathbf{TO}, qp : (s, qp) \in \mathbf{SQ}$$

(C1)

Set **T3** consists of the tanks that include the quality balance equations for all the possible quality properties. Thus, eq. 31 for tanks belonging to set **T3** is replaced by eqs. C2-C15.

$$QVFlow(s, qp, n) = q(s, qp, n) \cdot VFlow(s, n) \quad \forall t \in \mathbf{T3}, n, s : (t, s) \in \mathbf{T1}, qp : (s, qp) \in \mathbf{SQ}$$

(C2)

$$QVTank(t, qp, n) = q(s, qp, n) \cdot V(t, n) \quad \forall t \in \mathbf{T3}, n, s : (t, s) \in \mathbf{TO}, qp : (s, qp) \in \mathbf{SQ}$$

(C3)

$$q(s, qp, n) \cdot denV(t, n) = numQV(t, qp, n)$$

$$\forall t \in \mathbf{T3}, n, s : (t, s) \in \mathbf{TO}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLV}$$

(C4)

$$q(s, qp, n) \cdot denVSG(t, n) = numQVSG(t, qp, n)$$

$$\forall t \in \mathbf{T3}, n, s : (t, s) \in \mathbf{TO}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW}$$

(C5)

$$denV(t, n) = \sum_{s \in \mathbf{TI}} VFlow(s, n) + V^{ini}(t) \quad \forall t \in \mathbf{T3}, n = 1$$

(C6)

$$denV(t, n) = \sum_{s \in \mathbf{TI}} VFlow(s, n) + V(t, n-1) \quad \forall t \in \mathbf{T3}, n > 1$$

(C7)

$$denVSG(t, n) = \sum_{s \in \mathbf{TI}} QVFlow(s, qp, n) + V^{ini}(t) \cdot q^{ini}(s1, qp) \quad \forall t \in \mathbf{T3}, n = 1, qp = \text{sg}, s1 \in \mathbf{TO}$$

(C8)

$$denVSG(t, n) = \sum_{s \in \mathbf{TI}} QVFlow(s, qp, n) + QVTank(t, qp, n-1) \quad \forall t \in \mathbf{T3}, n > 1, qp = \text{sg}$$

(C9)

$$QVFlowSG(s, qp, n) = QVFlow(s, qp, n) \cdot q(s, qp1, n)$$

$$\forall t \in \mathbf{T3}, n, s : (t, s) \in \mathbf{TI}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW}, qp1 = \text{sg}$$

(C10)

$$QVTankSG(t, qp, n) = QVTank(t, qp, n) \cdot q(s, qp1, n)$$

$$\forall t \in \mathbf{T3}, n < n^{\max}, s : (t, s) \in \mathbf{TO}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW}, qp1 = \text{sg}$$

(C11)

$$\begin{aligned} numQV(t, qp, n) &= \sum_{s \in \mathbf{II}} QVFlow(s, qp, n) + V^{ini}(t) \cdot q^{ini}(s1, qp) \\ \forall t \in \mathbf{T3}, n = 1, s1 \in \mathbf{TO}, qp : (s1, qp) \in \mathbf{SQ}, qp \in \mathbf{QLV} \end{aligned}$$

(C12)

$$\begin{aligned} numQV(t, qp, n) &= \sum_{s \in \mathbf{II}} QVFlow(s, qp, n) + QVTank(t, qp, n - 1) \\ \forall t \in \mathbf{T3}, n > 1, qp : (s1, qp) \in \mathbf{SQ}, qp \in \mathbf{QLV} \end{aligned}$$

(C13)

$$\begin{aligned} numQVSG(t, qp, n) &= \sum_{s \in \mathbf{II}} QVFlowSG(s, qp, n) + V^{ini}(t) \cdot q^{ini}(s1, qp) \cdot q^{ini}(s1, qp1) \\ \forall t \in \mathbf{T3}, n = 1, s1 \in \mathbf{TO}, qp : (s1, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW}, qp1 = sg \end{aligned}$$

(C14)

$$\begin{aligned} numQVSG(t, qp, n) &= \sum_{s \in \mathbf{II}} QVFlowSG(s, qp, n) + QVTankSG(t, qp, n - 1) \\ \forall t \in \mathbf{T3}, n > 1, qp : (s1, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW} \end{aligned}$$

(C15)

Finally, the tanks that only require the quality balance equations for the sulfur content property ('sul'), and assume all the other properties of the outlet stream to be equal to the inlet stream, conform the set **T4**. Therefore, eq. 31 for tanks from set **T4** is replaced by eqs. C16-C25.

$$\begin{aligned} QVFlow(s, qp, n) &= q(s, qp, n) \cdot VFlow(s, n) \quad \forall t \in \mathbf{T4}, n, s : (t, s) \in \mathbf{II}, qp \in \{sg, sul\} \end{aligned}$$

(C16)

$$\begin{aligned} QVTank(t, qp, n) &= q(s, qp, n) \cdot V(t, n) \quad \forall t \in \mathbf{T4}, n, s : (t, s) \in \mathbf{TO}, qp \in \{sg, sul\} \end{aligned}$$

(C17)

$$\begin{aligned} q(s, qp, n) \cdot denVSG(t, n) &= numQVSG(t, qp, n) \quad \forall t \in \mathbf{T4}, n, s : (t, s) \in \mathbf{TO}, qp = sul \end{aligned}$$

(C18)

$$\begin{aligned} denVSG(t, n) &= \sum_{s \in \mathbf{II}} QVFlow(s, qp, n) + V^{ini}(t) \cdot q^{ini}(s1, qp) \quad \forall t \in \mathbf{T4}, n = 1, qp = sg, s1 \in \mathbf{TO} \end{aligned}$$

(C19)

$$\text{denVSG}(t, n) = \sum_{s \in \mathbf{TI}} QVFlow(s, qp, n) + QVTank(t, qp, n-1) \quad \forall t \in \mathbf{T4}, n > 1, qp = \text{sg}$$

(C20)

$$QVFlowSG(s, qp, n) = QVFlow(s, qp, n) \cdot q(s, qp1, n)$$

$$\forall t \in \mathbf{T4}, n, s : (t, s) \in \mathbf{TI}, qp = \text{sul}, qp1 = \text{sg}$$

(C21)

$$QVTankSG(t, qp, n) = QVTank(t, qp, n) \cdot q(s, qp1, n)$$

$$\forall t \in \mathbf{T4}, n < n^{\max}, s : (t, s) \in \mathbf{TO}, qp = \text{sul}, qp1 = \text{sg}$$

(C22)

$$\text{numQVSG}(t, qp, n) = \sum_{s \in \mathbf{TI}} QVFlowSG(s, qp, n) + V^{\text{ini}}(t) \cdot q^{\text{ini}}(s1, qp) \cdot q^{\text{ini}}(s1, qp1)$$

$$\forall t \in \mathbf{T4}, n = 1, s1 \in \mathbf{TO}, qp = \text{sul}, qp1 = \text{sg}$$

(C23)

$$\text{numQVSG}(t, qp, n) = \sum_{s \in \mathbf{TI}} QVFlowSG(s, qp, n) + QVTankSG(t, qp, n-1)$$

$$\forall t \in \mathbf{T4}, n > 1, qp = \text{sul}$$

(C24)

$$q(s, qp, n) = q(s1, qp, n)$$

$$\forall t \in \mathbf{T4}, n, s1 : (t, s1) \in \mathbf{TI}, s : (t, s) \in \mathbf{TO}, qp : (s, qp) \in \mathbf{SQ}, qp \neq \text{sul}$$

(C25)

Output flow and quality constraints for mixers

Eq. 34 from the paper takes the form given by eq. C26 for all mixers.

$$VFlow(s, n) = VF(u, n) \quad \forall n, u \in \mathbf{MX}, s : (u, s) \in \mathbf{UO}$$

(C26)

Eq. 35 from the paper is replaced according to the mixer type. The general set of mixers **MX** is divided into the following subsets: **MX1**, **MX2**, **MX3** and **MX4**. **MX1** are the mixers for which we only need a material balance around them; i.e. their mathematical model is composed by eqs. 32-33 from the paper and eq. C26.

MX2 is the subset of mixers for which we fix the value of the qualities at the outlet to a pre-specified value using eq. C27.

$$q(s, qp, n) = q^{\text{fix}}(s, qp, n) \quad \forall n, u \in \mathbf{MX2}, s : (u, s) \in \mathbf{UO}, qp : (s, qp) \in \mathbf{SQ} \quad (\text{C27})$$

MX3 is composed by the mixers that set the quality of their outlet streams equal to the quality of their corresponding main inlet stream, as expressed by eq. C28.

$$q(s, qp, n) = q(s1, qp, n) \quad \forall n, u \in \mathbf{MX3}, s : (u, s) \in \mathbf{UO}, s1 : (u, s1) \in \mathbf{UMI}, qp : (s, qp) \in \mathbf{SQ} \quad (\text{C28})$$

MX4 is constituted by mixers that consider the quality balance using eq. C29-C33.

$$QVFlow(s, qp, n) = VFlow(s, n) \cdot q(s, qp, n) \quad \forall n, u \in \mathbf{MX4}, s : (u, s) \in \mathbf{UI}, qp : (s, qp) \in \mathbf{SQ} \quad (\text{C29})$$

$$QVFlowSG(s, qp, n) = QVFlow(s, qp, n) \cdot q(s, qp1, n) \quad \forall n, u \in \mathbf{MX4}, s : (u, s) \in \mathbf{UI}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW}, qp1 = \text{sg} \quad (\text{C30})$$

$$VF(u, n) \cdot q(s, qp, n) = \sum_{s1: (u, s1) \in \mathbf{UI}} QVFlow(s1, qp, n) \quad \forall n, u \in \mathbf{MX4}, s : (u, s) \in \mathbf{UO}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLV} \quad (\text{C31})$$

$$\text{denVSG}(u, n) \cdot q(s, qp, n) = \sum_{s1: (u, s1) \in \mathbf{UI}} QVFlowSG(s1, qp, n) \quad \forall n, u \in \mathbf{MX4}, s : (u, s) \in \mathbf{UO}, qp : (s, qp) \in \mathbf{SQ}, qp \in \mathbf{QLW} \quad (\text{C32})$$

$$\text{denVSG}(u, n) = \sum_{s1: (u, s1) \in \mathbf{UI}} QVFlow(s1, qp, n) \quad \forall n, u \in \mathbf{MX4}, s : (u, s) \in \mathbf{UO}, qp = \text{sg} \quad (\text{C33})$$

Output flow and quality constraints for hydrotreaters

Eq. 34 from the paper takes the form given by eq. C34 for all hydrotreaters.

$$VFlow(s, n) = Yield_{HTU}(u, s) \cdot VF(u, n) \quad \forall n, u \in \mathbf{HTU}, s : (u, s) \in \mathbf{UO}$$

(C34)

For all the hydrotreaters, eq. 35 from the paper is replaced by eqs. C35-C41.

$$MFlow(s, n) = q(s, qp, n) \cdot VFlow(s, n) \quad \forall n, u \in \mathbf{HTU}, s : (u, s) \in \mathbf{UI} \cup \mathbf{UO}, qp = \text{sg}$$

(C35)

$$\sum_{s1: (u, s1) \in \mathbf{UI}} q(s1, qp, n) \cdot MFlow(s1, n) = \sum_{s: (u, s) \in \mathbf{UO}} q(s, qp, n) \cdot MFlow(s, n) + RS(u, n)$$

$$\forall n, u \in \mathbf{HTU}, qp = \text{sul}$$

(C36)

$$RS(u, n) \leq RSR^{\max}(u) \cdot L(n) \quad \forall n, u \in \mathbf{HTU}$$

(C37)

$$(1 - HTS^{\max}(u)) \cdot q(s, qp, n) \leq q(s, qp, n) \leq (1 - HTS^{\min}(u)) \cdot q(s, qp, n)$$

$$\forall n, u \in \mathbf{HTU}, s : (u, s) \in \mathbf{UOS}, qp = \text{sul}$$

(C38)

$$q(s, qp, n) = 0.98 \cdot q(s1, qp, n) \quad \forall n, u \in \mathbf{HTU}, s1 : (u, s1) \in \mathbf{UI}, s : (u, s) \in \mathbf{UOS}, qp = \text{sg}$$

(C39)

$$q(s, qp, n) = q^{\text{fix}}(s, qp, n) \quad \forall n, u \in \mathbf{HTU}, s : (u, s) \in \mathbf{UO} / \mathbf{UOS}, qp : (s, qp) \in \mathbf{SQ}$$

(C40)

$$\forall n, u \in \mathbf{HTU}, s : (u, s) \in \mathbf{UOS}, qp : (s, qp) \in \mathbf{SQ}, qp \neq \{\text{sg}, \text{sul}\}$$

(C41)

Output flow and quality constraints for other processing units (reformer, hydrocracker, fluid catalytic cracker)

Eq. 34 from the paper takes the form given by eq. C42 for the processing units from set **PU**.

$$VFlow(s, n) = Yield_{pu}(u, s) \cdot VF(u, n) \quad \forall n, u \in \mathbf{PU}, s : (u, s) \in \mathbf{UO}$$

(C42)

For all the units from set **PU**, eq. 35 from the paper is replaced by eqs. C43-C44.

$$q(s, qp, n) = SRF^{\text{fix}}(u, s) \cdot q(s1, qp, n) \quad \forall n, u \in \mathbf{PU}, s : (u, s) \in \mathbf{UO}, qp = \text{sul}$$

(C43)

$$q(s, qp, n) = q^{\text{fix}}(s, qp, n) \quad \forall n, u \in \mathbf{PU}, s : (u, s) \in \mathbf{UO}, qp : (s, qp) \in \mathbf{SQ}, qp \neq \text{sul}$$

(C44)

Bilinear terms

The bilinear terms appear in eqs. C2-C5, C10-C11, C16-C18, C21-C22, C29-C32, and C35-C36.