OPTIMAL MULTI-TIME PERIOD GASOLINE BLENDING

OPTIMAL MULTI-TIME PERIOD GASOLINE BLENDING

By

SHEFALI KULKARNI, B.E.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Applied Science

McMaster University

For all the Unparalleled Love and Blessings

Dedicated to

My Parents And Grandparents

MASTER OF APPLIED SCIENCE(2009) McMaster University

(Computational Engineering and Science) Hamilton, ON

TITLE: Optimal Multi-Time Period Gasoline Blending

AUTHOR: Shefali Kulkarni, B.E. (Gujarat University)

SUPERVISOR: Dr. Vladimir Mahalec

NUMBER OF PAGES: viii,115

Abstract

Multi Time period Gasoline blending is an example of multipurpose production system that is designed to produce multiple products by switching from one product to another. Various factors such as demand for gasoline, availability of supply component, and blend recipes vary with time. Task of the gasoline blender is to decide how much of each product to produce at what point in time (lot sizing) and what should be the blend recipe in order to minimize overall cost (optimize the blend recipe). The production plans need to account for set-up times between blends and to minimize switching between different product blends. Traditional optimization techniques provide a single optimal solution. This research is using evolutionary optimization algorithm called differential evolution to identify multiple solutions that all have the same total cost but offer the blend planner multiple choices in terms of how much of a given product to blend at what point in time.

Acknowledgements

This thesis has been a dedication of two years of work and would have been impossible to accomplish without the guidance and support of my supervisor, Dr. Vladimir Mahalec. I express my sincere gratitude to him.

There have been other people instrumental - directly or indirectly - during this course of two years. I would like to thank Dr. Antonie Deza, Jemmy Hu and Dr. Ned Nedialkov whose courses directed me in the right direction for this work. I would like to thank my colleagues and my team members - in particular Jing Wang, Charumitra Pujari, Aastha Trehan, friends from home, among others who have been kind enough to support me through out. Words of gratitude do not suffice for the support, and blessings of my parents and grandparents while my brother, Ashutosh, ensured all the smiles during tough times. Last but not the least I would like to express my gratitude to my significant other, Nirav Thaker, who has given me the guidance, and the strength to overcome the hurdles.

Contents

Chapter 1

Introduction

Gasoline is the most important and critical refinery product. 60-70% of the revenues of a refinery are generated from its gasoline production [29]. Hence, its production is considered as key to providing competitive edge to the refineries [18]. The demand of gasoline varies in time just as the properties of different grades of gasoline vary with geography of where the gasoline will be used.

This thesis will discuss Refinery Process and a Gasoline Blending System in the first chapter. It will give an introduction to Evolutionary Optimization technique in the next chapter. Then it will subsequently discuss the problem formulation, assumptions, data, and the algorithm to calculate fitness function. Numerical experiments and results will be discussed next followed by multi-process implementation of the differential evolution. The appendix A will give an overview of the software that was developed to aid

in the process to solve this problem.

1.1 The Process of Oil Refining

The crude oil obtained from the ground undergoes various steps of chemical processing before it becomes usable. The process of purification of crude oil into more usable products like gasoline, diesel, kerosene etc. is called as Oil Refining. Crude oil is made of hydrocarbons i.e molecules that contain hydrogen and carbon atoms. This makes crude oil a very versatile raw material since hydrocarbons have lot of energy and they can combine to form very long linear or complex chain of components. In order to extract some usable hydrocarbons from the crude oil, various chemical processes have been invented.

Oil refining consists of following major steps:

1. Crude distillation.
2. Conversion of intermediate streams into gasoline and diesel blend components.
3. Blending of final components e.g. various grades of gasoline and diesel fuels.

In a crude oil distillation process, the crude oil is heated and various chains of hydrocarbons are withdrawn from the unit based on their boiling temperature. The components obtained as a results of this process (in increasing

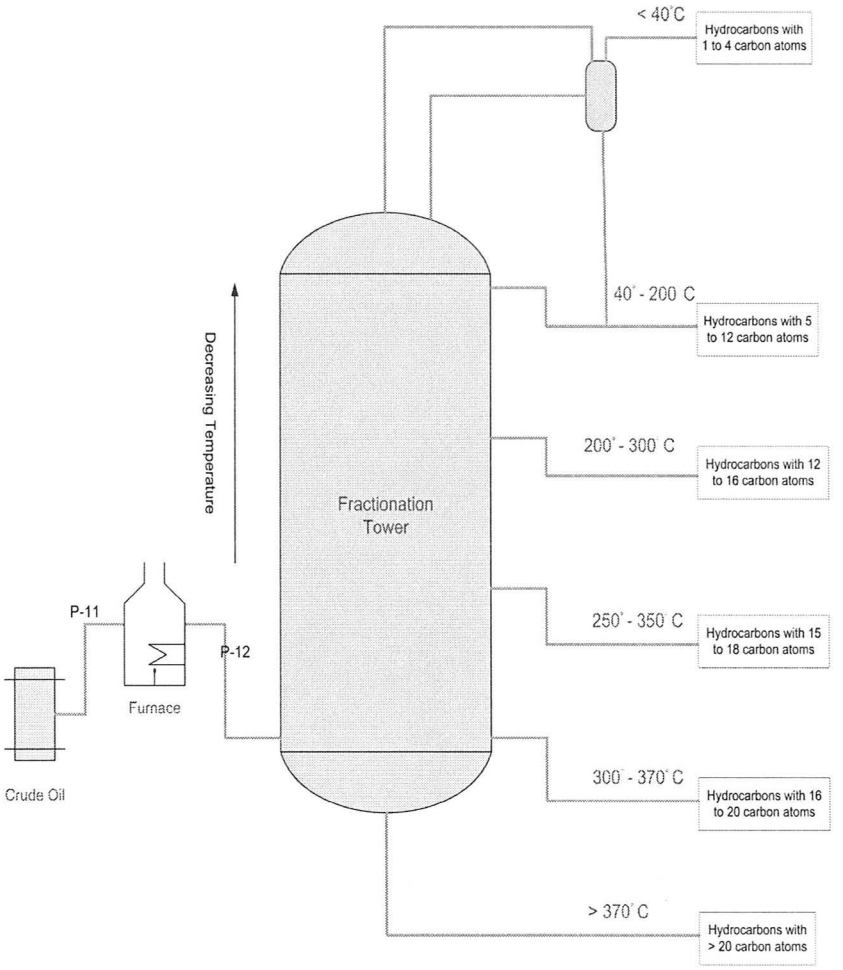

Figure 1.1: Crude Oil Distillation. Re-adapted from [1]

order of their boiling point or length of H-C chain) are Petroleum, Naphtha, Gasoline, Kerosene, Gas Oil, Lubricating Oil, Heavy Gas, and Residuals. They are categorized as light distillates (LPG, gasoline, naphtha), medium distillates (kerosene, diesel), and heavy distillates and residue (heavy fuel oil, lubricating oils, wax, tar). As seen in figure 1.2, the process of crude oil refining involves several steps. The important units are explained below.

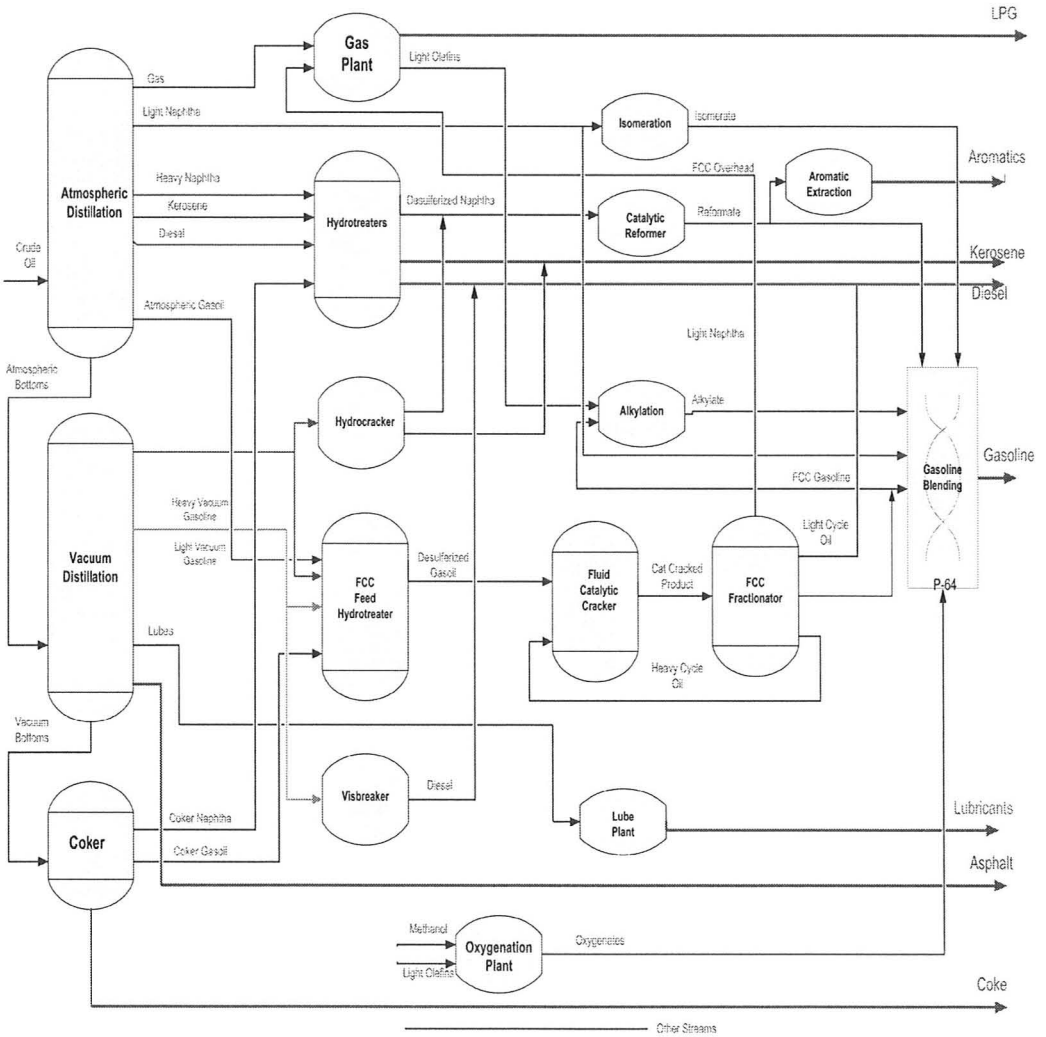

Figure 1.2: The Process of Oil Refining. Adapted from [26]

- Atmospheric Distillation: Crude oil is boiled and sent to the atmospheric distillation tower where is undergoes separation into end products such as gas, gasoline, naphtha, kerosene, and gas oil. The bottoms of this unit is then sent to the Vacuum Distillation Unit.
- Vacuum Distillation: In this unit the pressure above the feed is re-

duced than its ambient pressure which distills the atmospheric bottoms to gas oils and asphalt. The vacuum bottoms are fed to Coker.

- Coker: The purpose of this unit is to get feed from vacuum bottoms and break long chain of hydrocarbons into short chains thereby producing low molecular weight products like naphtha, gas-oils and coke.
- Hydrotreaters: This unit is used to carry out the catalytic chemical process of removing sulfur from its feed, called as hydrodesulfurization. With the addition of hydrogen this process breaks the carbon-sulfur bond to produce hydrogen-sulfur bonds. This process helps maintain the sulfur content in the gasoline in specified limits.
- Hydrocracker: Cracking is a process of breaking carbon-carbon bonds in heavy hydrocarbons or complex organic compounds to produce lighter hydrocarbons in presence of some catalyst. In Hydrocracker this process is carried out in the presence of high pressure of hydrogen gas. Just as in Hydrotreater, this process removes sulfur like components from the feed compounds.
- Visbreaker: A visbreaker thermally cracks large hydrocarbon molecules in the oil by heating in a furnace and thereby reducing its viscosity. This produces small quantities of light hydrocarbons. The process is called so because it cracks the residual oil by reducing its viscosity.
- Catalytic Reformer: This process converts refinery naphtha of low

octane number into product called reformate that has high octane numbers. Reformate is an important component for a high octane petrol. The process re-structures the naphtha in to more complex molecule in order to produce high octane rating molecule. Alkanes are obtained as byproducts in small amounts.

1.2 Gasoline Blending Process

As seen in the figure 1.2, Gasoline Blending is the last stage of crude-oil refining process. The end product of this stage are various products like the aviation fuel, diesels, and gasoline. Gasoline is a complex mixture of hydrocarbons obtained by the combination of two or more refinery products (with some additives). Gasoline blends vary widely in their composition; even those having the same octane number can be widely different. The various properties of gasoline depend on the types and relative proportions of each of their constituents[20].

Gasoline production is a batch process where the end-point is fixed in terms of the amount of gasoline to blend or the blend duration [18]. The typical grades of gasoline, distinguished by their octane numbers, are Regular (87), Mid-grade(89), and Premium(91 and 93). Superficially speaking, the aim of the blender is to combine the available blend components in a way so as to produce different grades of gasolines. However, numerous challenges are faced by a gasoline blend optimizer.

- Quality Specifications: Any gasoline grade must meet the quality specifications that may depend on geographical location of the shipping area. (Octane and Reid Vapor Pressure being the most important ones.). Quality giveaway is the difference between the true value of the blend and the specification limit of that property[27]. It is estimated that consistent octane giveaway of 0.1 octane number can cost refinery several million dollars a year [18]. Hence all the blends must be produced on time with minimal giveaway[27]. Re-blend might be necessary when a blend does not meet the quality specification. This may result into a significant loss to the refinery as it results into additional usage of crucial refinery resources (storage, blender etc)[18].
- Scheduling of blends: Main concerns in scheduling are (a) to obtain a feasible schedule satisfying all product demands. (b) to meet the goals set by the long-range planning and (c) to optimize the operation of all blending facilities itself (e.g. to minimize product and recipe changeovers - minimize task switching)[10].
- Inventory management: Inventory management of blend component tanks (managing multi-product or swing tanks) [13], availability of blend feed stocks and gasoline storage facilities need to be considered within the volumetric constraints of the refinery equipment capacities.

Thus, an ideal blend optimizer would provide the maximum possible profit

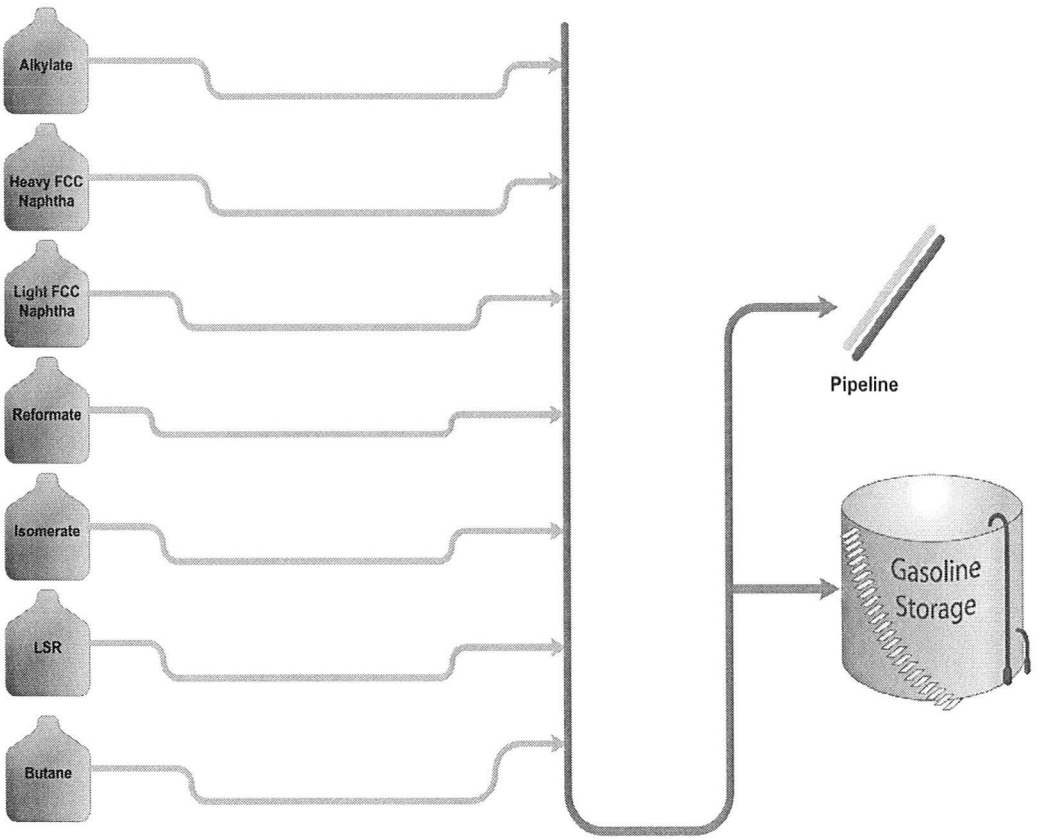

Figure 1.3: Gasoline Blending

from the blending process, while meeting all blend quality specifications, as well as satisfying demand, supply and availability limits[18]. Any gasoline automation system usually has 3 layers:

1. An off-line scheduler plans the refinery operations on a long (months), intermediate (weeks) or short (days) term range.
2. An online optimizer adjust the blend recipes based on current component availability as well as meet the specifications.
3. Controller controls the flow of blend components into the blender to

meet the recipe specification.

A refinery typically combines number of components into an in-line blender or directly into a blending tank (as modeled in this thesis) to produce various grades of gasoline. The most important consideration for a gasoline production system is to determine the optimum blend of raw materials to produce various grades of gasoline and thereby meet the property specifications of the end product. The combination of blend components,refereed to as blend recipes in this thesis, vary with demand for the gasoline as well as the availability of the components in that time period.

A refinery usually has a single blending unit. As a result of this it has to take care of optimally allocating the blender (blend scheduling) since there is a calibration time associated with the blending unit whenever there is task switch (a different gasoline is blended). Thus, minimizing the number of switches (from one grade to another) for the blender is another important consideration. In this thesis, a blender is modeled as a blending tank with a single virtual blender for each grade of gasoline and setup times are taken into consideration only when a particular grade of gasoline needs to be blended.

The properties of gasoline vary with the geographical location of the distribution area. For example, the property of Reid Vapor Pressure which determines the volatility of the gasoline blending system, is set based on the

average temperature of the location (Cold places can have a higher RVP in their gasoline). The value of octane number (the engine-knocking property) may vary from location-to-location as well.

Thus, a task of gasoline blending unit is to determine the amount of gasoline to blend at a given point in time (lot sizing), the proportion of each supplied component to use (blend recipes) and in what order should the various grades be blended (scheduling, resource allocation) over a given time horizon in order to meet the dynamically changing requirements. A system like gasoline blending is thereby characterized by many objectives.

1.3 History of Gasoline Blending

Mathematical Programming and hence Linear Programming gained a lot of popularity after Dantzing invented the Simplex algorithm in 1947 [5]. Gasoline Blending is often cited as an application of mathematical programming and considered as the triumph of applied mathematics (Operational Research). By late 1940, algorithms using this technique of Linear Programming were developed for the problems of scheduling, resource allocation, shipping and transportation and such alike. However, the lack of computational resources and commercial softwares, the application of this approach was highly limited and few problems could be solved on paper.[5]

With time and advances in computation power and learning through experience the refinery operations evolved itself. Non-linear properties were

converted to linear value with the introduction of blending indices in late 1950's[5]. Linear Programming, Sequential Linear Programming, gradient-based methods, dynamic programming, were subsequently researched and developed over the next decade as the availability of the commercial packages improved. By the end of this decade mathematical programming was used for financial forecasting, refinery planning, distribution planning in the refineries etc. Optimization approach was used in refinery-planning with a significant improvement over solving it on paper the matrices were to be input in column-by-column format. Report and matrix-generating tools hence came into picture even though the values in columns of the matrices had to be manually updated every time an LP was executed. The refineries soon realized that the problem was not the mathematics but maintaining the model and generating the reports. This resulted in the introduction of several in-house LP model-management projects like Amoco's MARS, Shell's AMBUSH, Exxon's PLATOFORM , Texaco's OMEGA etc [5].

Databases were incorporated in the decade of 1980 as more and more computational power became available[5]. By now we have variety of software products that take care of models ILOG, AspenTech.

1.4 Thesis Work

Currently in industry traditional optimization techniques (LP) are used for gasoline blending. This industrial approach gives a single solution for a

multi-time period blending problem. As mentioned previously the refineries need to calibrate the blending unit every time a task switch is made (also refereed as the set-up times).In this work, the model of gasoline blending was generated as a system of linear equations. The key assumption that enables generation of linear blending equations is that the volume of the blend (or, alternatively closing volume of the tank that receives the blended material from and in line blender) is known. The volume of the blend is a variable that is set by the evolutionary optimization algorithm. The optimum recipes were obtained for each time period for every gasoline grade by solving the LP for all time periods simultaneously.

Evolutionary optimization algorithm called differential evolution was used to determine the blend volumes for each gasoline grade in every time period. This determines the closing inventories for the linear model since the amount of product shipment from them is given.Each population member represents one multi time period solution to the blending problem. The blend recipes for each population member have been optimized via linear programming. Experiments were carried out on four time periods and fourteen time periods gasoline blending problem. Interesting results were found and they are discussed in chapter 4. This was later parallelized, using distributed computing API of Message Passing Interface-2 for performance gain.

In the thesis, mathematical modeling of the problem was done using cus-

tom software called as Auto-Equation Generator, AEG for short. AEG was designed and developed using Microsoft tools like Visual Studio 2008 for development, MS SQL as a Relational Database Management tool etc. The software generates system of linear equations from a user specified process system network topology. These equations were generated for every node and stream of the topology using pre-defined generic models from library of node models stored as tables in a Relational Database Management System. Every node instance of the network was attached to a single model from this library and is chosen by the user. The software interprets this generic custom model and generates large number of equations, variables and matrices before making them persistent for re-use. This information is communicated to solver in an ASCII format to solve the system. The usage of this software greatly improved the maintenance of large process system networks like heat exchanger network synthesis, multi-time gasoline blending etc. Making the variables and user input persistent not only eased the tracking of data input related error but also increased the productivity of those using heuristic-based algorithms. Figure 1.4 shows a simplified overview of the software while appendix A is dedicated for this software.

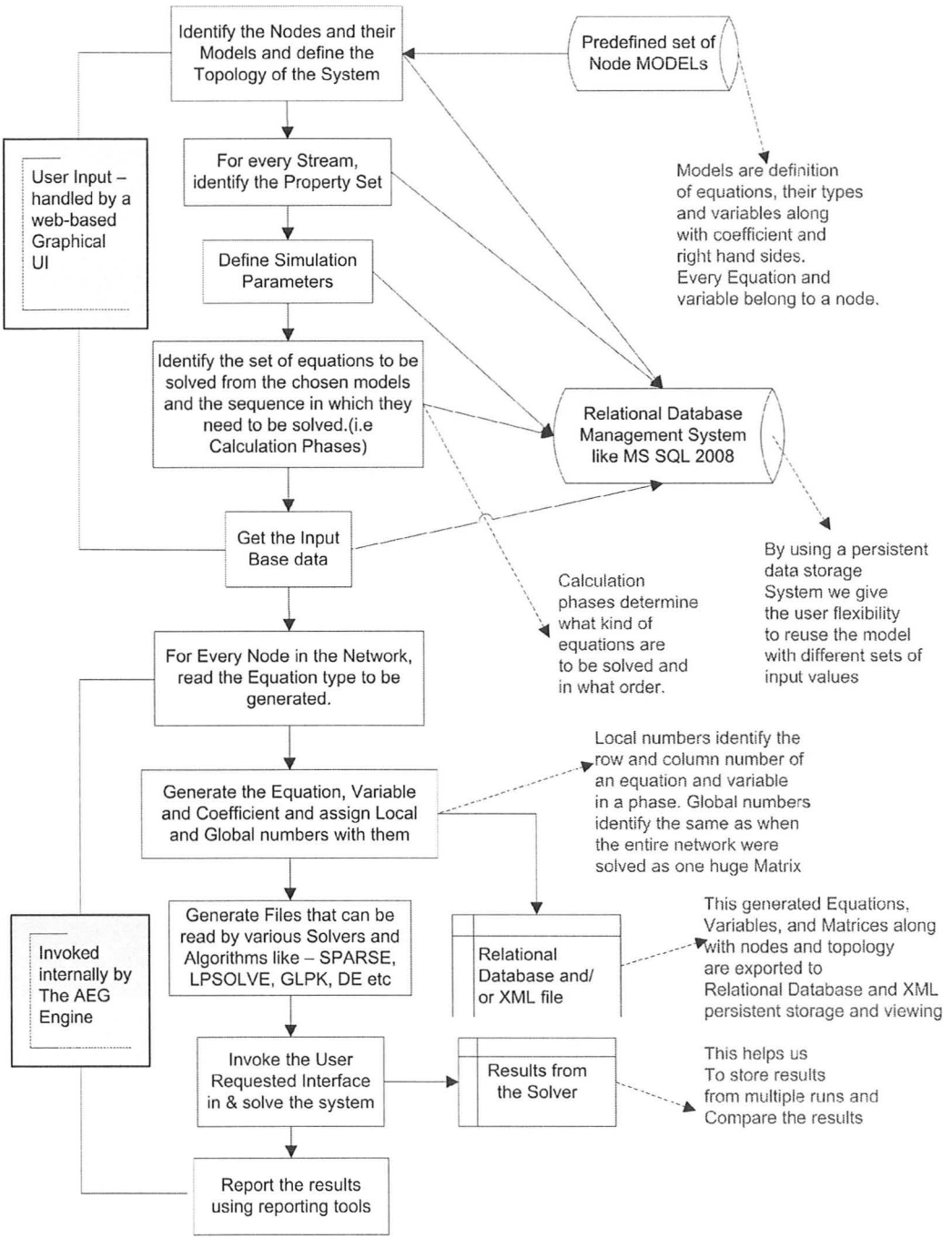

Figure 1.4: Summary of generating and solving process system mathematical models using AEG.

Chapter 2

Evolutionary Optimization

2.1 Introduction

Traditional optimization techniques like the linear programming, are either direct or gradient-based and rely heavily on the initial solution point[9]. The gradient-based approach requires the objective function to be unimodal (having a single minimum) and be at least one or two times differentiable [16]. If the objective function is singular or has large gradients the algorithm becomes numerically unstable[16]. On the other hand, the direct optimization method does not require the function to be derivable. From a randomly initialized point the search begins in a direction where the move the either accepted or rejected. However, both these approaches accommodate only unimodal functions, i.e functions with single minima or maxima since with the multi-modal functions, i.e. those with more than one optima, they pose

the challenge with a starting point. With these traditional methods the only way to treat conflicting objective functions is to reduce them to single objective function or treat them as constraints.

Multi-start methods which starts the process of optimization from different points were hence introduced. Each point serves as a sample point for a greedy, local optimization method. The local search could be derivative or direct-based. In this methods the number of starting points could not be effectively known since many could lead to the same minima.

Besides, these algorithms are difficult to adapt into a distributed computing infrastructure available to us.

Clearly such optimization techniques do not work efficiently when working on a multi-objective optimization problems like the gasoline blending, where objective functions may or may not be conflicting and there is more than one optimum solution.

2.2 Evolutionary Optimization

In contrast to gradient or direct optimization techniques discussed previously the evolutionary algorithms work with many different points like the multi-start methods and are iterative. However, unlike the multi-start methods where each member point is independent of other points, evolutionary algorithms evolve the solution by considering all the starting point solutions and try to gather the best of all the solutions. They are probabilistic,

population-based and "embarrassingly parallel". Evolutionary algorithms are based on Darwin theory of biological evolution and works on the principle of survival of the fittest. Thus, all the solution points are recombined, mutated and the best of them are chosen to survive in the next iteration. Evolutionary algorithms work with multi-modal, multi-dimensional, multi-objective optimization problems. Since every population member can be evaluated in isolation before they are combined and selected for next iteration they become easy to parallelize.

There are many approaches in this class of algorithms - Evolution strategies (developed by Rechenberg,1993 and Schwefel,1994), Genetic Algorithms (Holland,1962 and Goldberg,1989), Differential Evolution etc. among others. Even though all these techniques follow the Darwinian theory the difference lies in the way they encode parameters and with the algorithms used in recombination, mutation and selection . For example, evolutionary strategies use floating-point numbers to encode parameters while genetic algorithms use binary strings and hence are suited for combinatorial problem. Differential evolution works like genetic algorithms with parameters encoded with real numbers which makes it a numerical optimizer. For a problem like gasoline blending, which has many related or unrelated objective functions, and multiple optimizing parameters and optimal values, differential evolution seemed the best answer.

2.3 Differential Evolution

Differential Evolution (DE) is a global optimization algorithm that was originally proposed by Price and Storn in 1995[23]. DE has been found to be simple and highly effective in global optimization and hence has been widely studied and implemented on real world problems [23] [22] . They are found to be highly effective in global optimization of multi-modal multi-objective optimization problems[16]. The next section in this chapter talks about differential evolution algorithm in detail. Before that I will talk about the terms used in any evolutionary algorithm methodology.

1. Population - The set of randomly initialized vectors of optimizing parameters.
2. Generation - An iteration of crossover, combination and selection to choose the parents (new population members) of next iteration.
3. Vector - A one dimensional array of optimizing parameters that also qualifies as a single population member. It is analogous to biological Chromosome.
4. Mutation - It is analogous to biological mutation which maintains the diversity in the population members.
5. Crossover - Similar to biological crossover, this process combines two population members to produce a child population member to com-

pete against the parent.

6. Trial Vector - Resulting vector after perturbation and mutation.
7. Target Vector - The parent vector against which the trail vector competes to survive in the next generation.
8. Fitness Function - The value of the objective function for a population member.
9. Termination Criteria - The stopping criteria for the algorithm.

The differential evolution works with three distinct vectors from the population member pool to produce a trial population. It then performs crossover and mutation before selecting the one with better fitness function value. The pseudo code for differential evolution is in seen in algorithm 2.1.

The process of DE begins with initializing a population pool of N_p members. From the initial population, three mutually exclusive vectors are chosen that are not equal to the current target vector being evaluated. These vectors are scaled and mutated to form a trial vector. The trial vector is evaluated and is selected as the new parent if its fitness is no worse than the target vector. Though there are many variation of DE the one proposed as "Scheme DE1" by Rainer Storn[22] is as follows:

1. Initialize the Population members: DE treats all the variables as floating-point numbers even if they are discrete or integral. Every parameter of a population vector is randomly generated. For this

Algorithm 2.1: Pseudo Code for Differential Evolution.[16]

Input: N_p Number of population members;
C_r Crossover probability;
F Mutation Probability;
Output: Converged P
// P defines the population members
Initialize(P);
while *Not Converged* **do**
  foreach *Population member n in P* **do**
    repeat
      r_0 = rand(0,$N_p - 1$);
      r_1 = rand(0,$N_p - 1$);
      r_2 = rand(0,$N_p - 1$);
    until r_0,r_1,r_2 *and n are not mutually exclusive* ;
    /* r_0 is base vector;r_1 is the first vector and r_2 the second vector to calculate the difference vector, d */
    $d = P_{r_1}$ - P_{r_2};
    // v_n is the mutant vector
    $v_n = r_0 + F^*d$;
    // u_n is the trial vector
    u_n = crossover(P_n,v_n);
    if *fitness*(u_n) $\leq$ *fitness*(P_n) **then**
      /* The choice of u_n over P_n on equality, enables the DE to search for flatter surfaces. */
      $P_{n+1} = u_n$;
    else
      $P_{n+1} = P_n$;
    end
  end
end

to happen, upper and lower bounds, denoted by b_{upper} and b_{lower} are supplied for every parameter. Every parameter, j of each population member p is randomly (uniformly distributed) initialized as:

$$D_{j,p,initial} = rand_j(0,1) * (b_{upper} - b_{lower}) + b_{lower} \quad (2.1)$$

2. Mutation: As DE mutates and recombines the population members,

a trial population, N_p, made of trial vectors is generated. Mutation is carried out by scaling a difference vector and adding it to a base vector to produce a mutant vector. Three mutually exclusive vector indices: *first, second and base* are chosen, none of them equal to target vector index, p . The vectors at indices *first* and *second* are subtracted and scaled with a scaling factor, F. This weighted difference vector is added to the base vector to produce a mutant vector, v. This process is also termed as differential mutation.

$$v_{i,g} = D_{base,g} + F * (D_{first,g} - D_{second,g}) \tag{2.2}$$

3. Crossover: The user provides a crossover probability, C_r , which is used to recombine the mutant and target vectors to produce a trial vector, u. Every parameter of the trial vector u is chosen from either mutant or target vector. This process is also refereed to as discrete recombination. For every parameter, j, of trial vector, u

$$u_{j,p,g} = \begin{cases} v_{j,p,g} & rand_j(0,1) \prec C_r \\ D_{j,p,g} & \text{otherwise} \end{cases} \tag{2.3}$$

4. Selection: The vector with better fitness value is added to the trail population. If the trial vector has equal value of fitness function as the target vector, it is chosen in order to maintain diversity in the

population. This choice makes the DE explore flatter surfaces.

$$D_{p,g+1} = \begin{cases} u_{i,g} & \text{if fitness}(u_{i,g}) \leq \text{fitness}(D_{p,g}) \\ D_{j,p,g} & \text{otherwise} \end{cases} \tag{2.4}$$

As the new population is created, the process of mutation, crossover and selection is repeated until the convergence criteria (or termination criteria) is reached. The variations in DE are induced by how the base vector is chosen, number of weighted difference vectors that participate in generating the mutant vector, and the type of crossover used [16]. A variation is specified as *DE/base/n/crossover* where *base* indicates how the base vector is chosen; n represents the number of difference vectors that participate in mutation and *crossover*, the type of crossover used. Many variations of DE are researched, the most popular is the, *DE/rand/1/bin* where the base vector is randomly chosen with uniform binomial (*bin*) crossover and one difference vector participates to produce the mutant vector. The schematic diagram of DE algorithm is shown below in the figure2.1.

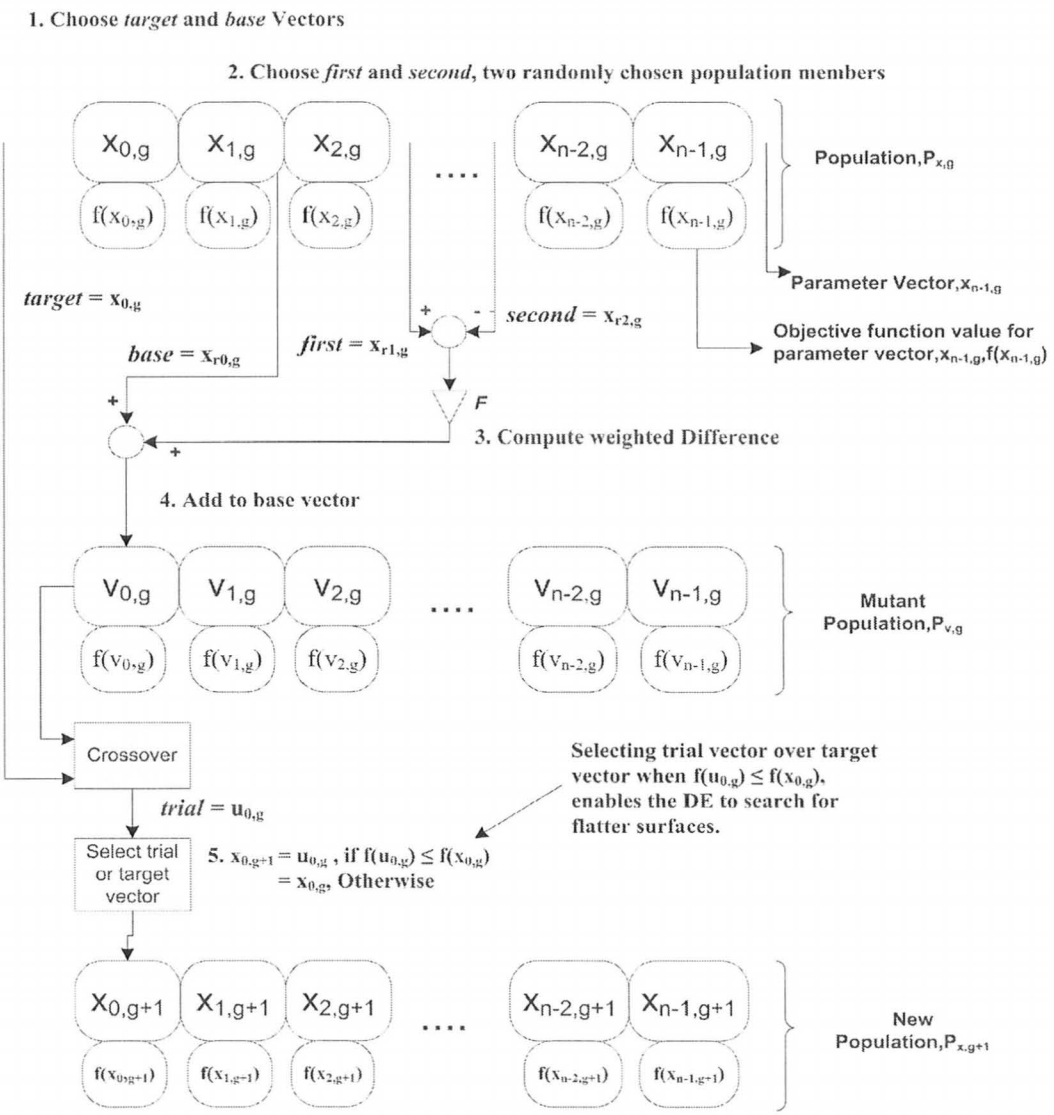

Figure 2.1: The Schematic Diagram for Differential Evolution (Adapted from [16]

2.4 DE: Multi-Time Period Gasoline Blending

In this thesis we used the DE code developed by Storn, Rainer [21]. The code provides methods to generate random numbers uniformly as well as various DE variations mentioned above. Any problem using DE needs to define their own Fitness Function calculation function called Energy(). The user also has to define the DE vector size - i.e the optimizing parameters. In the following section I will talk about how the DE optimizing parameters were defined as well as the algorithm to calculate the fitness function of population members.

2.4.1 Specifying the Optimizing Parameters

In order to make the formulation of the problem linear, the closing inventory for each grade of gasoline, g, in the blending tank at each time period, t, $V_{close,g,t}$, and the amount of each grade of gasoline blended, $V_{blend,g,t}$ were calculated by volumetric balance around each blend tank. These values were incorporated in multi-time period LP to solve.

The optimizing parameters for the DE, i.e the DE population vector definition, were defined as a vector of blend volumes for each gasoline grade in every time period assuming that each grade of gasoline was blended in each time period. The DE vector consisted of a blend probability, a number between 0 and 1, for each grade of gasoline in each time period. This num-

ber would identify if that particular grade of gasoline was blended or not as well as, if blended then in what amount. The DE vector was initialized to the size of total time periods multiplied-by total number of grades to blend. Thus, to blend a single grade of aviation gasoline for a time period of 4 days would make the DE vector of size 4. And, in order to blend three grades of gasoline over the time period of 13 days would make it of size 39. Each parameter was defined between a value of 0 and 1 - scaled value showing what fraction of maximum possible blend volume is to be blended for a given grade and the given time period. They were initialized randomly.

2.4.2 Known and Unknowns

Before proceeding to the algorithm of calculating the fitness function, the given and calculated data of the problem needs to be identified. Let,

G	Total number of gasoline grades to blend
T	Total Time periods.
g	Gasoline of grade g
dt	Length of each time period
t	Time period
F_{max}	Maximum flow rate for the blender
F_{min}	Minimum flow rate for the blender

V_{max}	Maximum volume of gasoline that the blender can blend
V_{min}	Minimum volume of gasoline that the blender can blend in time period t
$V_{close,g,t}$	The closing inventory of the blending tank with gasoline ,g, in time period t.
$V_{out,g,t}$	The demand for gasoline grade ,g, in time period t.
$V_{open,g,t}$	The opening volume of blending tank with gasoline grade ,g, in time period t.
$V_{blend,g,t}$	The blend amount of gasoline ,g, in time period t.
α	The threshold identifying whether we blend or not.
$s_{g,t}$	The setup time for the blending tank with gasoline,g, in time period t.
$V_{lost,t}$	The volume lost in time period t due to the recalibration time.
$V_{max,blend,t}$	The maximum volume that can be blended in time t.
$u_{g,t}$	An optimizing parameter in the DE trial vector, u, for gasoline grade ,g, in time period t. Randomly initialized

The user needs to specify total grades of gasoline to blend, G; the length of each time period, dt; total time periods, T; F_{max} and F_{min} for the blending tank in each time period; the demand for each grade of gasoline g in each time period t,$V_{out,g,t}$; opening inventory for each grade of gasoline $V_{open,g,t}$

for t =1; as well as the re calibration or set up time, $S_{g,t}$ .V_{max} and V_{min} are derived from F_{max} and F_{min} respectively. The $V_{blend,g,t}$ is calculated by the fitness function algorithm, so is $V_{close,g,t}$. The $V_{open,g,t}$ is specified only for the first time period for all grades of gasoline and calculated for the subsequent time periods. The maximum and minimum blending flow rates, F_{max} and F_{min}, of the blending tank were known and the parameter in the DE vector, $u_{g,t}$, was interpreted using a threshold value,α. The value of the threshold, α, is calculated as:

$$\alpha = F_{min} \div F_{max} \tag{2.5}$$

If the value in the DE parameter, $u_{g,t}$ is less than α, then the g grade of gasoline is not blended otherwise it is. If we blend this gasoline, the re calibration time was calculated (discussed next) as well the amount of gasoline g to blend, $V_{blend,g,t}$.

$$V_{blend,g,t} = F_{max} \times dt \times u_{g,t} \; ; \forall g, t \tag{2.6}$$

Using this, the closing inventory of the tank is calculated as:

$$V_{close,g,t} = V_{open,g,t} + V_{blend,g,t} - V_{out,g,t} \; ; \forall g, t \tag{2.7}$$

And, the opening inventory of the tank in the next time period is calculated as:

$$V_{open,g,t+1} = V_{close,g,t} \forall g, t \tag{2.8}$$

The detailed algorithm for calculating the energy or the value of fitness function is discussed in next section.

2.4.3 Calibration or Setup Time

As discussed in the introductory chapter, refineries typically have a single blending unit. However, this thesis solves the problem by creating 3 virtual blenders for each grade of gasoline - regular, mid-grade and premium. Every time a grade of gasoline is blended, the blending unit needs to be re calibrated for the next blend. This is usually termed as task switch. The algorithm considers this re calibration time as the time when no gasoline is blended and hence the volume of gasoline that the blender can blend in that time is lost. As a result, the maximum volume that the blender can blend is reduced due to the task switch. Since the DE optimizing parameters (trial vector) assumes that we blend each grade of gasoline per time period, the blender needs to be re calibrated every time we blend a grade. This is calculated as the volume lost, $V_{lost,t}$,which is calculated as:

$$V_{lost,t} = V_{lost,t} + F_{max} * S_{g,t} \forall t \tag{2.9}$$

The value of $V_{lost,t}$ is reset to 0 for each time period and accumulated only when we blend a grade of gasoline.

Algorithm

As seen in the algorithm 2.2, the fitness function calculates the blending amount for each grade of gasoline as well as the closing and opening inventories for each time period. This information is shared in an array with the linear solver. The solver then solves the system of linear equations and returns the value of the objective function (discussed in next chapter), which is also the fitness value of the trial vector. After all the population members are evaluated, the mutation and crossover for each member is performed and new trial population is generated.

Algorithm 2.2: Algorithm to calculate fitness function value for a trial population member for MTP Gasoline Blending

Input: G Total gasoline grade;
T Total time periods;
F_{max} Maximum inflow the blender can blend;
F_{min} Minimum inflow blender can blend;
$s_{g,t}$ Setup time required to switch to grade g in time t;
u trial vector;
dt Length of time period t;
Output: $V_{blend,g,t}$ Volume of gasoline g to blend in time t;
$V_{close,g,t}$ Closing volume in blending tank g in time t;
$V_{open,g,t}$ Opening volume in blending tank g in time t;
foreach *Time period t in T* **do**
  $V_{atmost} = V_{max}$;
  $V_{lost} = 0.0$;
  $sum = 0.0$;
  foreach *Gasoline Grade g in G* **do**
    $\alpha = F_{max} \div F_{min}$;
    if $u_{g,t} \leq \alpha$ **then**
      $u_{g,t} = 0.0$;
    else
      $V_{lost} = V_{lost} + F_{max}{*}dt$ - $F_{max}{*}s_{g,t}$;
    end
    $sum = sum + u_{g,t}$;
  end
  $V_{atmost} = V_{max}$ - V_{lost};
  foreach *Gasoline Grade g in G* **do**
    if $sum \succ 1.0$ **then**
      $V_{blend,g,t} = (u_{g,t} \div sum){*}V_{atmost}$;
    else
      $V_{blend,g,t} = u_{g,t}{*}V_{atmost}$;
    end
    if *t is last time period, T* **then**
      $V_{close,g,t} = 0.0$;
      $V_{blend,g,t} = V_{out,g,t}$ - $V_{open,g,t}$;
    else
      $V_{close,g,t} = V_{blend,g,t} + V_{open,g,t}$ - $V_{out,g,t}$;
    end
    if $t+1 \prec T$ **then**
      $V_{open,g,t+1} = V_{close,g,t}$;
    Update X_{big};
  end
end
Invoke Solver to solve;

2.5 Mutation and Crossover Factors

DE involves 3 main parameters - C_r (crossover factor), values of N_p (number of population members) and F(scaling factor). DE is affected by all these three parameters and hence performance of it varies. There is always a trade-off between the rate of convergence and robustness of an algorithm. These factors are hugely affected by the choice of scaling factor and the crossover rate. The higher is C_r, the faster is the convergence but objective function becomes less robust[21]. DE is very less affected by the value of C_r [21]. It acts more like a fine tuning parameter [21]. DE is highly affected by changes in F. But in general, large F favors global exploration, while small F favors local exploitation [8]. C_r determines the number of new components that can be introduced in the next generation. The proposed initial values of C_r and F are $F \in [0.5, 1]$, $C_r \in [0.8, 1]$ and $N_p = 10$ nDim where nDim is the dimensionality of the problem (number of optimizing parameters)[24]. The values of C_r and F are limited to [0, 1] [21][22][16]. Studies have shown classical DE, *DE/rand/1/bin* to have been highly successful where parameter dependence is small[16]. When the value of N_p is increased, values of C_r and F should also be increased [21].

Chapter 3

Multi-Time Period Gasoline Blending

3.1 Understanding the Problem

As said in the introductory chapter, the problem of gasoline blending is characterized by varying demand patterns. The gasoline requirement changes from time to time and is seasonal. As mentioned previously various factors affecting the production of gasoline are: availability of the raw components, demand for the gasoline, the recipes of various grade etc. The production needs to consider the availability of the raw components at any given time in order to determine the blend recipe. The job of the blend scheduler is to determine which product to blend at a given point time and in what quantity in order to meet the demand while minimizing the overall

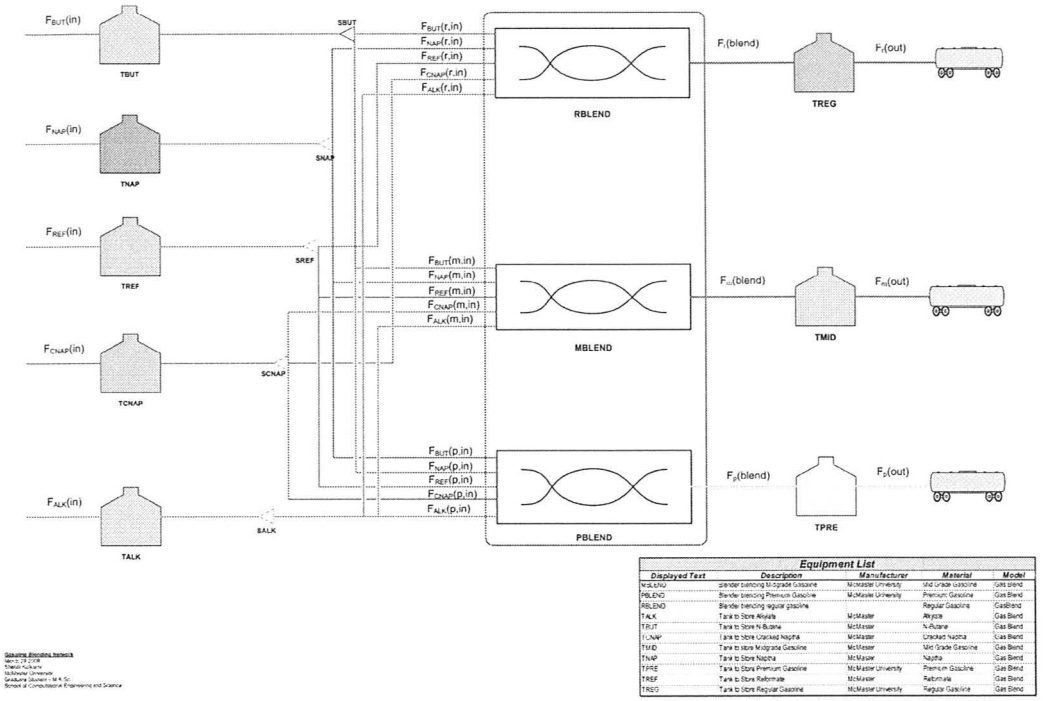

Figure 3.1: Gasoline Blending Problem

production cost among other considerations. The blender needs to be calibrated every time a different grade of gasoline is blended. Hence minimum number of switches must to made to avoid this overhead and larger batches of single grade with very little recipe variation should be targeted. The setup times are considered and discussed in the previous chapters.

The gasoline blending problem is mostly solved a non-linear problem. However, in the thesis we have been able to solve it as a linear problem thanks to choosing V_{blend} as a variable whose value is set in an outer optimization loop via differential evolution. The figure 3.1 shows the gasoline blending production system. A typical production system will have at least

following component: alkylate, light and heavy naphtha, reformate, butane, hydrocarbons, cracked naphtha. Seven physical properties are used to describe every raw material and its stream as well as for each gasoline grade and its stream. The formulation of the linear system of equations is based on determining how much amount of each gasoline grade and in what quantity. The demand or the lifting amount of gasoline from each of the blending tanks at the end of a time period is known and hence the closing inventories in all the blending tanks at the end of each time period can be calculated. These calculations are handled by the equations discussed in the DE chapter. The tanks are modeled as multiple outlet tanks to eliminate the splitters in the image.

3.2 Problem Formulation

The problem of gasoline blending was formulated in two ways: (a) First model blends the gasoline by summing up the volumes of the incoming blend components in the blending tank. (b) The second model calculates the contribution of each blend component as a fraction of total amount of gasoline blended.

3.2.1 Assumptions

While formulating the problem it was assumed that the demand for each gasoline grade in every period is known, so was the supply for blend components. The opening inventories for each of the blend component tank, as

well as the blending tanks were known. It is assumed that the supply of components in every time period had the same chemical composition as the previous and the current blend component, making them constant throughout the system. The tanks are constrained by their volumetric capacities. The specification of the chemical composition on every gasoline grade was given and they formed the constraints of the LP.

3.2.2 System of Linear Equations

As mentioned previously gasoline blending problem is mainly solved as a non-linear problem. We formulated the problem as a linear problem by calculating the closing inventory in the blending tanks. The formulation of the problem was done in two different ways which will be discussed in more depth in the following sections.

Volumetric Formulation

The volumetric formulation consisted of defining the volumetric contribution of every incoming blend component in the total blend of each gasoline grade in every time period to determine the blending recipe. The blending inlets were summed up to the amount of gasoline blended as its volumetric contribution in the total blend. The objective function consisted of minimizing the global cost over the entire time horizon.

Minimize:

$$\sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{g=1}^{G} V_{in,k,g,t} \times C_k \tag{3.1}$$

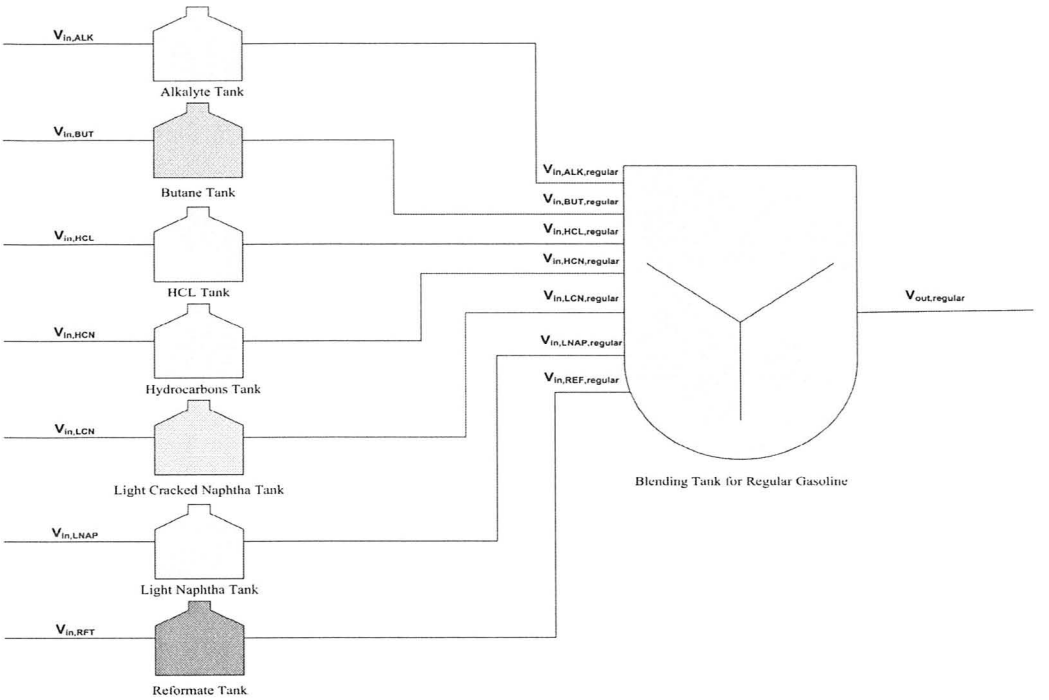

Figure 3.2: Volumetric Formulation for Regular Gasoline

where,

T	Total time periods
G	Total grades of gasoline
K	Total components
$V_{in,k,g,t}$	Incoming volume in blender g, from component k in time t (i.e. the volumetric contribution of component k in time t to blend gasoline grade g)
C_k	Unit cost of component k
g	Gasoline grade to blend. Can be *reg* (Regular),*mid* (Mid grade)or *prm* (Premium).

The physical composition of every gasoline grade was calculated based on quality balance equations around the blending tanks. The value of physical property, i, at the end of any time period for each gasoline grade is calculated by summing up the physical property in the volume of incoming flow from all the blend components into the blending tanks; the value of quality, i, in the opening volume of gasoline in the blending tanks in time t and subtracting the property of the gasoline that leaves the tank in time t. In the first time period $Q_{open,i,g,t}$ is known, and hence is the coefficient in equations 3.2; however, in the rest of the time periods $V_{open,g,t}$ is known and is the coefficient in the same equations.

$$\begin{aligned} & V_{open,g,t} \times Q_{open,i,g,t} - (F_{out,g,t} \times dt + V_{close,g,t}) \times Q_{i,g,t} \\ & + \sum_{k=1}^{K} (V_{in,k,g,t} \times Q_{i,k,t}) = 0 \; ; \forall \; i, g, t \end{aligned} \tag{3.2}$$

where,

$V_{open,g,t}$	Opening Volume in time t for blender g
$Q_{open,i,g,t}$	Opening quality value of quality i for gasoline g in time t
$F_{out,g,t}$	Outgoing flow in time t from blend g, the demand.
$V_{close,g,t}$	Closing inventory in time t for grade g of gasoline
$Q_{i,g,t}$	Calculated quality i for gasoline g in time t
$F_{in,k,t}$	Incoming flow for component k in time t, the supply.
$Q_{i,k,t}$	Quality i for component k in time t, assumed same for all t
dt	Length of time period t.

The volume in each blending tank is calculated with the volumetric balance equation:

$$\sum_{k=1}^{K} V_{in,k,g,t} \; - \; V_{open,g,t} + V_{close,g,t} - V_{out,g,t} = 0;; \forall g,t \tag{3.3}$$

The closing inventory of the blend component tanks is given by the following volumetric balance equations.

$$F_{in,k,t} \times dt - \; V_{close,k,t} \; + \; V_{open,k,t} \; - \; \sum_{g=1}^{G} (F_{out,k,t,g} * dt) = 0 \; ; \forall k,t \tag{3.4}$$

where,

$V_{close,k,t}$ Incoming flow from component k in blender g in time t.

$V_{open,k,t}$ Opening Volume in the component tank k in time t.

$F_{out,k,t,g}$ Outgoing flow from tank k into blending tank g.

The closing inventory in the component and blending tanks at the end of time period, t, must be within the volumetric capacity of the tank.

$$V_{min} \leq \; V_{close,k,t} \; \leq \; V_{max} \; ; \; \forall k,t \tag{3.5}$$

$$V_{min} \leq \; V_{close,g,t} \; \leq \; V_{max} \; ; \; \forall g,t \tag{3.6}$$

where,

V_{min} Minimum volume that the tank can hold, usually 0

V_{max} Maximum volume the tank can hold

To ensure that each gasoline grade maintains its physical composition calculated in equation 3.2, the value of the properties are governed by the following constraints:

$$Q_{i,min} \leq Q_{i,g,t} \leq Q_{i,max} \; ; \; \forall i, g, t \tag{3.7}$$

where,

$Q_{i,min}$ Minimum quality specification for quality i

$Q_{i,max}$ Maximum quality specification for quality i

Fractional Formulation

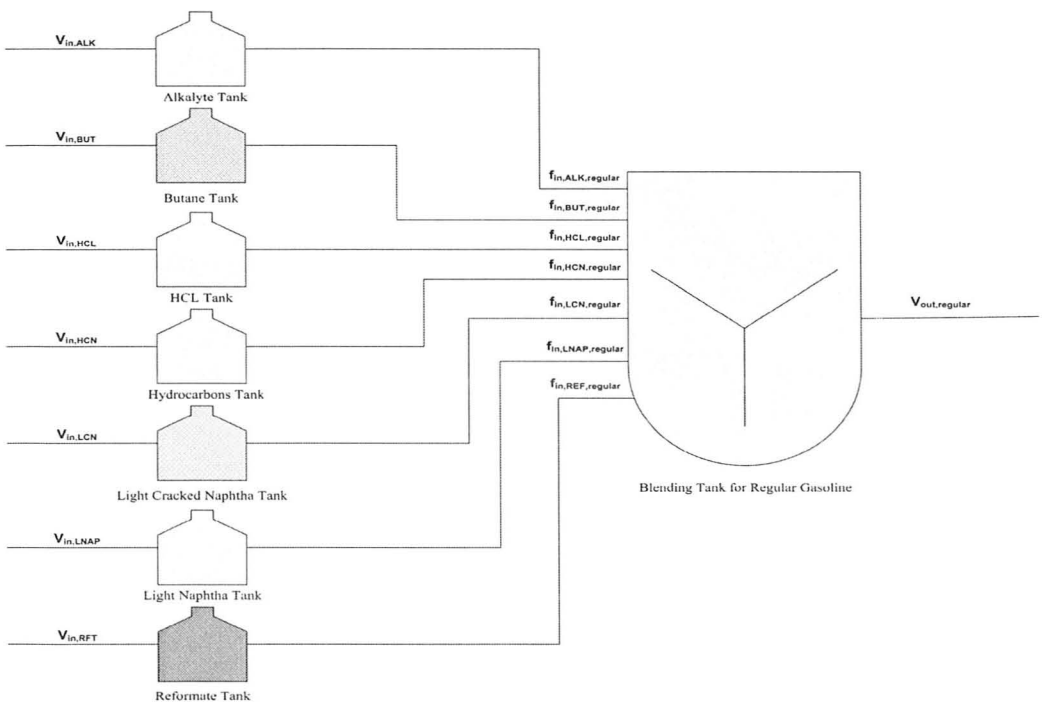

Figure 3.3: Fractional Formulation for Regular Gasoline

Unlike the volumetric formulation the fractional formulation calculates the contribution of each blend component as a fraction of total gasoline to blend. The actual volume of the component in the blended gasoline can be calculated by equation 3.8.

$$V_{in,k,g,t} = V_{blend,g,t} \times f_{k,g,t} \forall k, t \tag{3.8}$$

where,

$V_{blend,g,t}$ Amount of gasoline g to blend in time t.

$f_{k,g,t}$ Fraction of component k used to produce gasoline g in time t.

The objective function of this formulation is defined as:

Minimize:

$$\sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{g=1}^{G} (V_{blend,g,t} \times f_{k,g,t} \times C_k) \tag{3.9}$$

The physical properties of the gasoline in the blending tanks are handled by the following equation (Quality Balance Equations):

$$\begin{aligned} &(V_{open,g,t} \times Q_{open,i,g,t}) + \sum_{k=1}^{K} (V_{blend,g,t} \times f_{k,g,t} \times Q_{i,k,t}) \\ &- (V_{out,g,t} \times Q_{i,g,t}) - (V_{close,g,t} \times Q_{i,g,t}) = 0 \; ; \forall i, g \end{aligned} \tag{3.10}$$

The closing volume of the blend component tank in every time period t is given by:

$$V_{close,k,t} = F_{in,k,t} \times dt + V_{open,k,t} - \sum_{g=1}^{G} V_{out,k,g,t} = 0 \; ; \forall k, t \tag{3.11}$$

where,

$V_{out,k,g,t}$ Volume of component k used to produce gasoline $g$to blend in time t.

The sum of fractions of each component's contribution to the total blend must be equal to unity.

$$\sum_{k=1}^{K} f_{k,g,t} = 1.0 \; ; \forall g, t \tag{3.12}$$

The bounds on the closing volume of the blending and component tank as well the specification of physical properties are bounded by equation as in the previous formulation equations 3.6, 3.5 and 3.7. Additionally the bounds on the fractions are specified as (i.e the fraction of component k used to blend $V_{blend,g,t}$ of gasoline g in time t):

$$0.0 \leq f_{k,g,t} \leq 1.0 \; ; \forall k, g, t \tag{3.13}$$

3.2.3 Multi-Time Period Connection Equations

Information from the current time period is passed to the next time period by the multi-time period connectivity equations.

The component tanks are connected by their inventories. i.e the closing inventory of a time period is the opening inventory for the next time period.

$$V_{open,k,t} - V_{close,k,t-1} = 0 \; ; \; \forall k, t(t = 2 \; to \; T) \tag{3.14}$$

The blending tank inventories are similarly connected like above:

$$V_{open,g,t} - V_{close,g,t-1} = 0 \; ; \forall j, t(t = 2 \; to \; T) \tag{3.15}$$

The chemical composition of the gasoline at the end of every time period in the blending tank is the opening composition of the gasoline (equations 3.2and 3.10) in the next time period. They are equated by:

$$Q_{open,i,g,t} - Q_{i,g,t-1} = 0 \; ; \forall t(t = 2 \; to \; T), g, i \tag{3.16}$$

3.3 Chemical Properties

The chemical composition of the blend components and the gasoline is defined by the following set of properties:

- Aromatic (ARO): The aromatic content in the gasoline affects its ignition quality. Aromatic material usually has high octane content and is desirable in the gasoline end products.
- Benzene (BEN): Benzene as an additive in gasoline increases the octane rating and thereby reduces knocking. However, due to environmental restrictions the its usage is restricted.
- Octane Ratings: In an internal combustion engine the ignition occurs due to combustion of air-fuel mixture. When the entire mixture of fuel and air in the combustion chamber is not burnt properly, it undergoes spontaneous combustion. This is called detonation which occurs after the normal combustion is initiated. The octane rating determines the resistance of gasoline against this detonation. Thus, the octane rating of the fuel reflects the ability of the unburned end gases to

resist spontaneous auto ignition under the engine test conditions used [11]. There are two standards to test this resistance. They are called as the Motor and Research Octane Numbers, referred as MON and RON respectively. The difference in these two ratings for a chemical component lies in the reference against which they are compared and the conditions in which they are tested. Usually, MON ratings are obtained under more stressed conditions (severe, sustained high speed, high load driving) than RON (typical mild driving, without consistent heavy loads on the engine) which gives a lesser numerical value for MON than RON [11]. The average of RON and MON, also called as the "anti knock index", $\frac{(RON+MON)}{2}$ is the standard octane number seen on the gas stations and the single most important criteria for the motorists.

- Reid Vapor Index (RVI): This property defines the volatility of the gasoline. The required volatility depends on the average temperature of the market area. In cold climates lower volatility can cause failure in ignition hence usually higher values of RVI are desired. In hot climates higher number of this property could result into vapor lock where the liquid fuel changes in to gaseous state making the engine starve for the fuel and hence lower values of RVI might be desired.
- Sulfur (SUL): Various streams that enter into the gasoline blending

section of a crude oil distillatory contain sulfur to some extent. Due to environmental concerns about sulfur emission, its content is desirable to as low as possible value.

3.4 Blending Components

The main blend components used for this problem are described briefly.

- Light Naphtha (NAP): Naphtha is a chain of pentane and hexane. Processing of this intermediate product gives light naphtha that is used in gasoline for its high octane value.
- Reformate (REF): Reformate is a high octane rating liquid product obtained through the process of catalytic reforming of low-octane rating naphtha.
- Alkylate (ALK): Alkylates are the most expensive of all components are highly desirable for their exceptional anti-knocking properties. They are obtained through the process of alkylation which is an important (and economically expensive) refinery process. The octane rating of alkylates depend on the alkenes used in the process and also on the operating conditions.
- Light cracked Naphtha (LCN): Light cracked naphtha is obtained by process of fluid catalytic cracking where heavy molecules are converted into light weight molecules.

Chapter 4

Numerical Results

4.1 4-Time Period Optimization

This small size problem has been used to explore various facets of multi-time period gasoline blending and to illustrate various aspects of the solution method.

Solving the LP requires the knowledge of physical properties of the components to blend as well as the specifications of the same for the gasoline. The user also provides the available inventory of components in tanks and gasoline in the blending tanks in the beginning of the time horizon over which the problem is solved. Next couple of pages will show the data that was used to solve this small 4-time period, 3-grade gasoline blending problem.

The charts in figure 4.1 illustrate the varying demand and supply pattern

with time for different grades of gasoline and the 7 blend components respectively. The unit cost of blending components used in objective function calculation is shown in table 4.4.

Table 4.3 lists the opening inventory of the material in the component and blending tanks. The physical properties of the material in the component tanks (which are assumed to remain same throughout) and those of gasoline in the blending tanks at the beginning of time horizon is shown in table 4.6 and 4.5 respectively. The specifications (constraints in equations 3.7) of these physical properties for all gasoline grades appears in table 4.7.

	Time Period			
Grade	1	2	3	4
Regular	9000	5000	3000	9000
Midgrade	6000	7000	8000	8000
Premium	7000	9000	5000	4000

Table 4.1: Demand in *Bbl* for various grades of Gasoline in various time periods

	Time Period			
Component	1	2	3	4
Alkylate	5000	4000	3000	5000
Butane	4000	8000	3000	3200
HCL	5600	6000	7000	4000
Hydrocarbons	3000	4000	7000	7000
Light Cracked Naphtha	3000	4300	3200	5000
Light Naphtha	3000	3000	8000	6300
Reformate	5000	9000	6000	3000

Table 4.2: Supply in *Bbl* for Components in different time periods

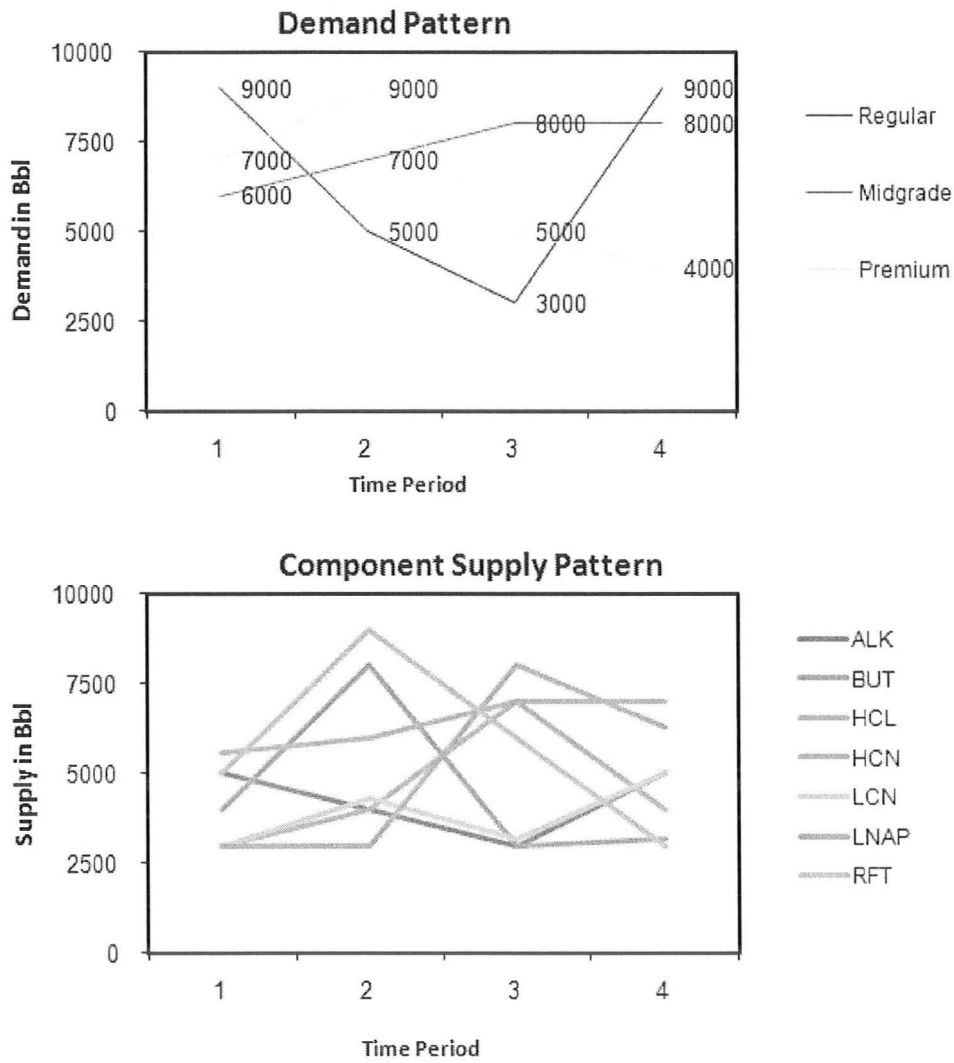

Figure 4.1: The first chart shows the demand, $V_{out,g,t}$ in Bbl per time period for each blending tank. The second chart shows the supply, $V_{in,k,t}$ in Bbl per time period for each blending component tank.

Material	Opening Volume
Alkylate	500
Butane	400
HCL	160
Hydrocarbons	350
Light Cracked Naphtha	700
Light Naphtha	200
Reformate	500
Regular Gasoline	7000
Midgrade Gasoline	6000
Premium Gasoline	6000

Table 4.3: Opening Volume in *Bbl* in various tanks

Component	Unit Cost
Alkylate	29.20
Butane	11.50
HCL	20.00
Hydrocarbons	22.00
Light Cracked Naphtha	25.00
Light Naphtha	19.70
Reformate	24.50

Table 4.4: Unit Cost in USD per *Bbl* for components

	Gasoline		
Property	Regular	Midgrade	Premium
Aromatics	21.061	21.061	14.569
Benzene	3.821	3.821	3.032
MON	85.00	86.00	87.00
Olefins	13.00	16.00	10.00
RON	92.00	94.00	97.00
RVI	5.458	13.00	15.60
SPG	0.750	0.80	0.80
Sulfur	0.008	0.005	0.005

Table 4.5: Physical Property of the gasoline in the blending tanks at the beginning of time horizon. They are recalculated for every time period as new gasoline is blended into the tank.

	Components						
Property	ALK	BUT	HCL	HCN	LCN	LNAP	REF
Aromatics	0.0	0.0	0.0	25.0	18.0	2.97	74.90
Benzene	0.0	0.0	0.0	0.50	1.00	0.59	7.50
MON	93.70	90.00	79.80	75.80	81.60	81.60	90.80
Olefins	0.00	0.00	0.00	14.00	27.00	0.00	0.00
RON	95.00	93.80	82.30	86.70	93.20	67.80	103.00
RVI	5.15	138.00	22.33	2.38	13.88	19.90	3.620
SPG	0.703	0.584	0.695	0.791	0.744	0.677	0.818
Sulfur	0.000	0.000	0.000	0.490	0.080	0.010	0.000

Table 4.6: Physical Property of the Components, assumed constant through out

	Regular		Midgrade		Premium	
Property	Min	Max	Min	Max	Min	Max
Aromatic	0.00	60.00	0.00	60.00	0.00	60.00
Benzene	0.00	5.90	0.00	5.90	0.00	5.90
MON	82.00	100.00	84.00	100.00	86.00	100.00
Olefin	0.00	24.20	0.00	24.20	0.00	24.20
RON	92.00	100.00	94.00	100.00	96.00	100.00
RVI	0.00	15.60	0.00	15.60	0.00	15.60
SPG	0.73	0.81	0.73	0.81	0.73	0.81
Sulfur	0.00	0.01	0.00	0.01	0.00	0.01

Table 4.7: Constraints on the Physical Properties for Gasoline

4.2 4-Time Period Results Discussion

Unlike the traditional direct optimization techniques that give a single point solution, differential evolution gave multiple solutions with same optimum value. The DE algorithm discussed in Chapter 2, determined the amount of each gasoline grade to blend in every time period, and there by calculated the closing inventory as the lifting amount for each time period was known. As discussed in Chapter 2, DE selects a trial vector when it's fitness is no worse than the target vector. Hence, trial population members with cost value within the 1% of the optimum cost were saved for analysis purpose. Multiple feasible solutions of such kind were obtained in every generation, and can be seen in the chart 4.2.

4.2.1 Gasoline Recipes

Eight solutions points from the chart 4.2 were randomly chosen with the optimum cost value and are discussed over the next few pages.

The solutions obtained had population members with same objective function value but in every solution different amount of each gasoline grade was blended in every time period. As a result,the proportion of blend components in every solution was different. Hence, blend recipes for each gasoline grade in each time period was different for different solution points. This can be seen in figures 4.5, 4.6 and 4.4 for selected eight solution members. Common indices combined to gather represent a single solution point. Each

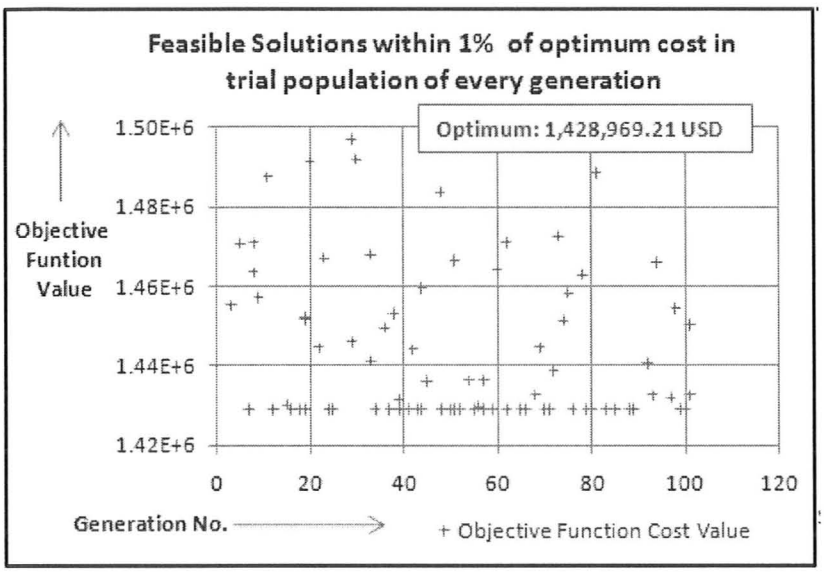

Figure 4.2: Each solution point is a feasible trial population member in various generations, whose fitness value is within 1% of the optimum fitness value.

index shows the % of each blend component used, the amount of gasoline blended and the closing inventory in the blending tank at the end of every time period. For example, consider index 1. The solution blends 4881.9 *Bbl* mid grade gasoline (figure 4.5), 6804.5 *Bbl* premium gasoline (figure 4.6) and 5313.7 *Bbl* regular gasoline (figure 4.4). In figure 4.4 in column ALK this solution uses 42.6% of Alkylate to blend 5313.7 *Bbl* regular gasoline. In the same a different solution, say indexed 3, uses 8.7% Alkylate to blend 7303.3 *Bbl* of regular gasoline. Similar such differences can be pointed out in these figures for each gasoline grade in every time period.

Since the amount blended varied in every population member, the closing inventory in each blending tank was different. This fact is illustrated in

charts of figure 4.3.

All the eight illustrated solution members have the same objective function cost value: $1,428,969USD$.

4.2.2 Mutation and Crossover Factor

Generations with smaller population sizes (10) were studied to understand the effect of crossover, mutation and strategy on the algorithm's convergence. For exponential strategy, the entire population converged in less than 25 generations when the mutation probability was fixed at 0.8 and the crossover probability was varied in the range [0.7,0.95]. The rate of convergence for each population members using an exponential strategy is shown in figure 4.7.As can be seen the population members converged at a slightly better rate as crossover was increased from 0.7 to 0.95. Similar observations can be made for the binomial strategy in figure 4.8 with members converging at a better rate with increase in the crossover probability.

However, the two strategies differed in that the population members converged in lesser number of iterations (≤ 25) for an exponential strategy of crossover than the binomial strategy (≤ 50) of the same.

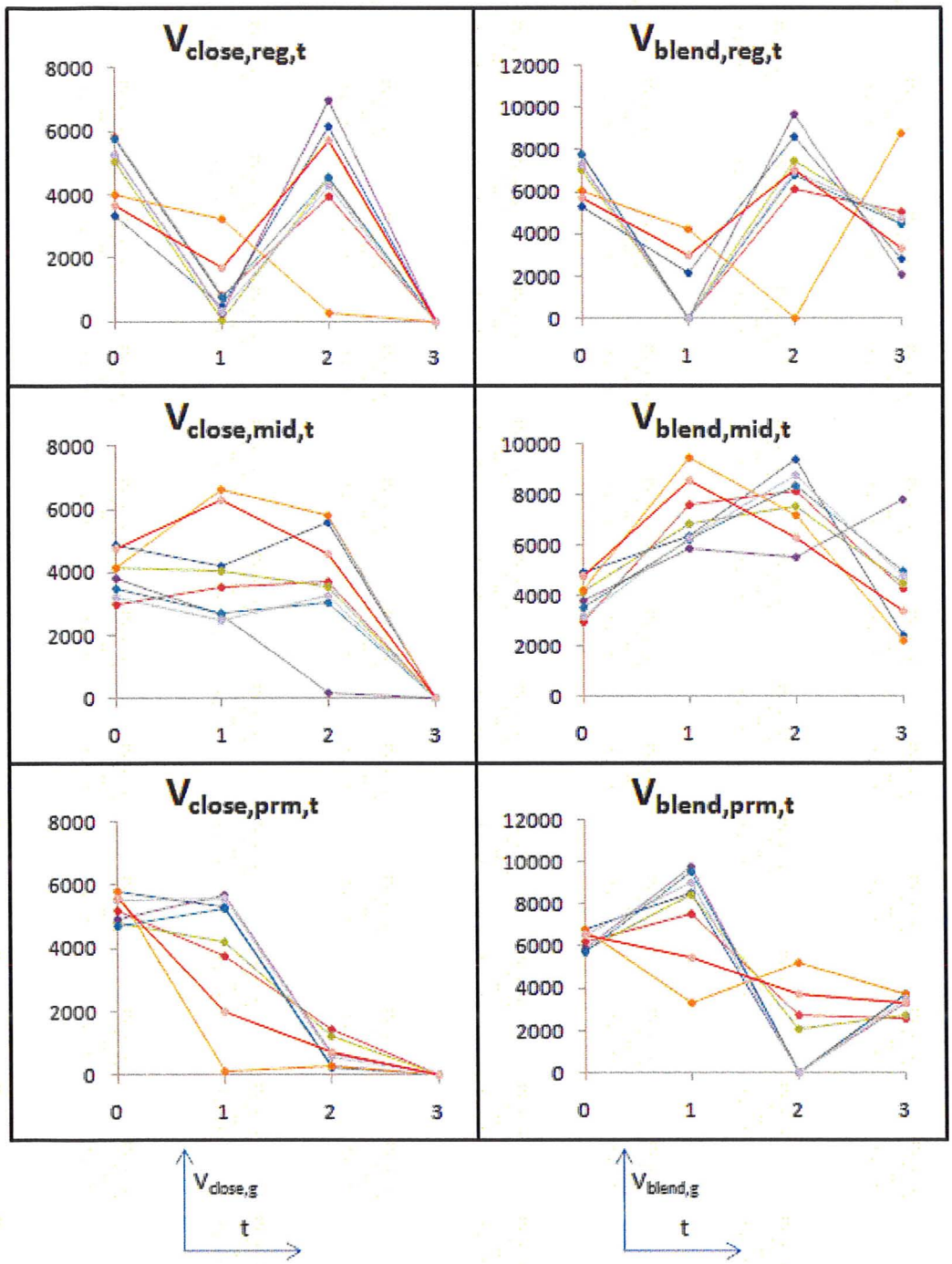

Figure 4.3: Plots illustrating varying amount of closing inventories as well as the amount of gasoline blended in the blending tanks for each gasoline grade in each time period, for different solutions

Multiple Regular Recipes										
Index	Time Period	ALK	BUT	HCL	HCN	LCN	LNAP	RFT	$V_{blend,reg,t}$	$V_{close,reg,t}$
1	0	42.6	15.5	15.1	4.6	0.0	6.5	15.6	5313.7	3313.7
	1	0.0	1.9	50.7	2.0	0.0	0.0	45.4	2171.7	485.3
	2	0.0	1.9	50.7	2.0	0.0	0.0	45.4	8648.5	6133.8
	3	39.9	3.7	34.5	2.0	0.0	0.0	19.9	2866.2	0.0
2	0	21.1	10.3	37.4	3.9	0.0	0.0	27.4	7791.0	5791.0
	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	791.0
	2	0.0	2.9	36.5	1.9	0.0	8.1	50.6	6137.6	3928.6
	3	9.8	2.2	45.8	1.6	2.8	0.0	37.8	5071.4	0.0
3	0	8.7	10.3	39.0	2.6	8.8	0.0	30.6	7030.3	5030.3
	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.3
	2	0.0	1.9	50.7	2.0	0.0	0.0	45.4	7454.5	4484.9
	3	0.0	1.9	50.7	2.0	0.0	0.0	45.4	4515.1	0.0
4	0	8.5	10.3	42.2	4.0	0.0	0.0	35.1	7266.1	5266.1
	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	266.1
	2	0.0	1.9	50.7	2.0	0.0	0.0	45.4	9684.9	6951.0
	3	29.0	2.8	34.8	0.0	12.5	0.0	20.9	2049.0	0.0
5	0	32.6	10.7	31.2	3.2	4.4	0.0	17.9	7757.4	5757.4
	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	757.4
	2	27.5	3.1	39.5	2.0	0.0	0.0	27.8	6791.4	4548.8
	3	23.8	2.6	36.9	0.0	12.5	0.0	24.2	4451.2	0.0
6	0	42.1	14.4	12.6	4.0	1.2	8.3	17.4	6012.9	4012.9
	1	1.5	1.9	50.1	2.0	0.0	0.0	44.4	4237.9	3250.8
	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	250.8
	3	0.0	1.9	50.7	2.0	0.0	0.0	45.4	8749.2	0.0
7	0	0.0	9.1	38.0	0.3	22.9	0.0	29.7	7281.3	5281.3
	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	281.3
	2	0.0	2.2	46.4	2.0	0.0	2.5	47.0	6972.9	4254.2
	3	0.0	1.9	50.7	2.0	0.0	0.0	45.4	4745.8	0.0
8	0	38.0	13.8	28.7	4.6	0.0	0.0	14.9	5680.5	3680.5
	1	0.0	1.9	50.7	2.0	0.0	0.0	45.4	3003.8	1684.3
	2	0.0	1.9	50.7	2.0	0.0	0.0	45.4	7001.9	5686.2
	3	69.4	5.0	22.5	2.0	0.0	0.0	1.1	3313.8	0.0

Figure 4.4: Regular Gasoline Blend Recipes along with blend volumes and closing inventories obtained for different time periods

Multiple Midgrade Recipes										
Index	Time Period	ALK	BUT	HCL	HCN	LCN	LNAP	RFT	$V_{blend,mid,t}$	$V_{close,mid,t}$
1	0	0	6.2	37.2	4.5	0.0	0.0	52.1	4881.9	4881.9
	1	0.0	3.3	40.4	2.0	0.0	0.0	54.2	6337.9	4219.7
	2	4.0	3.5	38.8	2.0	0.0	0.0	51.7	9351.5	5571.3
	3	0.0	3.3	40.4	2.0	0.0	0.0	54.2	2428.7	0.0
2	0	0.0	7.9	35.1	6.1	0.0	0.0	50.8	3005.7	3005.7
	1	0.0	3.3	40.4	2.0	0.0	0.0	54.2	7550.6	3556.4
	2	0.0	3.3	40.4	2.0	0.0	0.0	54.2	8148.7	3705.1
	3	72.1	6.6	11.1	2.0	0.0	0.0	8.2	4294.9	0.0
3	0	0.0	6.7	36.6	5.0	0.0	0.0	51.8	4161.7	4161.7
	1	0.0	3.3	40.4	2.0	0.0	0.0	54.2	6858.9	4020.6
	2	19.9	5.0	20.7	1.9	0.0	6.7	45.8	7511.0	3531.6
	3	13.3	3.9	35.0	2.0	0.0	0.0	45.8	4468.4	0.0
4	0	0.0	7.0	36.3	5.2	0.0	0.0	51.5	3820.2	3820.2
	1	0.0	3.3	40.4	2.0	0.0	0.0	54.2	5830.9	2651.1
	2	54.3	6.6	0.0	0.6	7.9	9.0	21.6	5529.3	180.5
	3	0.0	2.9	36.3	0.0	12.5	0.0	48.3	7819.5	0.0
5	0	44.8	9.3	17.7	5.5	0.0	0.0	22.7	3522.5	3522.5
	1	36.2	4.5	21.6	0.0	12.5	0.0	25.2	6189.0	2711.4
	2	15.6	4.0	34.1	2.0	0.0	0.0	44.3	8352.7	3064.1
	3	51.0	5.6	19.7	2.0	0.0	0.0	21.7	4935.9	0.0
6	0	0.0	6.6	36.6	5.0	0.0	0.0	51.8	4178.6	4178.6
	1	0.0	2.9	36.3	0.0	12.5	0.0	48.3	9439.9	6618.5
	2	67.9	6.4	12.8	2.0	0.0	0.0	10.9	7180.3	5798.8
	3	39.7	5.1	24.3	2.0	0.0	0.0	28.9	2201.2	0.0
7	0	0.0	7.6	35.5	5.9	0.0	0.0	51.0	3211.1	3211.1
	1	0.0	3.3	40.4	2.0	0.0	0.0	54.2	6268.2	2479.3
	2	0.0	3.3	40.4	2.0	0.0	0.0	54.2	8775.0	3254.3
	3	69.8	7.3	0.0	1.9	0.0	6.9	14.0	4745.7	0.0
8	0	0.0	6.2	37.1	4.6	0.0	0.0	52.1	4744.2	4744.2
	1	0.0	2.9	36.3	0.0	12.5	0.0	48.3	8552.4	6296.7
	2	41.8	5.2	23.4	2.0	0.0	0.0	27.5	6304.4	4601.0
	3	58.7	6.0	16.6	2.0	0.0	0.0	16.8	3399.0	0.0

Figure 4.5: Mid-grade Gasoline Blend Recipes along with blend volumes and closing inventories obtained for different time periods

Multiple Premium Recipes										
Index	Time Period	ALK	BUT	HCL	HCN	LCN	LNAP	RFT	$V_{blend,prm,t}$	$V_{close,prm,t}$
1	0	47.5	5.7	6.0	0.0	23.5	0.0	17.2	6804.5	5804.5
	1	47.1	6.8	10.7	1.9	0.8	0.0	32.6	8490.5	5294.9
	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	294.9
	3	56.1	7.8	0.0	2.0	0.0	4.2	30.0	3705.1	0.0
2	0	62.2	6.3	0.0	0.0	24.6	0.0	6.9	6203.3	5203.3
	1	53.0	7.1	8.5	2.0	0.0	0.0	29.2	7542.9	3746.2
	2	0.0	4.7	30.1	2.0	0.0	0.0	63.1	2713.7	1459.9
	3	0.0	4.7	30.1	2.0	0.0	0.0	63.1	2540.1	0.0
3	0	82.7	8.0	0.0	4.1	0.0	0.0	5.1	5808.0	4808.0
	1	48.6	6.5	6.2	0.0	12.5	0.0	26.1	8399.3	4207.3
	2	74.0	8.1	0.0	2.0	0.0	0.0	15.8	2034.5	1241.7
	3	0.0	4.7	30.1	2.0	0.0	0.0	63.1	2758.3	0.0
4	0	82.6	8.0	0.0	4.1	0.0	0.0	5.3	5913.6	4913.6
	1	40.9	6.6	13.5	2.0	0.0	0.0	37.0	9774.3	5688.0
	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	688.0
	3	0.0	4.7	30.1	2.0	0.0	0.0	63.1	3312.0	0.0
5	0	0.0	4.3	33.7	4.2	0.0	0.0	57.9	5720.1	4720.1
	1	0.0	4.7	30.1	2.0	0.0	0.0	63.1	9518.4	5238.5
	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	238.5
	3	0.0	6.3	7.0	1.8	0.0	13.3	71.6	3761.5	0.0
6	0	39.2	6.1	17.2	3.8	0.0	0.0	33.7	6808.4	5808.4
	1	0.0	4.3	26.0	0.0	12.5	0.0	57.2	3322.3	130.7
	2	0.0	4.7	30.1	2.0	0.0	0.0	63.1	5175.9	306.6
	3	56.3	7.3	7.2	2.0	0.0	0.0	27.1	3693.4	0.0
7	0	81.8	8.0	0.0	3.9	0.0	0.0	6.2	6507.7	5507.7
	1	27.9	6.0	18.8	2.0	0.0	0.0	45.3	9052.1	5559.8
	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	559.8
	3	56.1	7.3	7.3	2.0	0.0	0.0	27.3	3440.2	0.0
8	0	39.6	6.1	17.1	3.9	0.0	0.0	33.2	6575.3	5575.3
	1	25.7	5.6	16.0	0.2	11.0	0.0	41.5	5443.8	2019.1
	2	0.0	4.7	30.1	2.0	0.0	0.0	63.1	3693.7	712.8
	3	0.0	6.6	3.7	1.7	0.0	15.2	72.8	3287.2	0.0

Figure 4.6: Premium Gasoline Blend Recipes along with blend volume and closing inventories obtained for different time periods

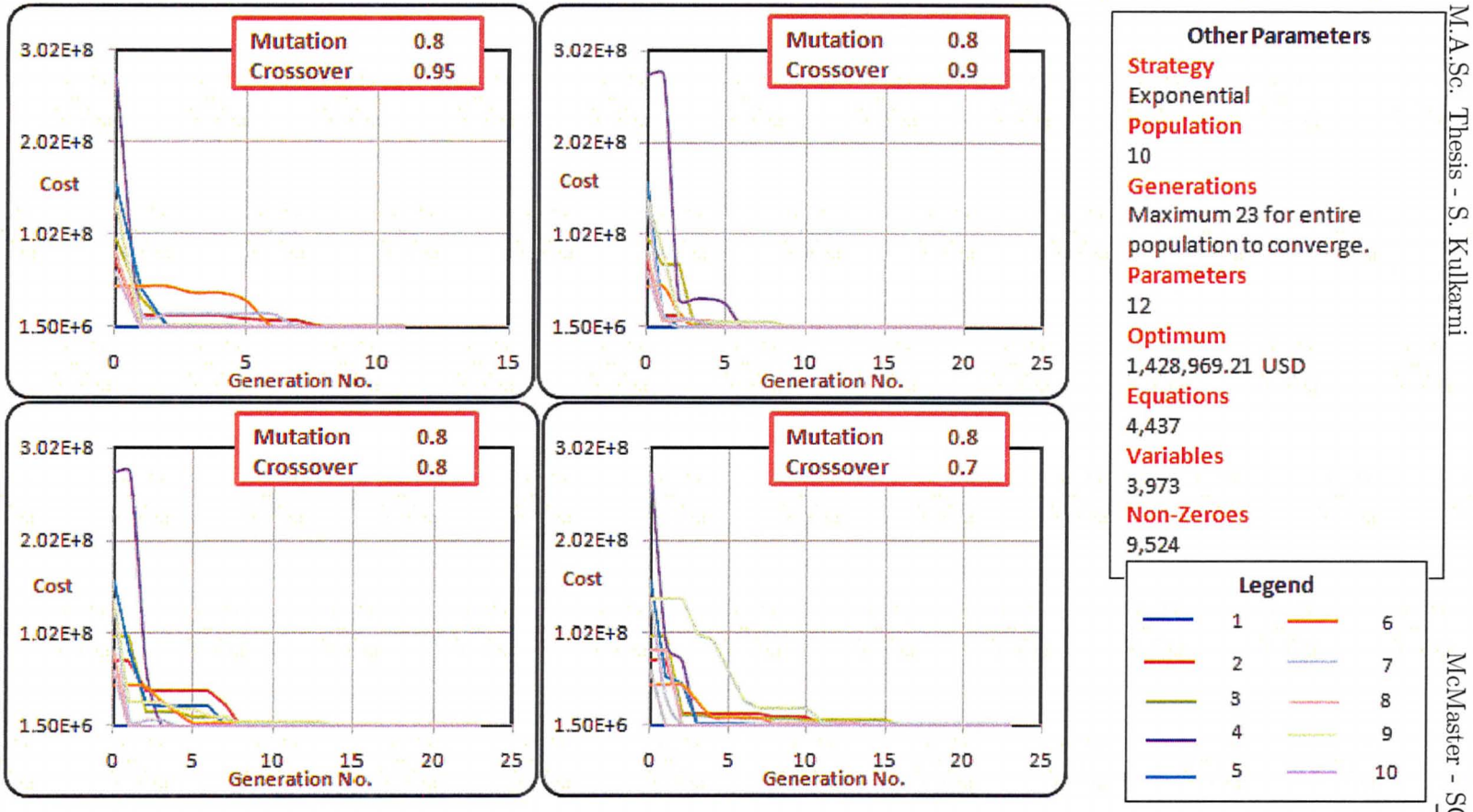

Figure 4.7: The rate of convergence for every population member with varying crossover probability and fixed mutation probability using an exponential strategy.

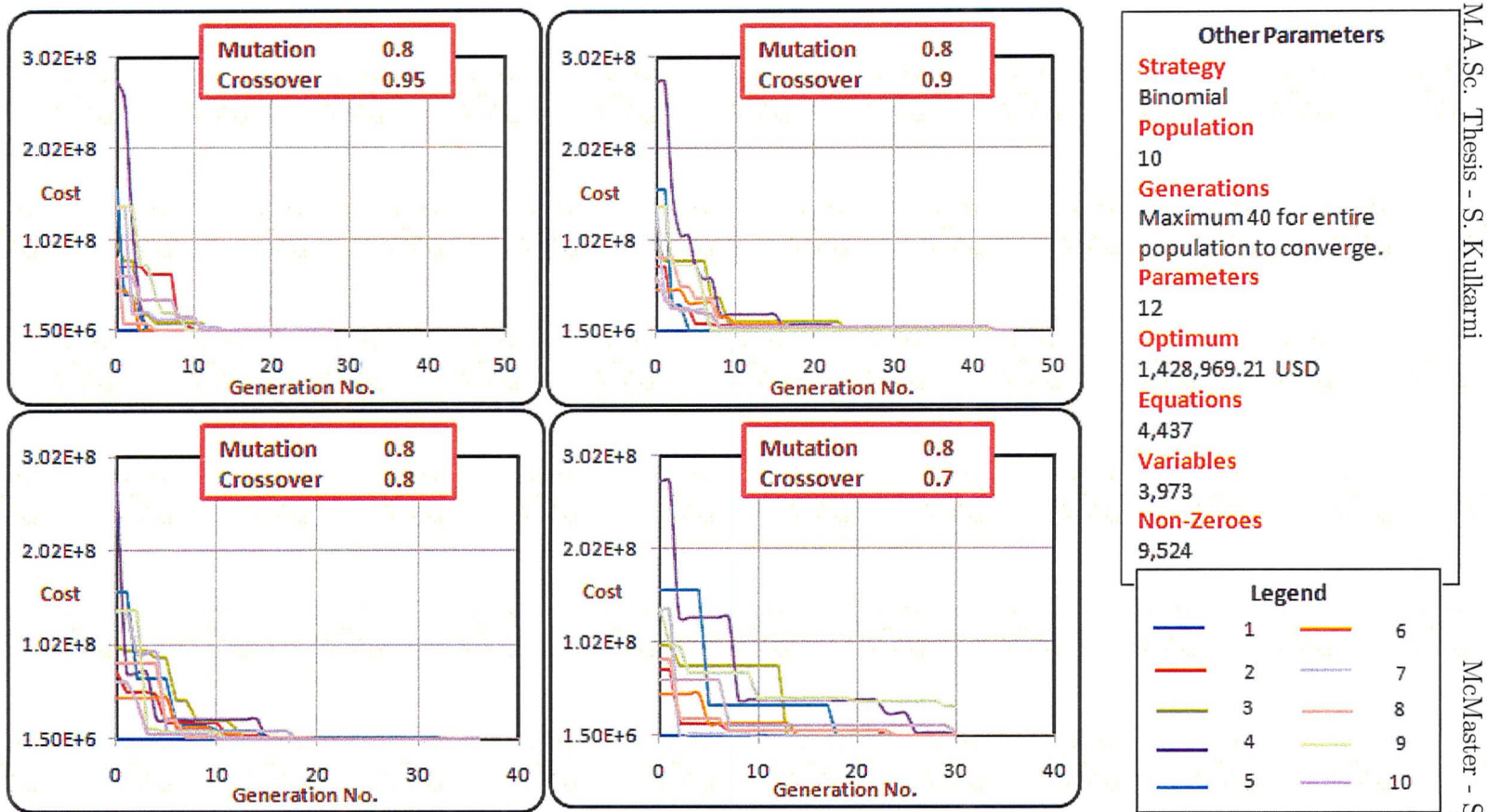

Figure 4.8: The rate of convergence for every population member with varying crossover probability and fixed mutation probability using a binomial strategy.

4.3 14-Time Period Optimization

A larger size real life problem was solved using data partially extracted from AspenTech. The physical properties of the blend components and gasoline in the blending tank at the beginning of the time horizon were same as in tables 4.6 and 4.5 respectively. The specification on the them for the each gasoline grade were same as in table 4.7. The demand for each gasoline grade is shown in table 4.8 and plotted in chart 4.9. In order to remain feasible the supply of all the components was fixed to be 500 $MBbl$ in every time period.

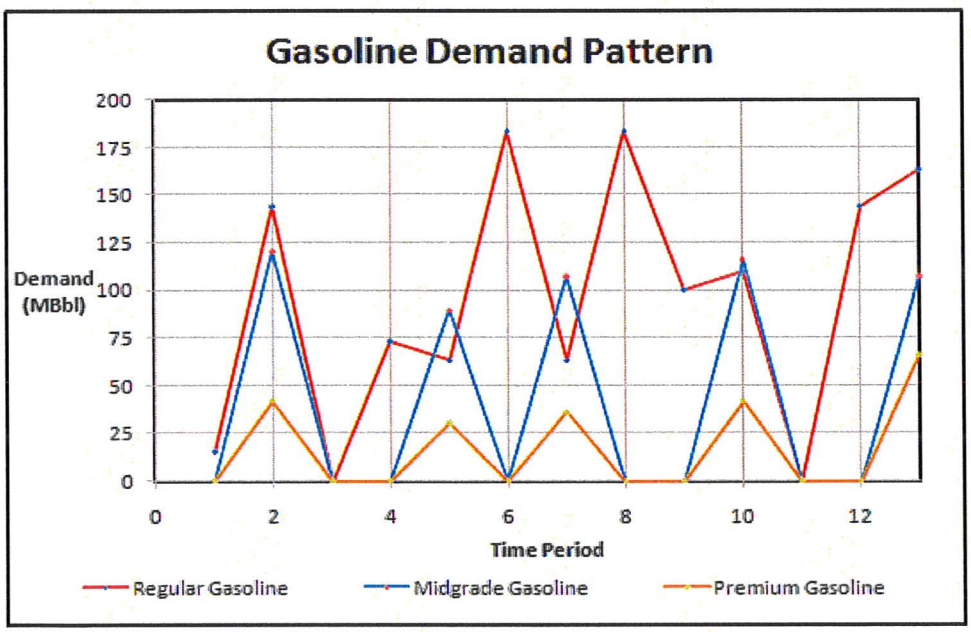

Figure 4.9: Demand, $V_{out,g,t}$, in $MBbl$ for each gasoline grade in every time period.

Time Period	Regular	Mid-grade	Premium
0	15.13	0	0
1	143.63	120	42
2	0	0	0
3	73	0	0
4	63	89.3	30
5	183.05	0	0
6	63	107.1	36
7	183	0	0
8	100	0	0
9	110	115.92	42
10	0	0	0
11	143.6	0	0
12	163	107.1	66

Table 4.8: Demand in $MBbl$ for various grades of Gasoline in each time periods

Material	Opening Volume
ALK	100
BUT	43

HCL	25
HCN	50
LCN	250
LNAP	125
RFT	150
Regular	70
Midgrade	185
Premium	11

Table 4.9: Opening Volume ($MBbl$) in blend component and tanks

4.4 14 Time Period Results Discussion

The nature of this problem resulted in large number of equations and variables. From [24] the size of the population members should be 5 to 10 times the vector size. The vector size (i.e the number of optimizing parameters) is 39 and hence, the population member size of 200 was used. Unlike the previous 4-time period simulation this problem converged slightly better using a binomial strategy of crossover, and can be seen in the the charts of figure 4.14.

Similar observations regarding results can be made in this problem.

While the chart 4.10 shows all the unique feasible solutions with cost value within 1% of the optimal value, the tables and charts in figures 4.11, 4.12 and 4.13 show the multiple solutions against the optimum cost value ($38,029,918.52 USD$). Once again, the results are to be interpreted as previous example.

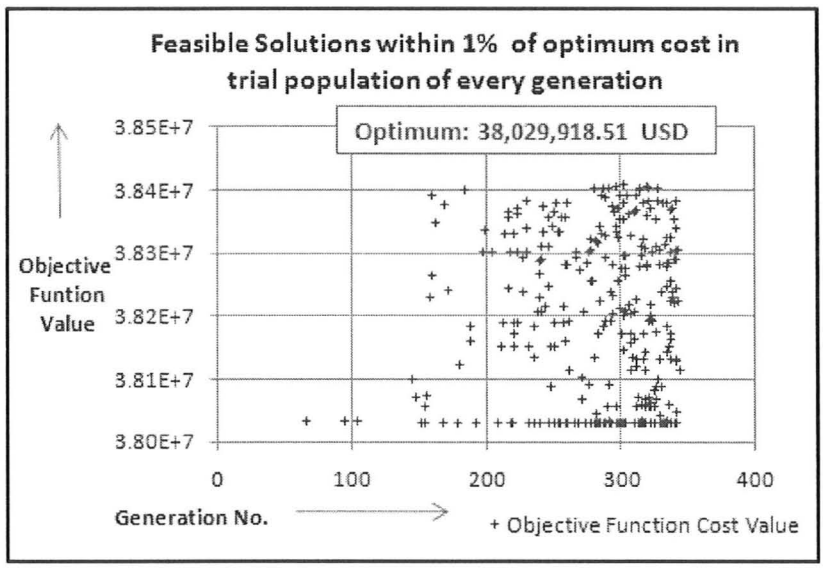

Figure 4.10: Each solution point is a feasible trial population member (14 Time Period), whose fitness value is within 1% of the optimum fitness value.

Time Periods												
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Regular Gasoline ($V_{blend,reg,t}$)												
117.5	81.2	114.6	105.0	66.3	67.8	133.3	129.2	129.9	0.0	94.2	49.3	82.4
95.9	66.8	105.9	81.7	140.1	56.4	53.4	184.7	189.2	91.8	51.9	0.0	52.6
68.8	125.2	72.4	59.1	182.5	50.6	53.4	184.7	189.2	87.0	51.9	0.0	45.6
146.6	71.6	170.0	98.5	72.6	111.6	50.4	144.7	52.5	0.0	116.5	25.5	109.8
109.7	81.8	103.4	97.9	174.3	0.0	94.0	105.1	143.1	0.0	147.2	87.9	25.9
146.6	71.6	170.0	98.5	82.3	111.6	50.4	145.2	52.5	0.0	116.5	25.5	99.6
80.2	164.7	71.4	112.4	126.3	152.5	0.0	111.6	126.3	0.0	129.4	37.7	57.8
108.9	136.1	41.2	41.4	152.1	106.7	163.8	125.2	34.6	101.6	56.0	30.1	72.8
Regular Gasoline ($V_{close,reg,t}$)												
172.3	109.9	224.4	256.4	259.7	144.4	214.7	160.9	190.8	80.8	174.9	80.6	0.0
150.8	74.0	179.9	188.6	265.7	139.0	129.4	131.1	220.3	202.1	254.0	110.4	0.0
123.7	105.3	177.6	163.8	283.2	150.8	141.2	142.9	232.1	209.1	261.0	117.4	0.0
201.5	129.5	299.4	324.9	334.5	263.1	250.5	212.2	164.8	54.8	171.3	53.2	0.0
164.6	102.7	206.1	231.0	342.3	159.3	190.3	112.5	155.6	45.6	192.8	137.1	0.0
201.5	129.5	299.4	324.9	344.3	272.9	260.3	222.5	175.0	65.0	181.6	63.4	0.0
135.1	156.1	227.5	267.0	330.3	299.8	236.8	165.4	191.7	81.7	211.1	105.2	0.0
163.8	156.2	197.4	165.8	254.9	178.5	279.3	221.5	156.1	147.7	203.7	90.2	0.0

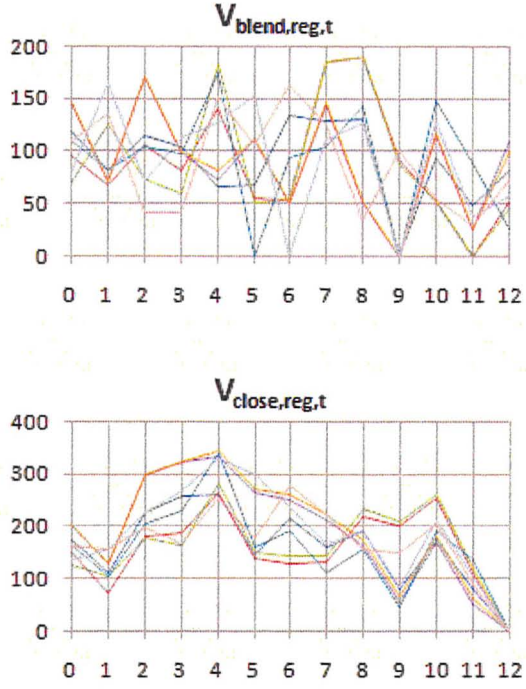

Figure 4.11: Each line in the graph shows the blend amount and closing inventory (in $MBbl$) in the regular gasoline blend tank in each time period for 7 randomly chosen solution points from chart 4.10. The corresponding values are shown in the adjacent table.

Time Periods												
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Midgrade Gasoline ($V_{blend,mid,t}$)												
27.3	0.0	27.2	35.5	0.0	70.7	46.7	0.0	50.1	0.0	35.8	0.0	61.2
27.9	68.8	74.1	26.6	0.0	45.4	36.2	0.0	0.0	0.0	0.0	0.0	75.4
36.3	54.8	47.9	47.0	0.0	59.4	36.2	0.0	0.0	31.4	0.0	0.0	41.5
33.4	26.7	0.0	81.5	58.6	68.4	0.0	0.0	42.0	0.0	0.0	0.0	43.9
0.0	55.7	27.7	0.0	0.0	28.0	77.7	55.1	0.0	47.2	28.0	0.0	35.0
33.4	26.7	0.0	81.5	52.8	68.4	0.0	0.0	42.0	0.0	0.0	0.0	49.7
49.8	0.0	39.4	25.3	53.7	0.0	39.5	33.8	27.6	0.0	33.0	0.0	52.3
71.1	0.0	55.8	0.0	27.9	33.0	0.0	54.8	0.0	62.4	0.0	0.0	49.4
Midgrade Gasoline ($V_{close,mid,t}$)												
212.3	92.3	119.4	154.9	65.6	136.3	75.9	75.9	126.0	10.1	45.9	45.9	0.0
212.9	161.7	235.8	262.4	173.1	218.5	147.6	147.6	147.6	31.7	31.7	31.7	0.0
221.3	156.0	204.0	251.0	161.7	221.0	150.1	150.1	150.1	65.6	65.6	65.6	0.0
218.4	125.1	125.1	206.6	175.9	244.2	137.1	137.1	179.2	63.2	63.2	63.2	0.0
185.0	120.7	148.4	148.4	59.1	87.1	57.7	112.9	112.9	44.1	72.1	72.1	0.0
218.4	125.1	125.1	206.6	170.0	238.4	131.3	131.3	173.3	57.4	57.4	57.4	0.0
234.8	114.8	154.2	179.5	143.9	143.9	76.3	110.1	137.7	21.8	54.8	54.8	0.0
256.1	136.1	191.9	191.9	130.5	163.5	56.4	111.2	111.2	57.7	57.7	57.7	0.0

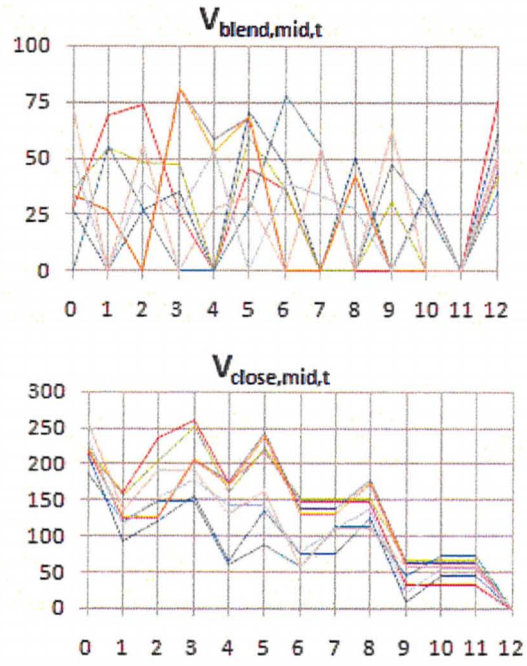

Figure 4.12: Each line in the graph shows the blend amount and closing inventory (in $MBbl$) in the midgrade gasoline blend tank in each time period for 7 randomly chosen solution points from chart 4.10. The corresponding values are shown in the adjacent table.

Time Periods												
0	1	2	3	4	5	6	7	8	9	10	11	12
Premium Gasoline ($V_{blend,prm,t}$)												
25.3	98.8	28.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	52.6
46.2	0.0	0.0	61.7	0.0	68.2	0.0	0.0	0.0	0.0	0.0	0.0	28.9
64.9	0.0	49.7	0.0	0.0	60.1	0.0	0.0	0.0	0.0	0.0	0.0	30.3
0.0	71.7	0.0	0.0	38.8	0.0	0.0	35.3	0.0	0.0	0.0	29.7	29.5
0.0	32.5	38.9	82.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	25.1	26.3
0.0	71.7	0.0	0.0	34.9	0.0	0.0	34.8	0.0	0.0	0.0	29.7	33.9
40.0	0.0	59.2	32.3	0.0	0.0	0.0	0.0	0.0	43.0	0.0	0.0	30.5
0.0	43.9	73.0	0.0	0.0	30.3	0.0	0.0	29.3	0.0	0.0	0.0	28.5
Premium Gasoline ($V_{close,prm,t}$)												
36.3	93.1	121.4	121.4	91.4	91.4	55.4	55.4	55.4	13.4	13.4	13.4	0.0
57.2	15.2	15.2	76.9	46.9	115.1	79.1	79.1	79.1	37.1	37.1	37.1	0.0
75.9	33.9	83.6	83.6	53.6	113.7	77.7	77.7	77.7	35.7	35.7	35.7	0.0
11.0	40.7	40.7	40.7	49.5	49.5	13.5	48.8	48.8	6.8	6.8	36.5	0.0
11.0	1.5	40.4	122.5	92.5	92.5	56.5	56.5	56.5	14.5	14.5	39.7	0.0
11.0	40.7	40.7	40.7	45.6	45.6	9.6	44.4	44.4	2.4	2.4	32.1	0.0
51.0	9.0	68.2	100.4	70.4	70.4	34.4	34.4	34.4	35.5	35.5	35.5	0.0
11.0	12.9	86.0	86.0	56.0	86.3	50.3	50.3	79.5	37.5	37.5	37.5	0.0

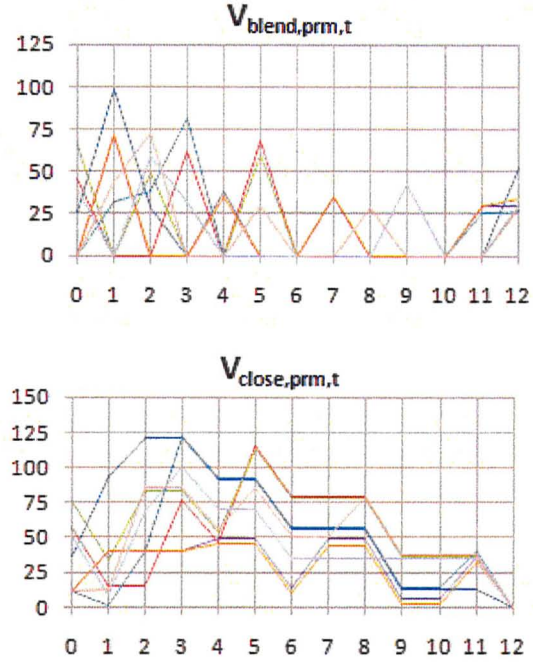

Figure 4.13: Each line in the graph shows the blend amount and closing inventory (in $MBbl$) in the premium gasoline blend tank in each time period for 7 randomly chosen solution points from chart 4.10. The corresponding values are shown in the adjacent table.

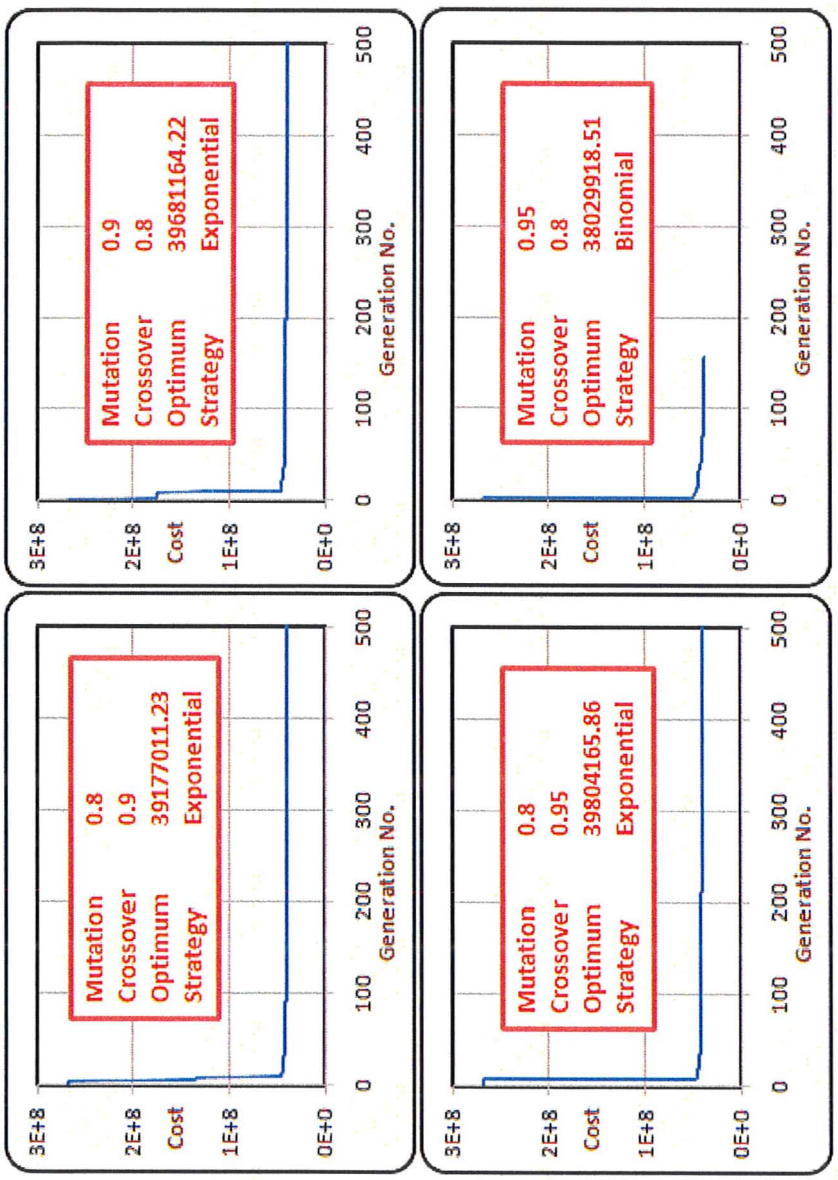

Figure 4.14: The rate of convergence plotted as best cost in every generation for the 14-time period optimization problem.

Chapter 5

Multi-Processor Optimization

The problems that can be easily divided across many cores into different parallel tasks without much effort are called as Embarrassingly Parallel problems. In differential evolution, evaluation of each population member is independent of other member. Hence, all the population members can be evaluated on separate cores before performing the operation of mutation and crossover where synchronization might be necessary. This makes the differential evolution algorithm a type of embarrassingly parallel problems. This fact about differential evolution was exploited and a distributed computing approach was adopted to solve large number of population members in parallel.

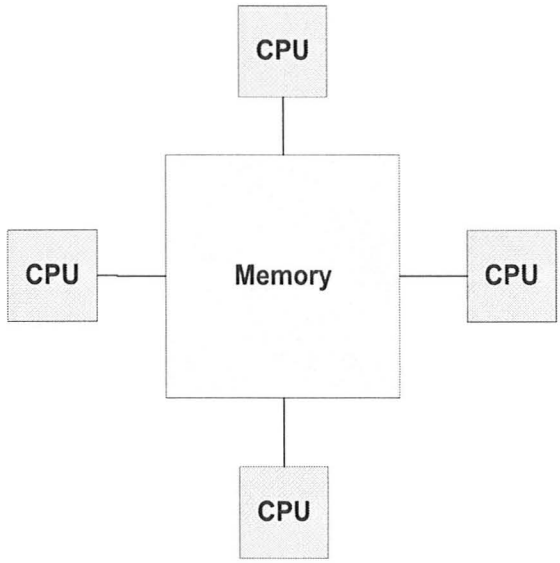

Figure 5.1: Shared Memory with Uniform Memory Access Time. Adapted from [3]

5.1 Shared and Distributed Memory models

The two main memory models available to the programmer are the distributed and shared memory models. The shared memory architecture has a global address space available for all the processors. They were originally used to designate a symmetric multi processor (SMP) system where all the processors would access the same memory space at same speed thereby having a uniform memory access (UMA) time [7]. A non-uniform access time (NUMA) of memory is achieved by linking several SMP through by various interconnects[7]. In an SMP, every processor has a non-shared cache at its disposal. These cache memories save memory access time due to locality of reference. However, sometimes when a cache memory is updated for one

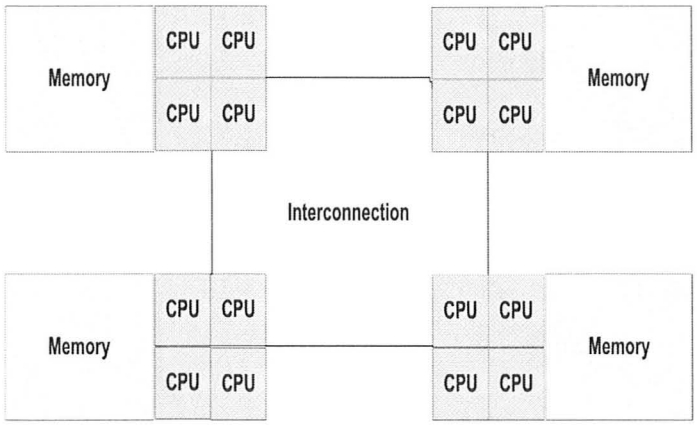

Figure 5.2: Shared Memory with Non-Uniform Memory Access Time. Adapted from [3]

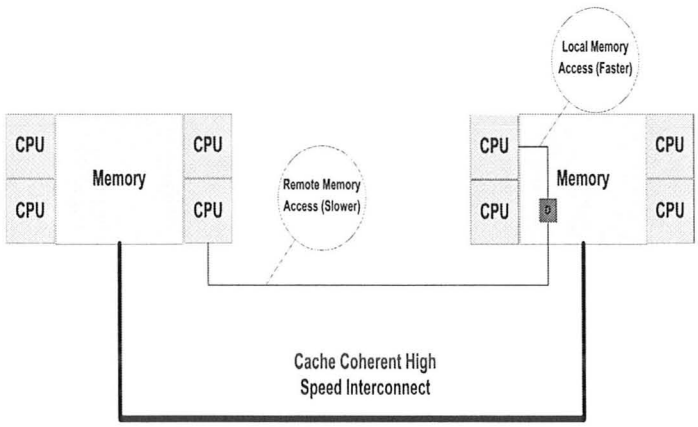

Figure 5.3: Cache Coherent Non-Uniform Memory Access Time (cc-NUMA). Adapted from [7]

processor it does not get reflected into other cache memory. This can lead to data consistency issues, also termed as false-sharing [7]. When a NUMA architecture machine maintains cache coherency, it is called as cc-NUMA (cache coherent NUMA) [7].

Distributed memory architectures differ from the SMP architecture in a

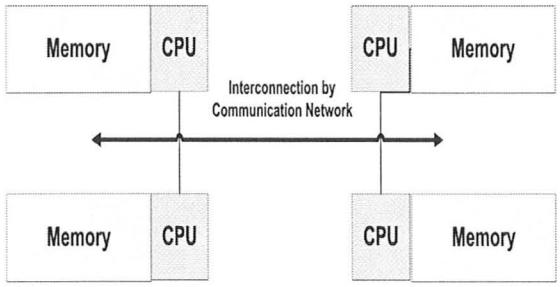

Figure 5.4: Distributed Memory Architecture. Adapted from [3]

way that there is no common sharable address space among different processors. In this memory model each processor has its own local memory. Inter-process communication is carried via network communication channels. Unlike the SMP models cache coherency does not apply here as each processor has its own local memory and any changes made therein do not apply for other processors[3]. The communication of data and tasks among various processors has to be explicitly handled by the programmer through the available programming interfaces.

5.2 Parallel Programming Models

The memory models described above are exploited by the programmer with the available programming models available at his disposal. However, a programming model may not be associated with a particular architecture. The choice of model is solely at programmer's discretion. The most popular programming models are the Threads and Message Passing Interface Model. In both these models the parallelism is identified by the programmer.

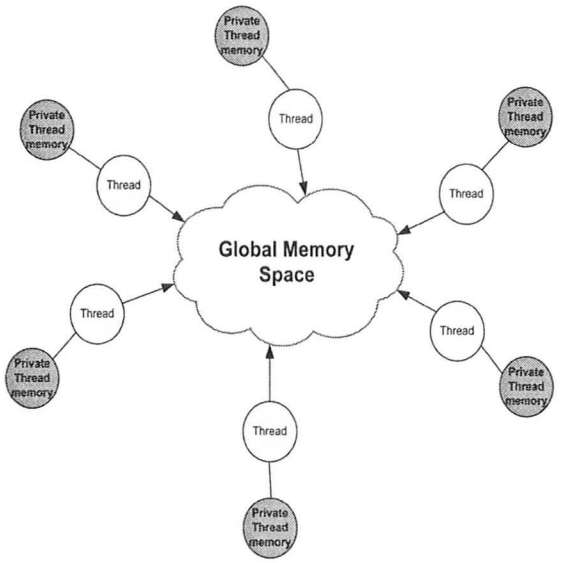

Figure 5.5: OpenMP Memory Model. Adapted from [7]

Thread-based Models

The Thread-based model includes the POSIX and the OpenMP implementation and is usually associated with the shared memory architectures. In this model, a single program has many concurrent execution paths. When the process begins, it forks into many parallel threads that work simultaneously using its own local private memory called as TLS - Thread Local Storage and a shared global address space. This is also called as the fork-join model. This kind of parallelism can be obtained by invoking some library routines (POSIX) or through compiler derivatives (OpenMP).

Distributed Processing Models

As mentioned previously distributed memory architecture has several processors with their own memory. As a result each processor needs to commu-

nicate its data explicitly through the inter-connects, i.e by passing messages. The message passing implementation was standardized with the introduction of message passing interface (MPI) - which is a specification of what a library implementing message passing routines should do. Recently, MPI was introduced for a Shared Memory Processor. From a programmer's view point, the entire process is being executed on many processors (i.e. multiple copies of the same program). Data Parallelism (where same computation is performed on different sets of unrelated data) is achieved when the master process distributes the data across processors using message passing routines (MPI specification of scatter) and then collects the results from the slave processes (MPI specification of gather).

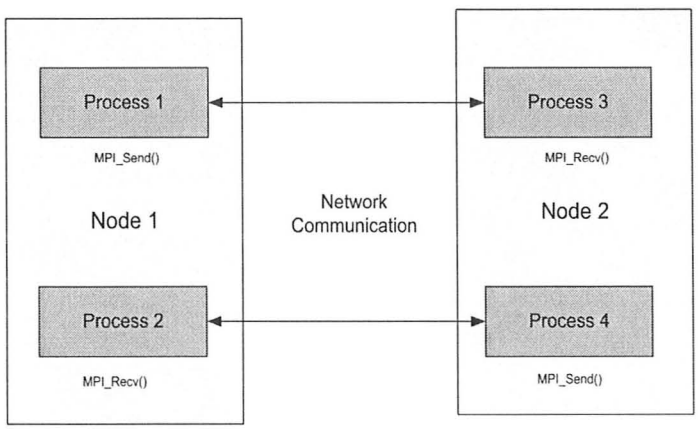

Figure 5.6: MPI Memory Model. Adapted from [3]

5.3 Parallelizing Differential Evolution

In task parallelism computer instructions are forked (threads) across various cores on same or different sets of data. Unlike task parallelism, data parallelism divides a chunk of data across various processors. The most common way of obtaining parallelization is Single Program Multiple Data technique (SPMD) [4]. In this method many processes execute the same program but on its own set of data. This kind of parallelism is found in differential evolution where each population member can be evaluated at the same time on different processors.

The differential evolution was parallelized using communication routines of Message Passing Interface specification. This specification was used over shared memory approach because the GNU's library of linear programming optimization (GLPK) and the Sparse 1.4, is not thread-safe. It means that the library environment is static and share-able across various threads which makes the data parallelism hard to achieve using shared memory architectures. Using OpenMP for thread-unsafe third party libraries would result into data corruption causing the program to crash. The process of parallelism for the differential evolution begins with initializing the MPI environment using MPI_Initialize routine. Each processor knows the size of its communication group and its rank within it. It uses this information determines the available concurrency i.e the number of population members

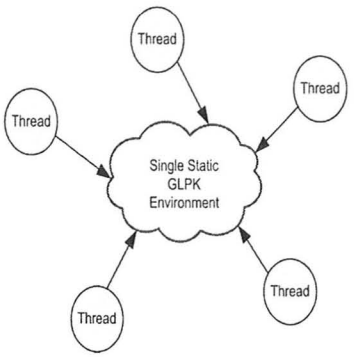

Figure 5.7: OpenMP Implementation of Differential Evolution. All threads accessed the same glpk environment leading to data inconsistency.

it can evaluate. This is obtained by:

$$avlConcurrency = N_p \div n \tag{5.1}$$

where,

N_p is Population member size

n is number of processors.

The index of the population members for each process is obtained by using the rank of each processor and the *avlConcurrency*. The master thread generates the entire population in random before scattering the population members (i.e. data) in equal or unequal chunks across various processors. This is achieved using the MPI_Scatterv routine which scatters a buffer in a specified chunk size to all the available tasks. This is a blocking operation which means that all the processes will be bared to proceed further until the master process finishes the scatter. The master process then gathers the updated population members and their fitness function, calculates the best

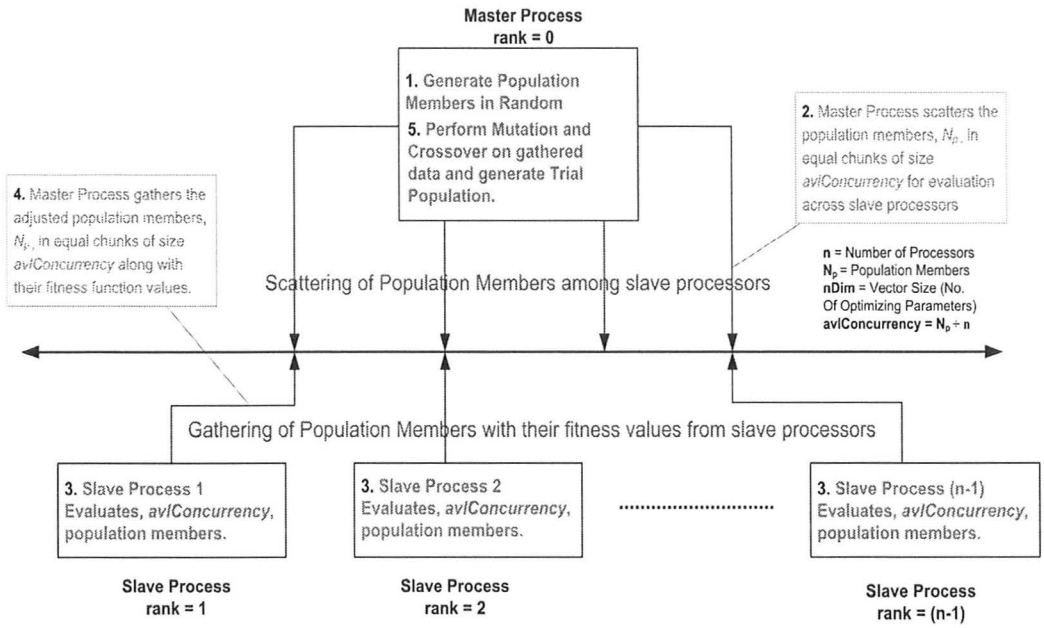

Figure 5.8: The Master Process, usually with a rank 0, generates the initial population members in random and scatters them in specified chunk size using the MPI_Scatterv routine in the specified chunk size.

fitness function member using the MPI_Gatherv routine , and performs crossover and mutation before proceeding to scatter the newly generated trial population. This is carried until the specified criteria of termination is reached.

5.4 Performance Gain and Speedup

The parallelized differential evolution for gasoline blending was executed on SHARCNET - Shared Hierarchical Academic Research Computing Network (https://www.sharcnet.ca/) using OpenMPI on its 10-19, (called as hnd10-hnd19) 32 core Opteron-based nodes of the cluster Hound. This clus-

ter uses the Linux-based Community Enterprise Operating System (CentOS 5.3) and Infiniband interconnect. Please see more information about this cluster on https://www.sharcnet.ca/my/systems/show/42.

The tests were performed on varying number of processors on hnd[10-19], all of then Opteron@2.2GHz. 500 generations of differential evolution, each of 200 population members, solving two sparse and one simplex for each population member converged from days (3-4) to hours ($\prec$ half a day). The graph of wall-clocked time vs. the number of processors is shown below.

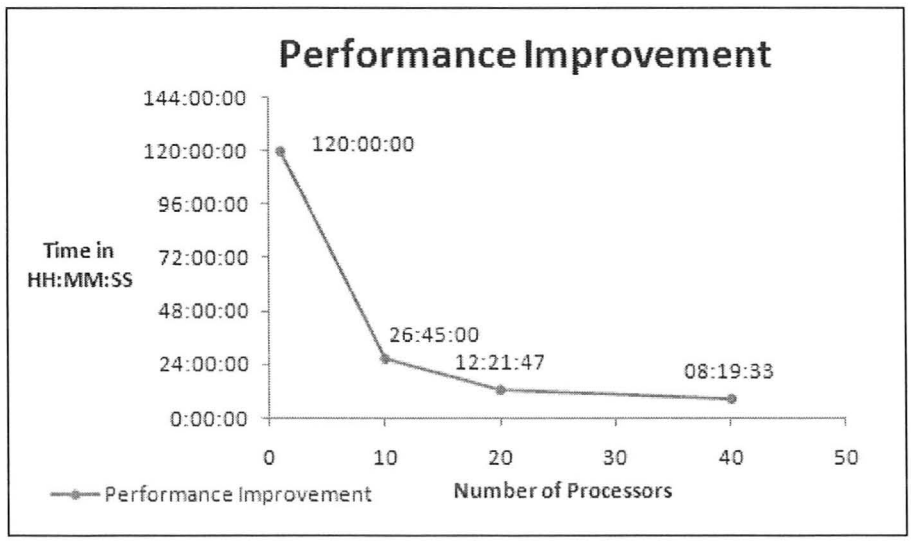

Figure 5.9: The performance improvement using OpenMPI on Opteron-based cores hosted on Hound cluster.

Chapter 6

Conclusion and Future Work

As can be seen from the numerical results, the differential evolution approach give refineries the choice of multiple blend recipes against same cost value. They can choose a solution that best suits their current condition at no or very little additional cost. For example, it can choose to use a recipe that preserves usage of Alkylate for its octane value. These solutions maintain the physical characteristic of the gasoline, and costs the same (or in a chosen margin) over a given time horizon. This approach of having multiple solutions by calculating blend amount of gasoline in every time period over a given time horizon gives an important unexplored facet of the nature of the problem. The algorithm will be enhanced to accommodate multiple objectives (e.g. preserve components); improve resource allocation strategy(e.g. multi-product swing tanks, blender allocation and scheduling etc) and will be parallelized using OpenMP for a possible better performance.

Bibliography

[1] Anonymous. http://www.chemcool.com.

[2] ASTM. Standard specification for automotive spark-ignition engine fuel. astm d4814-09. annual book of astm standards v.05.03, 1994.

[3] Blaise Barney. Introduction to parallel computing. https://computing.llnl.gov/tutorials/parallel_comp/ (accessed on august 11,2009).

[4] Paul Black. Algorithms and theory of computation handbook, crc press llc, 1999, "single program multiple data", in dictionary of algorithms and data structures. http://www.itl.nist.gov/div897/sqg/dads/html/singleprogrm.html (accessed on august 11, 2009).

[5] C E Bodington, Chesapeake Decision Sciences, San Anselmo, Chesapeake Decision Sciences, New Providence, and T E Baker. A history of mathematical programming in the petroleum industry. *Interfaces*, 1990:117–127, 2004.

[6] Der-ming Chang, Cheng-ching Yu, and I-lung Chien. Coordinated control of blending systems. *Control*, 6:495–506, 1998.

[7] Barbara Chapman, Gabriele Jost Pas, and Ruud Van Der. *Using OpenMP Portable Shared Memory Parallel Programming.* MIT Press, 2007.

[8] Swagatam Das, Amit Konar, and Uday K Chakraborty. Two improved differential evolution schemes for faster global search. *Population (English Edition)*, pages 991–998, 2005.

[9] Kalyanmoy Deb. Evolutionary practical optimization. *Mechanical Engineering*, pages 3093–3132, 2007.

[10] Klaus Glismann. Short-term scheduling and recipe optimization of blending processes. *Computers and Chemical Engineering*, 25:627– 634, 2001.

[11] Bruce Hamilton. Faq: Automotive gasoline. http://www.webpak.net.libaccess.lib.mcmaster.ca/~marriott/gasoline.htm (accessed on august 12 2009).

[12] M Mathew Leenus Jehan. Differential evolution for multi-objective optimization. *Chemical Engineering*, 031, 2003.

[13] J D Kelly and Honeywell Process Solutions. Logistics: the missing link in blend scheduling optimization. *Hydrocarbon Processing*, 2006.

[14] Jeffrey D Kelly and Danielle Zyngier. An improved milp modeling of sequence-dependent switchovers for discrete-time scheduling problems. *Society*, pages 4964–4973, 2007.

[15] Jeffrey Dean Kelly and Honeywell Process Solutions. The unit-operation-stock superstructure (uoss) and the quantity-logic- quality paradigm (qlqp) for production scheduling in the process industries. *Production*, 2005.

[16] Kenneth V Price. *Differential Evolution: A Practical Approach to Global Optimization.* Springer Berlin Heidelberg, 2005.

[17] Operational Research, The Tata Iron, Steel Company Limited, and Operational Research. Strategic and operational management with optimization at tata steel. *Interfaces*, pages 6–19, 1995.

[18] A Singh, J F Forbes, P J Vermeer, and S S Woo. Model-based real-time optimization of automotive gasoline blending operations. *Journal of Process Control*, 10, 2000.

[19] Aseema Singh. Modelling and updating in real time optimization of gasoline blending, master's thesis, 1997.

[20] Raymond L Smith, Carlos A V Costa, and Rua Roberto Frias. Life cycle assessment of gasoline blending options. *Environmental Science and Technology*, 37:3724–3732, 2003.

[21] Rainer Storn. Differential evolution (de) for continuous function optimization (an algorithm by kenneth price and rainer storn). http://www.icsi.berkeley.edu/~storn/code.html (accessed on august 11, 2009).

[22] Rainer Storn. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. *Science*, pages 1–15, 1945.

[23] Rainer Storn. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, pages 341–359, 1997.

[24] Rainer Storn, Siemens Ag, and Otto-hahn Ring. On the usage of differential evolution for function optimization. *Fuzzy Information Processing Society NAFIPS. Biennial Conference of the North American*, page 519523, 1996.

[25] M Fatih Tasgetiren, Quan-ke Pan, and Yun-chia Liang. A discrete differential evolution algorithm for single machine weighted tardiness problem with sequence dependent setup times. *Computers and Operations Research*, 36:1900–1915, 2009.

[26] Agilent Technologies. Petroleum refining process. http://www.chem.agilent.com/cag/isa/hpi/c1.htm.

[27] Peter J. Vermeer, Clifford C. Pedersen, William M. Canney, and John S. Ayala. Blend control system all but eliminates with re-blends for canadian refiner. *Oil and Gas Journal*, June 28, 1997.

[28] Matthew Bartschi Wall. A genetic algorithm for resource-constrained scheduling. *Design*, 1996.

[29] Allan D Waren. Omega: An improved gasoline blending system for texaco. *Interfaces*, 1989:85–101, 1989.

Appendices

Appendix A

Software Engineering

A.1 Software Development

As mentioned in the introductory chapter a multi-tier software (client-server architecture) was developed to help users solve process system networks to improve productivity and efficiency. A relational database model was developed to store and retrieve persistent information related to the system. Efforts were made to keep all the information data-driven to eliminate any information assumptions. A team of 5 people was involved at various layers (or sub-layers) as can be seen in the Figure A.1. The purpose of this software, implementation overview and future work in this regard will be discussed in this appendix.

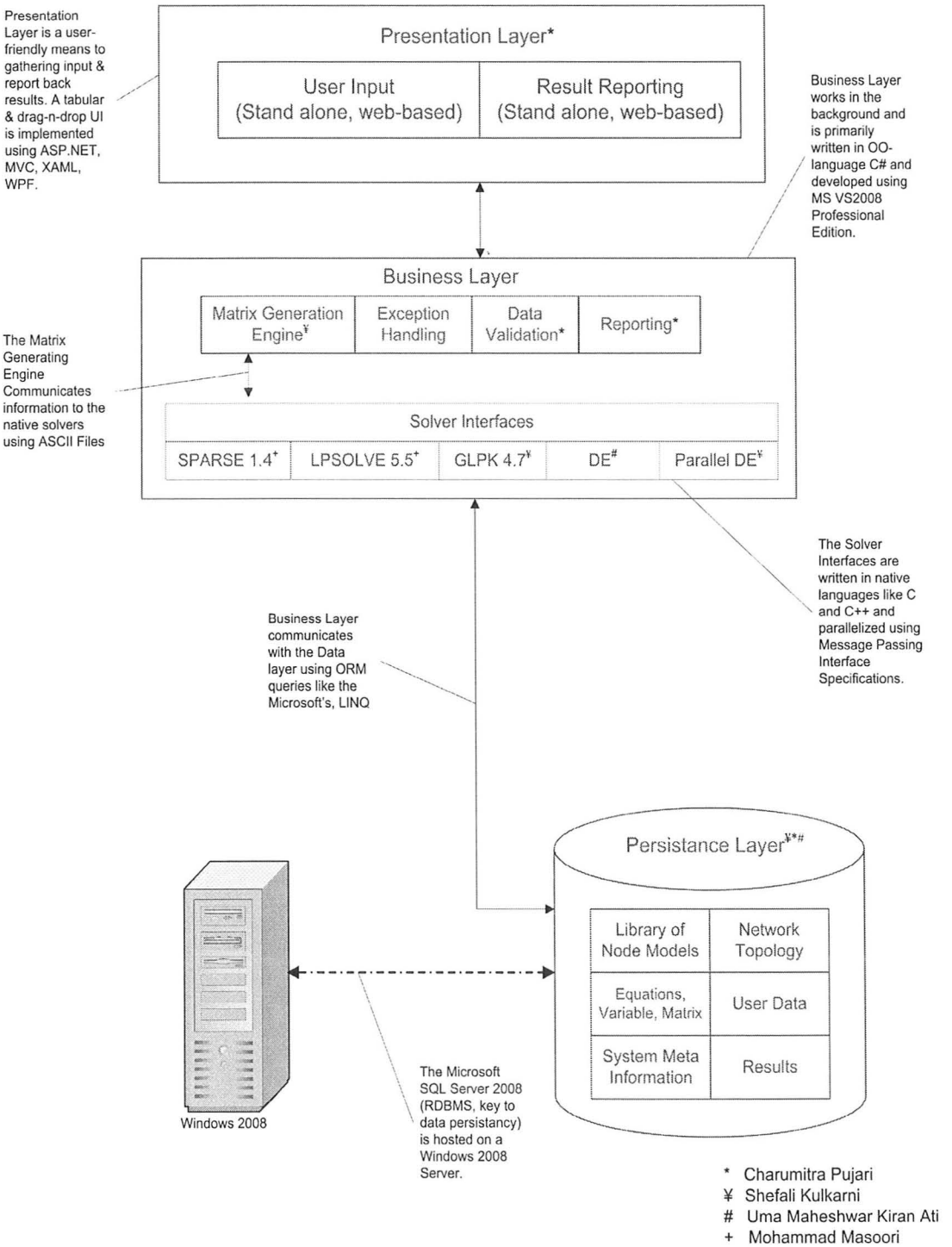

Figure A.1: Multi-Layered Software Implementation for Generating, Solving and Reporting of Process System Mathematical Models

The Software development and management tools used can be seen are shown in figure A.1. Some additional Software Productivity tools used are:

Subversion	Software Code Management and Version Control Server.
VS 2008 Professional Edition	Integrated Development Environment
VisualSVN	TortoiseSVN-based add-in for VS2008
.NET 3.5 Framework	C# Application Programming Interface
Bugzilla	Bugs and issue tracking
Scrumworks	Agile-based, Scrum Software Management Tool
Microsoft SQL Server 2008	Relational Database Management System
Windows 2008 Server	Windows-based server for Software Development
Visual Paradigm	Relational Database Designing Tool
LINQ	Language Integrated Query (.NET 3.0+)
Languages Used	C,C++, C#, XML, ASP etc.

This project was managed by a Incremental Agile Software Development methodology called Scrum. The tasks were identified and distributed to the members, who decomposed them into smaller tasks of 8 hours resulting in 40 hours of weekly time.

Software version controlling was done using VisualSVN (client) and Subversion (server). Issue tracking is being done by Bugzilla. Object Relational Mapping was used to query the database and Microsoft's .NET 3.0 Language Integrated Query which uses lambda expressions was used. The entire Project was divided into 3 main parts:

1. Mappings: Definition of all the classes that were mapped to the database.
2. Exception: Definition of all the possible exceptions: information in the database, input provided by the user etc.
3. Engine: The code that generates equations, variables, coefficients along with the files for the solver.
4. Test Process System: Testing for heat exchanger Network synthesis and gasoline blending problems.

As can be seen in Figure A.1, the software is a multi-tiered application with separation of all 3 layers: business, presentation and persistence. The back end uses Microsoft SQL server Express 2008, while the front end was developed using web-based technologies like XAML, ASP.NET etc (please refer to figure A.1 for the person to contact). The business logic was mainly written in C#, Microsoft's Object Oriented language. The above mentioned four items were written C#. However, the solvers were all implemented in native languages of C and C++ (please refer figure A.1 for details). Tremendous amount of time was spent in designing the database model, especially for the definition of generalized node models, referred to as Node Templates.

A.2 Database Model

Problems like Heat exchanger Network Synthesis and Multi time Period Gasoline Blending have thousands of equations, variables and constraints. Keeping track of these variables and their values is a mammoth task. Back tracking errors, coefficient accuracies, data input validation is like finding a needle in haystack. Not only a need to generate these equations, variables with easier data validation in an automated fashion was recognized but also the need to manage the models with ease was required. A database was designed to define a generic form of each kind of equation, variable and coefficient that belonged to the node. Streams are defined as belonging to a node's port with each stream carrying a set of physical properties with it called as Stream Property Set. Non-linear equations were handled by solving bilinear terms in different phases called as Calculation Phases or by external function calculation.

A.3 Generalized Node Models (Templates)

In order to define a model every equation, variable and coefficient were identified to belong to a node. The streams in the topology were identified to belong to the node. The position of the stream was defined by the kind of port it was attached to e.g. INLET,COLD_OUTLET etc.

Every stream belonged to two nodes - the start node, from where is

originates and the end node, where it terminates. These two streams are distinguished by an internal connector. So, every stream that originates in a node would terminate in a connector node, and the one that originates in the connector node terminates in a node. The properties of these two streams are connected by the PROPAGATE equations around the connector node, which equates the physical property in the connector nodes incoming stream to its outgoing stream.

A.3.1 Node Templates

Every Node that a user can create has a Node Template attached to it. At a database-level this is a unique integer number and is referred to as Template ID. A node can have more than once Template ID but a node instance can be attached to at most one Template ID.

TEMP_ID	Node Type	Template Description
9	TANK	Single Outlet Tank
16	TANK	Tank with Multiple Outlets
10	BLENDER	Blender with Volumetric inlets.
17	BLENDER	Blender with Fractional inlets.
2	HEAT_EXCHANGER	Heat Exchanger Template

Every Node Template has a Node Equation Template (NET) - defining the set of equations that it contains and a bunch of Node Variable Template (NVT) which are attached to Node Template and Node Equation Template. Each variable or equation is defined by five column properties that

help identify the variable or equation uniquely. This five column definition makes the definition of the RHS of an NET and Coefficient of an NVT to be generically defined. The three Template tables are linked by ID's as illustrated in figure.

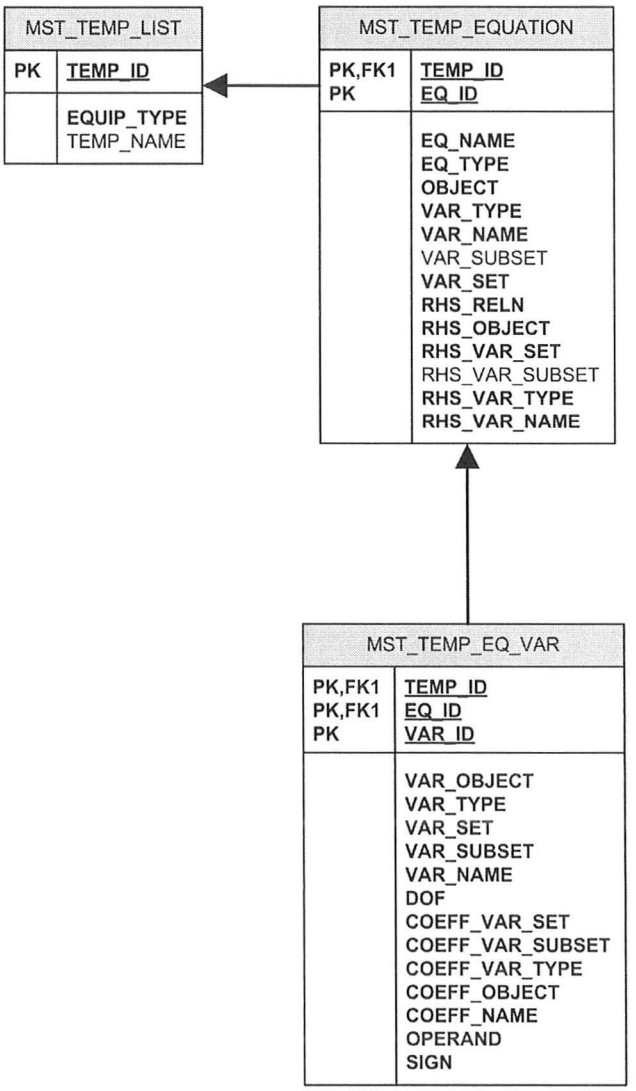

Figure A.2: Relationship amongst Template Tables

A.3.2 Node Equation Templates (NET)

As mentioned previously every equation belongs to the node. A node is attached to a Model ID which is a set of equations of different types. Every equation has a right hand side as well as set of variables and coefficients. A node can have multiple Model ID's with each model consisting different set of equations to solve the node. A user can attach atmost one Model ID to a node instance as well as choose the type of equations from it that he would like to generate and solve. This gives user the flexibility to use already existing models and not go through the pain of redefining them.

A generic equation or Node Equation Template (NET) defines a type of equation and its RHS. NET has the following structure:

Field	Description
TEMP_ID	An integer value uniquely identifying the model number and node to which this equation belongs to.
EQ_ID	An integer value uniquely identifying an equation in a Node Template.
EQ_NAME	Name of the Equation
OBJECT	The object to which this equation belong to. NODE itself or STREAM belonging to the node.

VAR_TYPE	The type of variable to which this equation belongs to. e.g BULK, QVOL. This depends on the VAR_NAME and it's VAR_TYPE defined in the Master Physical Property Table.
VAR_NAME	The Variable or property for which this equation is being defined for. e.g MFLOW, DENSITY, VOLUME etc.
VAR_SUBSET	This is the subset of the set to which this equation belongs to. The use of this column is reserved for future.
VAR_SET	If the OBJECT is STREAM, then this column will identify the port of the node e.g INLET, OUTLET etc. If the OBJECT is NODE, this column will contain SELF indicating that the equation is for node itself.
RHS_RELN	The Relationship of th equation with is RHS. Can be EQ (=), LE ($\leq$), GE ($\geq$) etc.
RHS_VAR_SET	Determining the Variable set of the RHS
RHS_VAR_SUBSET	Variable subset of the RHS.
RHS_VAR_TYPE	Variable type of the RHS.
RHS_NAME	Name of the RHS Variable (whether the RHS value is constant or a variable defined or calculated previously is identified by RHS_OBJECT).

RHS_OBJECT	Same as OBJECT but now it defines the RHS. This value is a SIMPAR (meaning simulation parameter) if the RHS is a constant (like ZERO) or a simulation parameter (like dt, length of each time period). This column can have a value FUNC, indicating that the RHS is calulated by an external function.

A.3.3 Node Variable Templates (NVT)

Every equation defined in an NET for a Node has bunch of Node Variable Templates (NVT's). NVT's are generic definition of variables and coefficients that belong to a given NET. A coefficient can be defined as a simulation parameter, a constant like unity, or as a variable that has been calculated in some previous phase. If an NET is expanded for an instance of a node, the corresponding NVT's are also expanded. Combination of an instance of equation, variable and coefficient, generated from a an NET and its NVT's creates a matrix entry. The structure of an NVT is as follows:

Field	Description
TEMP_ID	An integer value uniquely identifying the model number and node to which this variable belongs to.
EQ_ID	An integer value uniquely identifying this NVT's NET.
VAR_ID	An integer value uniquely identifying the NVT in a (Node Template, NET).
OBJECT	The object to which this NVT belongS to. NODE itself or STREAM belonging to the node.
VAR_TYPE	The type of variable to which this NVT belongs to. e.g BULK, QVOL. This depends on the VAR_NAME and it's VAR_TYPE defined in the Master Physical Property Table.
VAR_NAME	The Variable or property for which this NVT is being defined for. e.g MFLOW, DENSITY, VOLUME etc.
VAR_SUBSET	This is the subset of the set to which this NVT belongs to. This column was used to identify the time periods in Time period connectivity Equations.
VAR_SET	If the OBJECT is STREAM, then this column will identify the port of the node e.g INLET, OUTLET etc. If the OBJECT is NODE, this column will contain SELF indicating that the equation is for node itself.

DOF	Degree of Freedom. Identifies if this variable is fixed (FIXED) or calculated (CALC).
COEFF_VAR_SET	Determining the Variable set of the Coefficient.
COEFF_VAR_SUBSET	Variable subset of the Coefficient.
COEFF_VAR_TYPE	Variable type of the Coefficient.
COEFF_OBJECT	Same as OBJECT but now it defines the Coefficient. This value is a SIMPAR (meaning simulation parameter) if the RHS is a constant (like ONE) or a simulation parameter (like dt, length of each time period).
COEFF_NAME	Name of the Coefficient Variable (whether the Coefficient value is constant or a variable defined or calculated previously is identified by COEFF_OBJECT).
OPERAND	The binary operation to perform between the variable and the coefficient (e.g. MULT for multiplication)
SIGN	The sign of the Coefficient. Can be 1 or -1.

A.4 User Input

Before invoking the engine to generate equations, variables and matrices, user needs to define the process system. A complete set of process system is definition of six tables. An internally generated, Process System ID is assigned whenever a new process System ID is created. This is the unique

identifier that links the information of the process system spread across the tables in the database.

The user needs to begin by defining the topology, models to use for the node, initial parameters and Stream Property Sets. The information that needs to be provided is explained the subsequent sub-sections.

A.4.1 NET_NODE_MODEL and NET_TOPOLOGY

The Nodes and Streams form the terminals and arcs for a process system topology. Every node and stream is defined by model and property sets respectively. Node Models are the Templates discussed in previous sections; while the Property Sets describe the physical characteristics of the stream.

Nodes

Every node is defined by its name and the Template ID to use. This information is populated in the NET_NODE_MODEL table. Every node model in this table has parameters (e.g. Capital Cost, Area, Volume etc.) associated with it. This information is stored in the AMST_EQUIP_PARAM table along with the process system's initial conditions (input data), if any.

Streams

Each stream is uniquely defined by its name and its start node and end node. Each stream has physical properties associated with it - density, temperature, specific heat, flow, volatility etc. A set is defined by the user that can be customized by adding or removing the process system's

relevant or irrelevant properties to the stream. Such a set is called as Stream Property Set. A Stream with its name, terminal nodes and the property set to which it belongs is stored in the NET_TOPOLOGY. The elements (i.e the physical properties) of this property set are defined in the AMST_PROP_SET.

A.4.2 Calculation Phases

The user chooses from the NT the set of all the equations the process system needs to solve to solve. From these models he can filter equations that he will be using. The order of how these equations are to be evaluated is also important. Say, user would like to solve MASS_BALANCE equations before the ENERGY_BALANCE equations since the coefficient in a later phase is a variable that needs to be calculated before, in the earlier phase. Both of these information is supplied in the AMST_CALC_PHASE table, where each EQUATION_TYPE has a number 1..n associated with it - the iteration in which they are to be solved. More than one type of equations can be solved in one phase. (visa-a-versa is not possible).

A.4.3 Simulation Parameters

Any simulation parameters that the Process System has are defined in AMST_SIM_PAR.

Further filtering of the stream equations can be done by omitting the properties in the Stream Property Set. For example a model may have stream equations for temperature and density. However, if a process sys-

tem is not concerned about the density of the stream, any equations with VAR_NAME density will not be generated since the property set of the stream does not contain density. This is true even if the equation type of such an equation is a calculation phase in the system.

A.5 Matrix Generating Engine

After all the necessary information is filled up the engine will generate instances of equations, variables and matrices for each node and its streams. This information is stored in the NET_EQUATIONS, NET_VARS and NET_MATRIX tables. This makes the entire model persistent and experimenting with varying numerical values is only the matter of updating the AMST_EQUIP_PARAM.

A.6 Solver File Format

The information is shared with the native solvers in an ASCII File Format. This information consists of Matrix A, RHS b and the variable vector X in the equation $AX = b$. All the variables are given away to the solver, with their initial values, and bounds. The solver updates these values after each phase of evaluation before returning the final ASCII solution file to be stored in the database for analysis. In all five file ASCII Files are sent to the solver. The names and formats of these files are shown in figure A.3. The explanation of some of the fields is in the table below. The entire process

of generating matrices and solving them using this software is summarized in figure 1.4.

Field Name	Description
RUNID	The run number.
GLOBAL VAR NO	Global Variable Number. This is unique identifier of a variable over the entire process system (i.e. 1..n phases)
LOCAL VAR NO	Local Variable Number. This is a unique identifier of a variable for the given phase. This identifies the column of a matrix.
GLOBAL EQ NO	Global Equation Number - a unique identifier for an equation over the entire process system.
LOCAL EQ NO	Local Equation Number - unique identifier of an equation (row number of a matrix) for the given phase.
PHASE NO	The calculation phase to which the equation or matrix entry ([equation, variable, coefficient]) belongs.

RHS VALUE TYPE	This is used to interpret the value column in the rhs.aeg file. If this column has "0", the value column defines the RHS value itself. If if is $\succ 0$, the value of the RHS is is value of the variable with that global variable number. If it is a string then the RHS value is calculated by a function of similar name.
COEFF VALUE TYPE	This column of abig.aeg, has similar meaning for coefficients as the RHS VALUE TYPE has for the RHS.

xbig.aeg & xbase.aeg							← File Name
INT	INT	DOUBLE	DOUBLE	DOUBLE	DOUBLE	DOUBLE	← Data Type
RUNID	GLOBAL VAR NO.	VALUE	SOFT LOWER BOUND	SOFT UPPER BOUND	HARD LOWER BOUND	HARD UPPER BOUND	← ASCII File Field

xbase.aeg and *xbig.aeg* contain the initial values of the variables The values of variables in *xbig.aeg* are updated after every phase is solved. The values of the co-efficient are also obtained from this file(s).

rhs.aeg							← File Name
INT	INT	INT	INT	DOUBLE	STRING	STRING(2)	← Data Type
RUNID	GLOBAL EQ NO.	LOCAL EQ NO.	PHASE NO.	RHS VALUE	RHS VALUE TYPE	RELATION	← ASCII File Field

This file is used to form the RHS vector, *b*

abig.aeg											← File Name
INT	INT	INT	INT	INT	INT	DOUBLE	CHAR	SIGNED INT	BOOL	STRING	← Data Type
RUNID	LOCAL EQ NO.	GLOBAL EQ NO.	PHASE NO.	LOCAL VAR NO.	GLOBAL VAR NO.	VALUE	OPERAND	SIGN	DOF	COEFF VALUE TYPE	← ASCII File Field

This file is used to form the LHS matrix, *A*

eqlist.aeg			← File Name
INT	STRING	STRING	← Data Type
GLOBAL EQ NO.	EQUATION NAME	PROPERTY NAME	← ASCII File Field

Data Types
DOUBLE: Double Precision Floating Point
INT: Unsigned Integer
CHAR: Character
SIGNED INT: Signed Integer
BOOL: Boolean (0 or 1)

This file has the list of all the equation names and is used for debug purpose.

varlist.aeg			← File Name
INT	STRING	STRING	← Data Type
GLOBAL VAR NO.	VARIABLE NAME	NODE NAME	← ASCII File Field

This file has the list of all the variable names and is used for debug purpose.

Figure A.3: ASCII File Specification for Solver Interface

Appendix B

List of Variables

Variable	Description
I	Total qualities
G	Total Products
K	Total Components
T	Total Time Periods
dt	Length of time period
$V_{in,k,j,t}$	Incoming Volumetric flow for blender g from component k in time t
$V_{out,k,j,t}$	Outgoing Volumetric flow for component k in blender g in time t
C_k	Unit cost of component k
$V_{open,g,t}$	Opening Volume of blender g in time t
$V_{out,g,t}$	Outgoing volume for blender g at time t
$V_{close,g,t}$	Closing Volume of blender g in time t

$Q_{open,i,g,t}$	Opening quality i of blender g in time t
$V_{blend,g,t}$	Volume blended in time t in blender g
$f_{k,g,t}$	Fraction of Component k used to blend product in blender g in time t
$Q_{i,g,t}$	Calculated quality i of product g in time t
$F_{out,g,t}$	Demand for product g in time t
$F_{in,k,t}$	Supply for component k in time t
$Q_{i,k,t}$	Quality i for component k in time t (assumed constant for all time periods)
$V_{close,k,t}$	Closing inventory of component k in time t
$V_{open,k,t}$	Opening volume of component tank k in time t
$V_{k,min}$	Minimum volume specification for tank with component k
$V_{k,max}$	Maximum volume specification for tank with component k
$Q_{i,g,min}$	Minimum specification for quality i for product in blender g
$Q_{i,g,max}$	Maximum specification for quality i for product in blender g

Appendix C

List of Abbreviations

Abbreviation	Description
ALK	Alkylate
BUT	Butane
REF	Reformate
HCL	hcl
HCN	Hydrocarbons
LNAP	Light Naphtha
LCN	Light Cracked Naphta
RON	Research Octane Number
MON	Motor Octane Number
ARO	Aromatic
BEN	Benzene

SUL	Sulfur
SPG	Specific Gravity
OLE	Olefin
RVI	Reid Vapor Pressure
AEG	Auto-Equation Generator
DE	Differential Evolution
LP	Linear Programming
EP	Embarrassingly Parallel
SMP	Symmetric Multi Processor
UMA	Uniform Memory Access
NUMA	Non-uniform memory access
cc-NUMA	cache coherent Non-uniform memory access
MPI	Message Passing Interface
TLS	Thread Local Storage
POSIX	Portable Operating System Interface for Unix
GLPK	GNU Linear Programming Kit
NT	Node Template
NET	Node Equation Template
NVT	Node Variable Template
ASCII	American Standard Code for Information Interchange

List of Figures

List of Tables

List of Algorithms