MULTI-ACTIVITY TOUR SCHEDULING PROBLEMS

A LARGE-SCALE NEIGHBORHOOD SEARCH ALGORITHM FOR MULTI-ACTIVITY TOUR SCHEDULING PROBLEMS

By RANA SHARIAT, B.Sc., M.Sc.

A Thesis Submitted to the School of Graduate Studies in the Partial Fulfillment of the Requirements for the Degree Master of Science

McMaster University

Master of Science (2022)

Hamilton, Ontario (School of Computational Science & Engineering)

TITLE: A LARGE-SCALE NEIGHBORHOOD SEARCH ALGORITHM FOR MULTI-ACTIVITY TOUR SCHEDULING PROBLEMS

AUTHOR: RANA SHARIAT (McMaster University)

SUPERVISOR: Dr. Kai Huang

NUMBER OF PAGES: x, 72

Abstract

In this research, we study multi-activity tour scheduling problems with heterogeneous employees in a service sector where demand varies greatly during the day. This problem relates to assigning the employees' working days throughout the planning horizon and working periods on each working day. In each period, an activity type is given which requires certain skill of the employees. The goal is to reduce the overall over- and under-coverage for each period and activity. The shifts and breaks defined with variable starting slots and durations make the problem flexible and hard to solve. In order to address the problem, an integer programming (IP) approach is first proposed. Due to the problems' high degree of flexibility, it is impossible to solve instances involving numerous employees and activities in a timely and efficient manner. This leads to the proposal of a heuristic method based on a large neighborhood search (LNS) algorithm. We first create the daily schedules by a context-free grammar (CFG). Then we solve a resource-constrained shortest path problem (RCSPP) to create weekly schedules. Heuristic search is performed on weekly schedules. Moreover, when a constraint on task repetition is added, a CFG is unable to express this constraint, so we incorporate an extension of the IP into our proposed algorithm. Importantly, our approach does not use any closed-source commercial solver like CPLEX. Computational experiments are carried out on the industrial and randomly generated instances to evaluate the performance of the IP solved by CPLEX and the proposed algorithm. Results reveal that our method outperforms CPLEX in both solution time and solution quality in larger instances.

To brave people of Iran who fight for freedom

Acknowledgements

I want to start by expressing my gratitude to my supervisor, Dr. Kai Huang, for giving me a chance to work in this research and for all of his support over the past two years.

Additionally, I want to thank my parents and brother, who have always been there for me, supported my choices, and helped me on the road to success.

Last but not least, I want to express my deep appreciation to my loving spouse Amir, who has always supported me through challenging circumstances with his affection.

Contents

Abstract **iii**

Acknowledgements **v**

Acronyms **x**

1 Introduction **1**

2 Literature Review **4**

2.1 Shift Scheduling 4

2.1.1 Exact Methods 4

2.1.2 Heuristic Methods 7

2.2 Tour Scheduling 11

2.2.1 Exact Methods 12

2.2.2 Heuristic Methods 14

2.3 Contributions 18

3 Model Development **20**

3.1 Problem Setting 20

3.2 Tour Scheduling Integer Program 20

4 Solution Methodology **25**

4.1 Context-Free Grammar 25

4.2 Daily Schedules 29

4.3 Weekly Schedules 32

4.4 Large Neighborhood Search 36

4.5 Improvements . . . 39
4.5.1 Starting Slots Selection . . . 39
4.5.2 Daily Schedules Selection . . . 40

5 Repetition Constraints **41**

6 Computational Experiments **45**
6.1 Industrial Instances . . . 45
6.2 Random Instances . . . 47
6.2.1 Low Daily Flexibility . . . 48
6.2.2 High Daily Flexibility . . . 50

7 Conclusions and Future Research Directions **53**
7.1 Summary . . . 53
7.2 Future Research Directions . . . 54

A Chapter 3 Supplement **56**

B Chapter 3 Supplement **59**

C Chapter 4 Supplement **61**

D Chapter 5 Supplement **63**

E Chapter 5 Supplement **66**

Bibliography **68**

List of Figures

4.1 The weekly scheduling DAG 33

List of Tables

3.1 Parameters for each enumerated 4-hour shift 21
3.2 Sets, indices, parameters, and decision variables for the IP model 22

5.1 Sets, indices, parameters, and decision variables for the task assignment IP . . . 43

6.1 Parameters of industrial instances 46
6.2 Running times (in seconds) & objective cost of the proposed algorithm and CPLEX solver 47
6.3 Randomly generated low daily flexibility instances 49
6.4 Running times (in seconds) of the proposed algorithm and CPLEX solver for the low daily flexibility random instances 49
6.5 Objective cost of the proposed algorithm and CPLEX solver for the low daily flexibility random instances 50
6.6 Randomly generated high daily flexibility instances 50
6.7 Running times (in seconds) of the proposed algorithm and CPLEX solver for the high daily flexibility random instances 51
6.8 Objective cost of the proposed algorithm and CPLEX solver for the high daily flexibility random instances 52

A2.1 Parameters for each enumerated shift for each duration between 8 to 16 periods and each start period 60
A2.2 Sets and parameters for the IP model 60

A5.1 Sets, indices, parameters, and decision variables for the task assignment IP . . . 67

Acronyms

B&P branch-and-price

CFG context-free grammar

CG column generation

IP integer programming

LNS large neighborhood search

MIP mixed integer programming

RCSPP resource-constrained shortest path problem

Chapter 1

Introduction

Organizations in the service industry, like retail sectors or restaurants, usually are open 24 hours and seven days a week, with a wide fluctuation in demand throughout the day. Hence, workforce scheduling is critical and keeps a service organization competitive since it directly impacts its profit and customer service quality. An efficient schedule could reduce the over or under-coverage problem in these environments. However, efficient scheduling could be computationally expensive since the schedules should be flexible due to various types of workers with different skill sets, availabilities, and the highly flexible environment. The flexibility is reflected by shift types, shift start time, shift end time, break placements, assigned activities, etc.

Baker (1976) has suggested a classification for workforce scheduling problems. Days-off scheduling, shift scheduling, and tour scheduling are the three main categories under this division. The days-off scheduling involves choosing the working days and rest days throughout the planning horizon. Shift scheduling decides which working periods to give each employee during the working day. In tour scheduling, other than rules on shifts, constraints related to the planning horizon are included. Based on these rules, the working days and working periods of each working day for each employee are determined. Hence, days-off and shift scheduling are combined in tour scheduling.

In this research, we study a multi-activity tour scheduling problem for heterogeneous employees in the service industry like retail sectors or restaurants. Due to the mentioned properties of these environments flexible shifts need to be considered. The hierarchical structure of schedules

in this study are as following. A task is defined as consecutive periods that an activity is being performed. Combination of tasks builds a timeslot. Two timeslots with a break in between build a daily schedule. Finally, daily schedules and off days are combined to make weekly schedules. First, we develop an exact integer programming (IP) model. The large size of the problem and high computational cost is why we develop a heuristic method. The objectives are to develop an algorithm in which constraints could be easily added or removed. Also, using any commercial solvers should be avoided.

In Chapter 2, a study of the literature is conducted on the exact and heuristic methods developed for the service sectors' shift and tour scheduling problems. In Section 2.3, we have shown our contributions in detail by comparing our proposed method to the most relevant studies in the literature.

In Chapter 3 we first define the problem setting studied in this research. We have bounds on the shift length, activity duration, break placement, and weekly parameters for scheduling shifts, similar to other studies in the literature. However, there are no limits on the timeslot duration or number of tasks each employee can perform at each timeslot. So, the shift is flexible, and the number of feasible schedules is enormous. Also, there are no bounds on the break duration, but the break length is a function of shift length. That is why we have two shift types in our problem. A tour scheduling IP model is proposed for this problem. To include the shift types in the model, we have enumerated shifts. The parameters for each enumerated shift are shift start time, shift end time, and break length.

In Chapter 4, we develop an algorithm to solve the problem described heuristically. This algorithm employs a large neighborhood search (LNS) approach to repeatedly assign weekly schedules to the employees until an optimal local solution is identified. The weekly schedules are generated in two steps to reduce computational time. context-free grammar (CFG) and automaton modeling techniques are widely used in the literature to model problems with constraints on the sequence of decision variables. In our problem setup, employees could have multiple activities in their skill set. Since a CFG is more expressive than an automaton and the computational cost for larger instances is less than an automaton, we chose this method to model the daily schedules. This step consists of two sub-steps. First, we generate feasible timeslots

using CFG modeling techniques and algorithms developed for generating feasible solutions for this technique. Next, we generate daily schedules using a CFG. However, the same algorithms used to generate timeslots cannot be used due to the different structures of the problem. We proposed modifications to the algorithm to build feasible daily schedules. The second step solves a resource-constrained shortest path problem (RCSPP) to generate weekly schedules from the daily schedules as weekly parameters cannot be expressed by a CFG. A bidirectional A* search algorithm is used in the literature to build weekly schedules. We use a new exact technique based on a bidirectional A* search algorithm considering one resource and the upper bound on the resource consumption, which uses several heuristics in the preprocessing and core stages of the method to search faster. However, we modified the algorithm to include multiple resources and lower bounds on resource consumption. Also, due to the layered structure of our weekly scheduling graph, some simplifications are made in the preprocessing step to make the algorithm faster. In addition, two improvement techniques are also suggested to reduce the computational time heuristically for the instances with a larger number of feasible daily schedules.

In addition, a new constraint on task repetition is indicated in Chapter 5, which prohibits workers from performing tasks with similar activities on each working day. This constraint can help to improve the fairness between the employees. Also, for some environments, different activities are associated with different floors of a building. So, changing activities means moving between different floors for several times. When such a constraint is added, the tour scheduling IP formulation and the proposed algorithm must be adjusted. As this constraint cannot be expressed by a CFG, we propose a task assignment IP which uses the solution found by a LNS as an initial solution. Some parts of this solution, such as working days, shift periods, and break periods are extracted as parameters. Then, the task assignment IP assigns activities using an open-source solver.

The computational experiments and results for the tour scheduling IP and the proposed heuristic algorithm are shown in Chapter 6 for both industrial and randomly generated instances. In Chapter 7, a summary of the study, results, and future research directions are included.

Chapter 2

Literature Review

2.1 Shift Scheduling

The specification of each employee's working periods during their working days is called shift scheduling. In the following, we review the exact and heuristic approaches for this problem.

2.1.1 Exact Methods

The shift scheduling problem was first introduced by Edie (1954) for toll booth personnel scheduling. In this study, it is suggested to use probability theory tools to address the issue quantitatively while taking into account service quality and economic goals. Later, Dantzig (1954) proposed using linear IP techniques to address the toll booth problem. The proposed IP formulation has a form of set-covering problem, which assigns potential patterns, or shifts, to each employee in order to meet demand. In this formulation, the binary decision variables determine whether or not an employee is allocated to a shift.

In the set-covering formulation, the possible shifts are enumerated explicitly. However, a complete enumeration of shifts is infeasible due to the astronomical number of shifts when model flexibility in terms of shift start, length, break time, and break placement increases. So, implicit models were proposed to handle problems with high flexibility. Bechtold and Jacobs (1990) study a shift scheduling problem with flexible break times. They were the first to present an implicit integer linear programming model. In this approach, a break decision variable is defined

that shows the overall number of breaks taken by the entire set of employees during a period, as opposed to enumerating all possible combinations of shifts and breaks in the set-covering formulation. The computational results show that, in terms of execution time and computer memory needs, the implicit model outperforms the set covering formulation in more flexible problems. Later, it was demonstrated by Bechtold and Jacobs (1996) that these two formulations are equivalent.

Thompson (1995) integrates the works of Bechtold and Jacobs (1990) on the formulation of implicit flexible breaks and Moondra (1976) on implicitly depicted shifts with flexible starting times and lengths. In the proposed IP model, a set of shifts with the same cost, working periods, break duration, minimum and maximum duration limit, and minimum and maximum limit on the before and after break work durations are referred to as a shift type. The number of shifts of each type starting and ending at each period as well as the number of breaks starting at each period are included as decision variables. The characteristics of each shift type are ensured by various sets of constraints. Compared to Bechtold and Jacobs (1990), this approach can generate more shift alternatives and find the optimal solutions faster.

Aykin (1996) proposed an implicit shift scheduling formulation with various breaks and break windows. The decision variables in this method are comparable to those suggested by Bechtold and Jacobs (1996), but two rest breaks and one lunch break are permitted. Later, Aykin (1998) added heuristic improvements to the exact method. To optimally solve the largest examples addressed up to that time, they devise a branch-and-cut algorithm in which upper bounds are developed for the variables, and a rounding heuristic is employed.

Although the set-covering model has limitations for large size problems with many activities, diverse employees, and high levels of flexibility, a B&P method was proposed by Mehrotra et al. (2000) to overcome this issue by generating only a small fraction of feasible shifts. The set-covering formulation of the shift scheduling problem, which included minimum and maximum restrictions on the number of employees and breaks at each period as well as different windows for rest, meal, and break times, is proposed in this research. A column generation (CG) algorithm is designed to solve this problem. The computational results show that this approach can reduce computational time significantly compared to the implicit models.

The presented works solve shift scheduling problems, in which working and break periods are assigned to each employee. However, the following exact works take into account multi-activity shift scheduling problems. In these problems, besides break and working periods, the activity carried out in each period must be determined.

As it is hard to express problems with constraints on the sequence of the decision variables using mixed integer programming (MIP), constraint programming techniques are proposed, which provide a simpler way to express complex constraints. Demassey et al. (2005) presented a CG method based on constraint programming using regular language, which is recognized by an automaton. This method's capacity to tackle different problems while avoiding significant changes to the algorithm itself is what makes it interesting. They used Dantzig (1954)'s formulation to represent the problem, in which all feasible shifts are created, and the optimal set is chosen. A cost-regular constraint is proposed to generate only negative reduced cost schedules. Constraint satisfaction techniques. This research examines three problems: First, the homogeneous employees with objective cost equal to the sum of the schedules; second, considering over coverage and under coverage; and third, heterogeneous employees considering preferences. The method is implemented for the first problem type.

Côté et al. (2011a) suggested a constraint programming-based method using a CFG and automaton. They propose two MIP models. One model uses the automaton, while the other uses a CFG to produce graphs of feasible sequences. In the numerical experiments for multi-activity shift scheduling problems, it is shown that both models reduce computational time and simplify modeling when compared to the compact MIP formulation. Later, Côté et al. (2011b) suggested a new implicit shift scheduling model when symmetry is brought in by homogeneous employees. Although this work solves large-scale multi-activity problem instances, it cannot handle personalized scheduling problems. For addressing this gap, Côté et al. (2013) employed the B&P algorithm to the classical set covering formulation. The pricing subproblem employs a CFG to produce feasible schedules for each employee and is solved with dynamic programming by traversing the grammar graph. Large-scale examples can be solved using this method, and the adaptability of CFG allows for the modeling of various scheduling problems. However, when the number of total working periods expands more than one day, this method will become inefficient.

Boyer et al. (2014) proposed an extension to the B&P method proposed by Côté et al. (2013). They investigate a shift scheduling problem with heterogeneous employees with various skills and availability. In addition to activities, they have defined uninterruptable tasks with fixed lengths and hard constraints on their sequences and completion time. For the restriction on the sequence of the tasks, two formulations are suggested. Similarly, a set covering model is proposed and a CFG generates feasible shifts with complex constraints like task time windows, break placement, and continuous working period length in the pricing problem. For the case studies that are based on real cases, three branching strategies are offered. Employees are scheduled using two different timetables: fixed working hours over a week and flexible hours for one day.

2.1.2 Heuristic Methods

When the size or the flexibility level of the scheduling problems increase, the corresponding decision variables and constraints will increase, which makes the scheduling problems more difficult to solve. In many cases, the exact methods can't solve realistic instances. Therefore, to solve realistic instances, heuristic methods based on local search, LNS, and relaxations are also very popular in solving scheduling problems. Moreover, meta-heuristics based on CG and B&P methods are proposed.

Shift design problems are also studied in the literature, which entails determining the start time, duration, and staffing level of each working shift within a company. Musliu et al. (2004) suggested a shift design method with two steps. In the first step, the shifts are designed. Next, employees are assigned to shifts. They use this two-stage method since it is easier to solve the problem in several stages. One of the objectives of the problem is to minimize the number of shifts which is NP hard. So, the local search method is used to solve this problem. Various shift types with minimum and maximum bounds on shift start and length for each shift type are considered. For each solution, neighborhoods are defined using several move types. Tabu search is also incorporated into the move type selection procedure to prevent search cycles. Moreover, moves are ranked according to their level of promise to avoid investigating the entire neighborhood. As a result, the neighborhood search for the current solution is interrupted each time a better incumbent solution is discovered. Additionally, depending on the requirements and

the structure of shift types, an algorithm is developed to determine an initial solution.

An algorithm utilizing local search operators and constraint programming techniques was employed by Gaspero et al. (2010) to address an extension of the shift design problem. Break scheduling is also included in this problem. Different shift types are defined in this study with restrictions on the earliest and latest start time and length. The first step involves designing shifts and figuring out how many staff should be assigned to each using the hill-climbing local search method. Following that, constraint programming is used to set breaks, including lunch breaks, if the shift duration is long enough, with the earliest and latest start time and length limitations. Additionally, Gaspero et al. (2013) reviewed the literature on shift design and break scheduling problems and the approaches for solving them. Further, two strategies based on local search methods are explained in depth, and one is used in a real-world case study.

Some researchers propose their own heuristic methods based on the problem structure. To set weekly work schedules in a chain of retail clothing stores, Pastor and Olivella (2008) presented a heuristic method including working time accounts and minimum and desired level of employee capacity for each period in the problem. Although a weekly scheduling problem is studied, this method cannot be regarded as a tour scheduling method because it does not consider days-off scheduling. This approach introduces working time accounts, which track each employee's weekly working hours. Considering this balance, each employee may owe working hours to the company or the other way around. Additionally, the minimum and desired number of staff is considered for each period. There are two phases in the solution strategy. Each employee receives a schedule based on a set of feasible schedules in the first phase using a mixed integer linear program. If solving the problem takes longer than the allotted time, the answer quality is improved through local optimization utilizing the neighborhood search method. The schedules are changed in the second phase using established heuristic algorithms to reduce surpluses and shortages by reducing or lengthening working hours.

Next, we review the heuristic methods to solve multi-activity shift scheduling problems. Meisels and Schaerf (2003) investigated a multi-activity shift scheduling problem which has various shift types with fixed beginning and ending times using a local search method. Soft constraints on distributing assignments evenly among staff and preferences for shifts are also

incorporated. They first provide a model of constraint programming. Then, a local research method based on the hill-climbing strategy is proposed. The four hill climbing techniques that have been suggested vary in the area of the neighborhood that they examine, from analyzing a single neighbor chosen randomly to examining the entire neighborhood. The search space is then expanded by defining two more move types. Whether the requirements are met or not, this new search space encompasses all potential assignments. The cost function has been altered by giving different parts of the function modifiable weights to explore the larger search space efficiently. According to the computational results, the extended search space method performed better when compared to other local search methods.

Bonutti et al. (2017) addressed call centers and supermarkets' multi-activity shift design problem based on local search. The proposed method, differently from Gaspero et al. (2013) uses a single-stage simulated annealing method to search the solution space. The search method core is a composite neighborhood that simultaneously modifies the shift staffing, shift shape, and break position. There are different types of shifts described, and each type has unique characteristics including minimum and maximum length, earliest and latest start times, planning days that it can be used on, and break presence. Some break-related parameters are also given if the shift type has one. The first step is to generate shifts at random for each shift type to meet the requirements of each period. Next, five moves are introduced to define the neighborhoods. To enhance the solution, the moves are chosen using a simulated annealing method.

Demassey et al. (2006) extended their previous work (Demassey et al. (2005)) by proposing a B&P method. They propose a new cost-regular optimization constraint that directs the search heuristically in the pricing problem by offering upper and lower bounds. The shortest and longest paths in a layered directed graph, where the layers correspond to the states of the automaton, are employed to compute these bounds. The computational results on the generated instances showed that the method could solve scheduling problems with less than three activities in 2 hours. The approach did not converge in a reasonable amount of time for the data from the real-world case.

Bhulai et al. (2008) suggested a new two-step heuristic method for allocating shifts in multi-skill call centers by including agent groups. Depending on their skills, the employees can work

in various agent groups. The first phase involves determining the staffing requirements for each agent group for each period. The optimal number of shifts for each type is chosen in the second stage. The best solution is chosen using an IP approach. Then, a developed heuristic algorithm relying on a linear programming model assigns the agents to the shifts at each period. The computational outcomes demonstrate that the problems with two to five skills are not numerically challenging and can be solved using this approach.

In addition to local search techniques, LNS has been used in the literature for shift scheduling problems. The reason is that larger neighborhoods reduce the likelihood of becoming trapped in a local minimum. Quimper and Rousseau (2010) proposed two algorithms that use regular and context-free languages to describe the complicated realities of shift scheduling problems. In the first method, all the possible sequences are formulated using regular languages. Then, the expanded automaton graph is used to generate feasible schedules. In the second method, a CFG model is suggested, which encodes richer languages than an automaton in the sense that a CFG can recognize any regular language, but the reverse is not true. In this method, parsing trees from the grammar graph show the possible schedules. The CYK (Cocke (1969) Kasami (1966) Younger (1967)) parser algorithm is used to build a graph embedding all the parsing trees in the grammar. In order to extract the optimum cost parsing tree from the graph, they also develop a cost-assigning method. Finally, a LNS algorithm is designed to assign the best-generated schedule using the above techniques to each employee in a round-robin way. During each iteration of the search, the destructor operator removes the schedule for an employee, and the constructor operator finds the optimal schedule based on the schedules assigned to the other ones. Up until an optimal local solution is discovered, the algorithm keeps running. The initial solution is a random assignment of feasible schedules to the employees. Both methods are used to solve scheduling instances. Each time a local optimum is reached for an instance, it is restarted using a new initial solution. The computation results demonstrate that as the number of activities increases beyond 3, the CFG method's computing time becomes less than the regular method. For further comparison, a one-activity shift scheduling problem using the grammar method is evaluated with an exact MIP model proposed previously in the literature. CPLEX is used to solve the MIP, and the process ends after one hour of computation or a 1% gap. The findings demonstrate that the suggested approach discovers solutions at 1% of the optimal solution in

less than 25 seconds for all the instances and that the proposed method's minimum cost is lower or equal to the MIP's minimum cost.

Restrepo et al. (2012) present a heuristic CG technique for shift scheduling problems that involve multiple activities, breaks, variable shift lengths, and over time. In this approach, the set covering problem, which determines employee shift assignments, is solved by the master problem. The CG is used to solve the linear relaxation of the master problem. Then, an integer solution is obtained by solving the reduced master problem under the integrality constraint. The set of possible shifts is generated in the pricing problem. In this pricing problem, a feasible shift is presented as a path in a graph. So, the pricing problem is modeled as an RCSPP. Some of the most difficult constraints are included in the structure of the built graph.

Hernández-Leandro et al. (2019) studied a multi-activity shift scheduling problem with heterogeneous employees and established constraints on the earliest start time and latest finish time for each activity. Lagrangian relaxation is used in the suggested meta-heuristic method to obtain a selection of promising shifts. Specifically, the objective function includes the relaxed demand restrictions, and the subgradient method is used to solve the Lagrangian dual. A feasible solution is found by solving the restricted set covering problem. This approach uses a CFG to represent feasible shifts. In many cases, the computational results are superior to those obtained using Côté et al. (2013)'s B&P approach. Additionally, the method's processing time is generally significantly faster.

2.2 Tour Scheduling

Tour scheduling, where the working days during a time horizon and the working periods for each working day are determined, combines days-off and shift scheduling. The exact and heuristic approaches to this problem will next be discussed.

2.2.1 Exact Methods

Mathematical programming approaches such as IP and MIP models are also used for tour scheduling problems. The shift start time bands, which include potential start times, were first introduced by Jacobs and Brusco (1996). They solve the tour scheduling problem optimally utilizing an implicit compact IP model that uses shifts and days-on variables. Toll collector scheduling has been done using this model. The results demonstrated a significant improvement in scheduling efficiency compared to the scenario where personnel begins at the same time on each day of their tour.

Brusco and Jacobs (2000) integrate meal-break windows into the tour scheduling problem in addition to starting time bands. The implicit IP approach suggested in this paper uses shifts, working days, and breaks as decision variables. The shift and break variables are connected by applying the same constraints Bechtold and Jacobs (1990) suggested.

Larger linear programming models can be solved to optimality by using B&P methods which make use of CG algorithm. Brunner and Bard (2013) discusses the problem of service workers' flexible shift scheduling at postal processing and distribution centers. Different shift starting times, lengths, lunch break allowances, and the day off assignments are all part of the schedule flexibility considered in this paper. The problem is modeled as a set-covering MIP to solve the linear relaxation of the problem. The B&P method is also utilized to obtain integer solutions. The master problem is a set covering formulation assigning tours associated with each employee type (regular or flexible). Additionally, two different subproblems for both regular and flexible workers are presented. Unlike the flexible workers, the shifts are explicitly generated for the regular workers. According to computational findings, adding more flexibility to the problem can result in significant cost reductions.

The exact approaches to multi-activity tour scheduling problems are next reviewed. Regarding problems with hierarchical workforce tour scheduling with off days, Seçkiner et al. (2007) presented an IP approach. Employees have varying levels of qualification, as seen by the hierarchical structure. The staff members who are the most qualified can execute each activity, whereas the staff members who are the least competent can only perform one. This model is an

expansion of Billionnet (1999)'s, which only allows for the assignment of employees to one shift type; in contrast, this model allows for assigning employees to various shift types. Additionally, based on how many working days are allotted to them, the number of off days is determined.

Kabak et al. (2008) suggested an MIP model minimizing the number of staff required in a retail sector while meeting the demand. To accomplish this goal, they employ a two-stage strategy. An hourly staff requirement is initially calculated using historical data and a sales response model. The second stage involves assigning full-time and part-time workers to a predetermined set of daily shifts with fixed start times, durations, and break times using an explicit MIP model. Additionally, it is presumed that every employee can carry out every activity. Meeting the standards outlined in the union contract regarding the upper bound for the part-time staff to full-time staff ratio and limits for each staff member's working days and hours per week are also included in this model. In addition, a simulation is run to validate and improve the output of the sales response model.

Jones and Nolde (2013) presented an MIP multi-activity workforce scheduling model for the Swiss Migros market for the heterogeneous workforce. The duration of each activity is also constrained in this model. Activity shifts are proposed to be enumerated to include these limits, where an activity shift is a set of consecutive periods where a specific activity is performed. Employees are then assigned to these activity shifts using the MIP. Additionally, there are minimum and maximum thresholds for employee numbers in each period, the length of breaks (breaks are considered activities in the model setup), working days, and uninterrupted work hours. This model aims to reduce the sum of squared errors between the expected and assigned staff numbers for each period. The objective function is approximated with a convex piecewise affine function to convert the problem into an MIP model. Some constraints were softened to achieve a feasible schedule and avoid infeasibility. Studies on computational scheduling at Migros stores with 16 employees and 3 activities revealed that it takes roughly 5 to 10 minutes to address instances with 1-hour periods, a horizon of 7 days, and 4 to 20 employees. However, the same instances with 30-minute periods were resolved in 10 to 25 minutes.

Besides decomposition methods like B&P, Benders decomposition is also proposed to solve scheduling problems. Restrepo et al. (2018) used a strategy that combines Benders decomposition

and CG to solve the multi-activity tour scheduling problem for homogeneous employees. The subproblems in Benders decomposition method allocate activities and break periods to produce daily shifts. To represent the rules in generating shifts in the subproblems, a CFG is used. Then, daily shifts and tour patterns are connected by the master problem. The master problem is solved by CG since there may be an excessive number of shifts if fully enumerated. According to the computational results, high-quality solutions can be developed for instances with up to ten activities, and the strategy performs better than a B&P approach.

2.2.2 Heuristic Methods

For tour scheduling problems with flexible shift start time, Bailey (1985) suggested a formulation to reduce the overall expense of assigning staff to the shifts and staff under coverage. First, an IP optimization model is used to assign employees to shift and days-off patterns. The start time of the shift is unrestricted. The length of the shift, however, is set at 8 hours. A heuristic method is additionally suggested to set the shift start time and restrict the occurrence of shift patterns with a maximum difference in start timings. Compared to a two-step optimization strategy recommended in the literature, where the days-off problem is handled first, and then seven shift scheduling problems are solved later, the computational findings of this model demonstrate an improvement in objective cost.

Using the set-covering formulation, Easton and Rossin (1991) suggested a methodology to assign predetermined tours to employees. There are two distinct categories of full-time and part-time workers, and their ratio is constrained by a parameter. While part-time employees work shifts of four hours with no break, full-time employees could be given nine-hour shifts beginning at any time of the day with a fixed break. Then, based on CG, they create a heuristic method to generate a subset of possible tours. Two distinct studies including solely full-time workers and both types of the workforce were conducted. In comparison to formulations based on all feasible tours, the method for the first set of experiments could obtain the optimal cost with fewer variables. When all possible tours were constructed, the objective cost for the second method was equivalent to accurate methods and lower than other heuristic methods in the literature.

Additionally, the computational cost was significantly lower than that of the competing alternatives. In addition to the flexibility in the tour scheduling problem that Easton and Rossin (1991) proposed, Brusco and Jacobs (1993) also put up a model with flexibility in the arrangement of off days and meal breaks. Up to 2 billion viable tours were generated by the stated environment. The problem was thus solved using a novel heuristic based on simulated annealing. The results of the experiments demonstrated that near-optimal solutions could be discovered, and the method was very favorably comparable to an IP solution strategy developed for this problem in terms of solution quality and computing time.

Al-Yakoob and Sherali (2008) looked into tour scheduling of the gas station employees while taking into account their preferences for shifts, off days, and work locations. Three staff classifications are present in this issue, and three shifts covering the day are specified. For solving the linear relaxation of the problem's set-partitioning formulation, a CG approach is suggested. The subproblem includes restrictions on the number of consecutive working days and shifts assigned on consecutive days. A sequential variable-fixing heuristic is suggested to produce a feasible schedule for each employee.

Exploiting variable neighborhood search technique, Talarico and Duque (2015) offered two personnel management strategies—exact and heuristic—for an Italian grocery chain's check-out operation. In these methods, the set of feasible daily shifts is enumerated. The length of the shift, the number of working days, and the number of off days are all constrained in this problem. In addition, there must be at least one break for each daily shift and a 2-hour break after every 6 straight hours of work. In order to assign the workers to a set of daily shifts during the week, an exact set-covering formulation is first proposed. Then, a three-phase meta-heuristic method that combines an exact and heuristic approach is developed. In the first phase, which is an initial solution generator, only a subset of feasible weekly shifts is generated and a set covering model assigns shifts to employees. A variable neighbourhood descent approach is employed in the second phase to improve the initial solution using 4 operators. Two techniques are utilized in phase three as diversification strategies to avoid the local optimum. The outcomes of the meta-heuristic algorithm are compared with the exact method solution provided by CPLEX. The average optimality gap was 1.3% compared to CPLEX, but the computing time was reduced by

35.4%.

Next, an overview of heuristic solutions to multi-activity tour scheduling problems is provided. Detienne et al. (2009) offered two IP models for employee timetabling to satisfy the requirements at the lowest possible cost. For each employee a set of feasible work pattern is given. The first model is a set-covering problem and workers are assigned to the predetermined work patterns. The second model is a multi-dimensional, multi-choice knapsack formulation. A Lagrangian lower bound, a heuristic approaches based on cut generation, and an exact method based on Benders decomposition are proposed. The master problem in the exact approach is a multi-dimensional, multi-choice knapsack problem and the sub-problem is a b-matching problem.

To address the scheduling of heterogeneous part-time service employees, Hojati and Patil (2011) proposed a heuristic method based on a two-stage IP method according to the problem structure. Different employee availability possibilities and weekly work restriction considerations are addressed in the problem setting. The goal is to reduce overstaffing and ensure that employees complete the required number of hours of work each week. There are two steps in the suggested heuristic method. An IP model is used to select suitable shifts that satisfy employee needs in the first step. Afterward, a heuristic based on IP is used to assign the shifts to the employees. The computational results showed that the method is robust and can find close-to-optimal solutions.

A more flexible tour scheduling problem with flexible shifts is studied by Qu and Curtois (2020). They presented a variable neighborhood search method for a multi-activity tour scheduling problem. In this problem, there are limits on the maximum consecutive working days, minimum and maximum working periods and shift length, minimum rest time between the shifts and activity duration, and valid shift start times. To enhance the solution, four distinct neighborhood operators are defined.

To schedule multi-activity tours for heterogeneous staff, Restrepo et al. (2016) presented a B&P method. The problem setting includes adjustable shift start times, lengths, breaks, and day off patterns. Based on the Dantzig-Wolfe decomposition, two formulations are suggested. In the first formulation, columns represent daily shifts produced by a CFG, and tours are produced in the master problem by taking into account the weekly regulations. At each node of the B&P

algorithm, the employee holding the largest fractional values is chosen as branching rule imposes. The second formulation employs tours as columns, which are generated in the subproblems in two steps. The RCSPP is used to construct tours once daily shifts are generated using a CFG. This formulation's branching rule combines the first formulation's rule with an aggressive variable-fixing method. The findings demonstrate that for all instances with an optimality gap smaller than 1%, the second formulation is stronger in terms of its lower bound and high-quality solutions.

Gérard et al. (2016) studied a flexible multi-activity tour scheduling problem. Due to the problem setting and larger time horizon, the B&P method proposed by Restrepo et al. (2016) cannot be used, and instead, heuristic versions of B&P method based on the nested dynamic program are proposed. This problem includes days-off scheduling, shift scheduling, activity assignment, and pause and lunch break assignment for heterogeneous employees. The purpose is to minimize both over and under-coverage along the planning horizon. The problem is solved using four different approaches: a compact MIP model, a B&P method using a heuristic-based nested dynamic program approach, a diving heuristic, and a greedy heuristic utilizing the same subproblem solver. The lack of effectiveness of the RCSPP and grammar-based formulations for the pricing subproblems is attributed to the numerous lower and upper bounds and negative costs for some arcs for the first method, and multiple bounds and long time horizons for the second one. The studied problem has the following hierarchical structure. A task is a span of time during which a single activity is carried out over successive intervals of time; a timeslot is a set of tasks; a day-shift is a set of timeslots divided by a lunch break, and combinations of day-offs and day-shifts result in individual planning for an employee. The problem is resolved using the dynamic programming approach from the inner to the outer level. Some states are heuristically removed from the pricing algorithm, and pause assignment is simplified. Additionally, the branching step selects the most fractional triplet employee-activity-period, then a depth-first strategy is used to search the tree. The computational findings on real-world and randomly generated instances demonstrate that, in many situations, the suggested methods produce optimal or almost optimal solutions within a 24-hour time limit.

Later, in their thesis, Pan (2018) investigated a multi-activity tour scheduling problem for heterogeneous employees that was modeled after the one suggested by Gérard et al. (2016).

They also contained various additions, such as the addition of interruptions besides breaks, variable rest durations between two successive daily shifts, predetermined timeslot length and shift start, and transitions between activities. To address the issue, a B&P method based on CG is suggested. In order to accelerate the B&P convergence, a dual ascent heuristic is also developed. In addition, A pricing problem is presented that uses automata to create timeslots, which are then combined using daily automata and RCSPP to create daily shifts. RCSPP is then used to create feasible weekly schedules. To deal with large-scale problems, various heuristic strategies based on CG, LNS, and tabu search are also discussed. The results show that LNS can get feasible solutions quickly. However, methods based on CG can get high-quality solutions but fail due to the convergence suffering of CG for large-scale instances.

Pan et al. (2018) presented a hybrid heuristic that combines tabu search and LNS techniques for a different problem form the one presented in Pan (2018). In contrast with the previous model, no limit is set on the number of tasks in each timeslot. Also, the activity performed in each task is not determined. There are constraints on the length of an activity, consecutive working hours, daily working hours, break duration, the amplitude of a daily shift, weekly working hours, and rest between two daily shifts. First, the MIP model is suggested to minimize the overall over- and under-coverage. After that, a two-phased heuristic technique is suggested. The employee requirements for each period are employed to construct an initial greedy solution in the first phase, and tabu search is used to integrate the problem's constraints. Utilizing regular language, feasible timeslots and then daily shifts are generated in the second phase. Later, feasible weekly schedules are created utilizing the already-built daily shifts by employing the RCSPP. The findings indicate that, for situations involving two to five activities and ten to sixty employees, the suggested hybrid technique can, in a reasonable amount of time, identify a solution that satisfies all legal requirements while violating workload demand by an average of 4.5%.

2.3 Contributions

This study investigates a multi-activity tour scheduling problem with heterogenous employees and a high degree of flexibility for the shifts, breaks, and activities. The contributions are as

follows.

First, an IP is suggested in which, in contrast to formulations in Pan (2018) and Gérard et al. (2016), any activity could be carried out by any skilled person. The mentioned studies suggest an MIP model in which the number of tasks in each timeslot and the type of activities within each task should be predefined. This restriction is not included in the paper proposed by Pan et al. (2018). however, there are bounds on the duration of breaks while the breaks are defined differently in our problem. Break duration in our model is a function of shift length. That is why we use shift types in our approach.

Second, we developed a heuristic LNS method that generates timeslots, daily schedules, and weekly schedules in a sequential manner, inspired by Pan (2018). However, we use a CFG in place of the automaton used by Pan (2018), which is expected to perform better for the problems with employees with a larger number of skills. We employ algorithms developed by Quimper and Rousseau (2010) to generate feasible timeslots. Due to the difference between the leaf nodes in the daily schedule grammar graph and those in the timeslot graphs, the same techniques cannot be used to construct daily schedules from timeslots. So we make adjustments to the algorithms in Quimper and Rousseau (2010). Then, because a CFG cannot express the weekly rules, the weekly schedules are created by combining daily schedules using a fast, exact algorithm for RCSPP developed by Ahmadi et al. (2021). This approach uses heuristics and preprocessing processes to solve the RCSPP more quickly. This approach has been modified to be consistent with our scheduling problem, which contains lower bounds on the resources in addition to the upper bounds and multiple resources instead of just one.

Finally, we propose a new constraint on task repetition that prevents employees from performing tasks with a specific activity more than once daily. This constraint cannot be represented using a CFG; hence a task assignment IP is suggested. The heuristic LNS model's solution is used as the initial solution for the task assignment IP. In this IP model, the initial solution's shift and break start time and duration and activities that can be performed by each employee each day are included as parameters.

Chapter 3

Model Development

3.1 Problem Setting

Scheduling the workers is a challenging task in a service industry organization where demand is very variable. The underlying setting is as follows. Periods, which are 30 minutes, are the shortest time interval that generates the planning horizon. Employees may be assigned to various shifts with various lengths and starting times daily. A shift could last from 4 to 12 hours. Additionally, shifts of less than 8 hours have a 30-minute break, and shifts of 8 hours or more have an hour-long break. These breaks may begin at least 3 hours after the start of the scheduled shift and must end prior to the shift's final period. Each employee has a skill set that details the tasks they are capable of carrying out. Each activity has a minimum and maximum time limit. Employees should be assigned to either an activity or a break in each period of their allocated shift. Moreover, there are restrictions on the total number of periods worked per day, the total number of periods worked per week, the number of working days per week, and the consecutive working days each week. The goal is to assign schedules to use employees with the least amount of overall excess and shortage of employees across the planning horizon.

3.2 Tour Scheduling Integer Program

Since there is only a restriction on the length of each shift in this setting, the shifts that employees can be assigned to are rather variable. Therefore, if all of the shift-related characteristics are

included as decision variables, the mathematical model will be very large. We must enumerate every potential shift for each duration and start time throughout shop business hours and assign employees to those enumerated shifts in order to simplify the model and reduce the number of constraints. After enumeration, the model will receive as input parameters the shift length, start period, working periods, and the number of break periods for each enumerated shift. For instance, Table 3.1 contains a collection of enumerated shifts with a 4-hour length if the store is open for 6 hours. The shifts with a particular start time, duration, and break times allocated to each employee are called actual shifts. For enumerated shift 1, the start period is 1, the end is 8, the length is 8, and the break length is 1. The actual shift created from enumerated shift 1 have one break period (30-minute break) in period 7. The break should be at least 3 hours after the start of the shift and end before the last period.

Table 3.1: Parameters for each enumerated 4-hour shift

Enumerated Shift	Start Period	End Period	Length (periods)	Break Length (period)
1	1	8	8	1
2	2	9	8	1
3	3	10	8	1
4	4	11	8	1
5	5	12	8	1

In the situation where the shifts are not enumerated, we must use the variable $z_{e,i,p,d}$, which represents that employee e is allocated to a shift beginning in period p with duration d on the day i, in place of variable $z_{e,s,i}$ in our IP model, where s is the enumerated shift index. We found that enumeration of the shifts would lead to a smaller model comparing the number of variables and constraints. We define the sets, indices, parameters, and decision variables in Table 3.2.

Appendix A contains the model formulation. For all activities and periods, the objective function seeks to reduce the weighted total of shortage and excess. Both shortage and excess are allowed in the problem setting. However, they are penalized by costs equal to c_s and c_e, respectively. Based on the environment of scheduling and the importance of shortage or excess, the values for these costs could be defined to better meet the goal of demand coverage. If the staff is not hired, the constraint sets (1) & (2) make sure that the sum of working days in the time horizon for each employee is equal to zero, and if the staff is hired, the sum is between a

Table 3.2: Sets, indices, parameters, and decision variables for the IP model

Sets	
I	set of days in the planning horizon
S	set of enumerated shifts
P	set of periods
A	set of activities
E	set of employees
A_e	set of activities that employee e has skill and is able to perform
B	set of break durations, where $B = \{1, 2\}$ (30-minute and 1-hour break, respectively)
Indices	
i	index of days, where $i \in I$
s	index of enumerated shift, where $s \in S$
p	index of periods, where $p \in P$; period p is defined as the interval $[p, p+1)$
a	index of activities, where $a \in A$
e	index of employees, where $e \in E$
b	index of break durations, where $b \in B$
Parameters	
p_n	number of periods in each day
c_s	shortage cost for each period and activity
c_e	excess cost for each period and activity
$d_{i,p,a}$	number of workers required on day i, in period p, for activity a
l_s	enumerated shift s length
$\overline{l}_s$	enumerated shift s start period
$\underline{l}_s$	enumerated shift s end period
lb_s	enumerated shift s break length
$wp_{p,s}$	is 1 if period p is a working period of the enumerated shift s, otherwise 0
$[l_a^{ac}, u_a^{ac}]$	bounds on the duration of activity a
$[l_e^{dp}, u_e^{dp}]$	bounds on the daily working periods for employee e
$[l^{wp}, u^{wp}]$	bounds on the weekly working periods
$[l^d, u^d]$	bounds on the weekly working days
u^{cd}	maximum number of consecutive working days
$w_{e,i}$	is 1 if day i in previous week was a working day for employee e, otherwise 0; $i \in \{-(u^{cd}-1), \ldots, 0\}$
Binary Decision Variables	
$z_{e,s,i}$	is 1 if and only if employee e is assigned to the enumerated shift s on day i
$w_{e,i}$	is 1 if and only if employee e is working on day i
$y_{e,i,p,b}$	is 1 if and only employee e on day i starts a break in period p with length b
$t_{e,i,a,p}$	is 1 if and only employee e on day i starts to do activity a in period p
$x_{e,i,a,p}$	is 1 if and only employee e on day i performs activity a in period p
h_e	s 1 if and only if employee e is assigned to at least one shift in the planning horizon
Integer Decision Variables	
$\underline{u}_{i,p,a}$	shortage on day i in period p for activity a
$\overline{u}_{i,p,a}$	excess on day i in period p for activity a

minimum and maximum. In the event that it is a working day, constraint set (3) assigns the staff exactly one enumerated shift. Otherwise, there is no shift assignment. The maximum number of consecutive working days is limited by constraint set (4). The sum of all successive $u^{cd}+1$ variables for working days must be less than or equal to u^{cd}. If an employee does not possess the required skill, we cannot allocate this employee to an activity that requires the same skill under the constraint set (5). Constraint set (6) assigns a staff member a break period with a length dependent on the given enumerated shift s. The break must begin at least three hours after the allotted enumerated shift start period ($\bar{l}_s$) and terminate at least one period before it ends ($\underline{l}_s$). Therefore, the employee's break may begin during the periods $\bar{l}_s+6$ to $\underline{l}_s-lb_s$. Every employee on staff is guaranteed a maximum of one break per day according to the constraint set (7). Employees not assigned a shift that day should not be put on break periods. The shortage and the excess amount for each activity at each period are determined using the constraint set (8). Therefore, the total excess and required staff for that activity equal the number of workers allocated to conduct that activity and the shortage during that time. Unless it is a break time, constraint set (9) requires assigned personnel to do exactly one activity throughout each period of the assigned shift. Additionally, a staff member must be assigned to a shift that includes the period in which they perform an activity. If the employee is assigned to the enumerated shift s and period p is a working period for s, the first expression on the right side will equal 1. If an employee has begun their break and has not yet ended in period p, the second expression will equal 1. In order to have a break time in period p, the break time must not begin earlier than period $p-b+1$ if the break duration is equal to b. To keep the third index for the y variable feasible, the upper bound on d is set to $min(b,p)$. Constraint sets (10) to (12), proposed by Pan et al. (2016), relate the variables x and t associated with performing and starting an activity. If an employee is already engaged in an activity before period p, constraint set (10) prohibits starting that activity in period p. This is done to avoid going over the allowed activity duration. The cause of p's range is $[2, p_n-l_a^{ac}+1]$ is that starting an activity must take place during a time frame that allows for its continuation for at least l_a^{ac} periods. The x variable will always equal 1 if activity begins during a period due to the constraint set (11). Constraint set (12) requires the variables to take on values in such a way that if a worker is engaged in an activity during a period, they were either engaged in that activity during the preceding period or they have

just begun it. Constraint set (13) ensures that if an employee is doing an activity in the first period, they have started doing the activity in the first period. The minimum activity duration is constrained by the constraint set (14). As a result, if an employee starts an activity during a period, they must perform it throughout all l_a^{ac} periods. The maximum activity duration is set by the constraint set (15). Therefore, if an employee begins doing activity a in period p, the activity a could be assigned to at most l_a^{ac} periods in the following $u_a^{ac}+1$ periods beginning from period p. Because the indices for the summation on the right side of the constraint cannot exceed the upper bound limit, p does not exceed $p_n - l_a^{ac} + 1$. The minimum and maximum working periods per day and week for the hired personnel are determined by constraint sets (16) to (19). The decision variables are defined by the constraint sets (20) to (27). A numerical example of the tour scheduling IP could be found in Appendix B.

Chapter 4

Solution Methodology

In this chapter, we develop an algorithm to solve the problem described in Chapter 3 heuristically. First, we generate feasible daily schedules based on the considered daily rules. Second, a feasible combination of daily schedules is chosen to create weekly schedules. The generation of daily schedules is done in two steps: timeslot generation and daily schedule generation. To generate timeslots, we model the timeslot rules using a CFG. Next, two timeslots and a break are combined using a CFG to generate daily schedules. As a CFG cannot represent the weekly constraints, a dynamic programming RCSPP is employed to generate the weekly schedules considering the daily schedules. A large-neighborhood search method is used to assign the weekly schedules to the employees in a round-robin way. Finally, two improvements are proposed to reduce the computational time of the algorithm heuristically, which are used for large-scale instances.

4.1 Context-Free Grammar

A CFG is defined as a tuple of the type $G = \langle N, S, \Sigma, P \rangle$, where N stands for the set of *non-terminals* in upper case symbols and Σ for the set of *terminals* in lowercase symbols. S indicates the *starting non-terminal*. P indicates *productions* of type $A \rightarrow w$, where w is a sequence of terminals and non-terminals and $A \in N$. A *parsing tree* is a tree composed of root S, non-terminal inner nodes, and terminal leaves. Furthermore, each inner node's A children join to produce w when $A \rightarrow w \in P$. The sequence of terminals created by a parsing tree's leaf nodes

is recognized by the grammar. The set of all sequences recognized by the grammar is defined as context-free language. A production $A \to w|v$ denotes either $A \to w$ or $A \to v$.

If each production results in two non-terminals or one terminal, such as $A \to a$ or $A \to BC$, then the grammar has Chomsky normal form (first described by Chomsky (1959)). A directed acyclic graph (DAG) encoding all feasible parsing trees for a CFG in Chomsky normal form is produced via Algorithm 1 proposed by Quimper and Rousseau (2010) on page 10, which is referred to as *QR1* in this research. Each leaf node in the DAG has a terminal and a position in the sequence assigned to it. An inner node is of type $N(A, i, j)$ and has non-terminal A, i as its starting position, and j as its span. If a substring is of length j, starting at position i, and formed from non-terminal A, then this node is considered a member of the graph. A list of pairs of nodes of type $\langle N(B, i, j), N(C, i + k, j - k)\rangle$, where $A \to BC \in P$, form the children of each inner node. This indicates that a non-terminal A starting at position i with a span of j can be created by concatenating two substrings. The first is a substring starting at position i with a length of j generated from non-terminal B, and the second is a substring starting at position $i + k$ with a length of $j - k$ generated from non-terminal B.

Following Quimper and Walsh (2007), the dynamic programming algorithm CYK parser (Cocke (1969), Kasami (1966), and Younger (1967)) is employed to develop the *QR1* algorithm. If a sequence is from a context-free language, the CYK parser returns a parsing tree. The terminals and their positions in the sequences are displayed in this graph's leaves. Each inner node, denoted as $N(A, i, j)$, is associated with a non-terminal A that starts in position i with span j.

Context-free language can be improved by specifying valid spans for the subsequences suggested by Quimper and Rousseau (2010) in order to impose length restrictions on the subsequences. For instance, $A_{S_a} \to B_{S_b} C_{S_c}$ indicates that the sets S_a, S_b, and S_c specify the acceptable spans for the A, B, and C subsequences, respectively. Line 16 of *QR1* must be modified to accommodate these constraints by adding $j \in S_a$, $k \in S_b$, and $j - k \in S_c$.

In order to determine the price of the grammar graph's best cost sequence, Quimper and Rousseau (2010) suggest using Algorithm 2 on page 12, referred to as *QR2* in this research.

The sequence can be obtained by retracing the grammar graph using the best cost subsequence selected at each inner node or leaf node. The aforementioned algorithms will be used in the development of our solution methodology.

For workforce scheduling problems, both regular and context-free languages have been utilized in the literature. In these languages, the schedules are expressed as sequences. Pesant (2004) presented regular constraints to generate sequences that belong to a regular language. Demassey et al. (2006) extended Pesant (2004)'s work to cost-regular constraints to accept sequences with a cost threshold. A regular language is the sequences recognized by an automaton. An automaton is defined as a set of states and transitions, which could be shown by a directed graph. The nodes and edges of this graph are states and transitions of the automaton, respectively. Quimper and Rousseau (2010) proposed two methods using regular and context-free languages for multi-activity shift scheduling problems. They demonstrated that a CFG is more effective when the problem has many activities. In this situation, there are fewer nodes in the grammar graph than nodes and edges in the automaton expanded graph. Moreover, compared to an automaton, a CFG is more expressive, and writing the model is more straightforward. Hence, we will model our problem using a CFG.

In the setup described in Section 3.1, weekly restrictions, including the minimum and the maximum number of working days and periods per week, and restrictions on the consecutive working days, cannot be presented as a CFG. As a result, we will not include the weekly constraints in the first stage. The CFG model is illustrated in the following for feasible daily schedules that takes employee e's daily rules into account. We incorporate the hierarchical schedule structure suggested by Gérard et al. (2016) into this model. In this structure, a task is a sequence of periods where the same activity is performed consecutively. A sequence of tasks creates a timeslot. Note that two tasks with the same activity cannot be performed in a row. The reason is that if this constraint is not considered, the bounds on the activity duration would be violated. A daily schedule is created by placing two timeslots in succession with a break in between. In the

following, we propose the CFG model for the problem described in Chapter 3 .

$$S \implies RW_1R|RW_2R|R \tag{1}$$

$$W_{1[a_1,b_1]} \implies P_{[6,b_1-2]}bP \tag{2}$$

$$W_{2[a_2,b_2]} \implies P_{[6,b_2-3]}bbP \tag{3}$$

$$R \implies RR|r|\epsilon \tag{4}$$

$$P \implies A_aP'_a|A_a \qquad \forall a \in A_e \tag{5}$$

$$A_{a[l_a^{ac},u_a^{ac}]} \implies A'_a \qquad \forall a \in A_e \tag{6}$$

$$A'_a \implies aA'_a|a \qquad \forall a \in A_e \tag{7}$$

$$P'_a \implies A'_{a'}P'_{a'}|A'_{a'} \qquad \forall a, a' \in A_e,\ a' \neq a \tag{8}$$

$$a_1 = \begin{cases} l_e^{dp} & \text{if} \quad 8 \leq l_e^{dp},\ u_e^{dp} \leq 15 \\ l_e^{dp} & \text{if} \quad 8 \leq l_e^{dp} \leq 15,\ 16 \leq u_e^{dp} \leq 24 \\ 0 & \text{if} \quad 16 \leq l_e^{dp},\ u_e^{dp} \leq 24 \end{cases} \quad a_2 = \begin{cases} 0 & \text{if} \quad 8 \leq l_e^{dp},\ u_e^{dp} \leq 15 \\ 16 & \text{if} \quad 8 \leq l_e^{dp} \leq 15,\ 16 \leq u_e^{dp} \leq 24 \\ l_e^{dp} & \text{if} \quad 16 \leq l_e^{dp},\ u_e^{dp} \leq 24 \end{cases}$$

$$b_1 = \begin{cases} u_e^{dp} & \text{if} \quad 8 \leq l_e^{dp},\ u_e^{dp} \leq 15 \\ 15 & \text{if} \quad 8 \leq l_e^{dp} \leq 15,\ 16 \leq u_e^{dp} \leq 24 \\ 0 & \text{if} \quad 16 \leq l_e^{dp},\ u_e^{dp} \leq 24 \end{cases} \quad b_2 = \begin{cases} 0 & \text{if} \quad 8 \leq l_e^{dp},\ u_e^{dp} \leq 15 \\ u_e^{dp} & \text{if} \quad 8 \leq l_e^{dp} \leq 15,\ 16 \leq u_e^{dp} \leq 24 \\ u_e^{dp} & \text{if} \quad 16 \leq l_e^{dp},\ u_e^{dp} \leq 24 \end{cases}$$

According to the production set (1), an employee will either be assigned to a shift that lasts longer than or equal to 8 and less than 16 periods with one break period (W_1), longer than or equal to 16 and less than or equal to 24 periods (W_2) with two break periods, or no shift at all. Productions (2) and (3) show how daily schedules can be created using timeslots (P) and breaks (b). Additionally, the break start time is restricted to a time that is at least six periods following the start of the shift. Moreover, the minimum and maximum duration limit on P is set considering the break start time and the overall break and second timeslot duration. The parameters a_1, a_2, b_1, and b_2 are determined by the limits on the daily working periods.

For instance, the employee e can only be assigned to shifts of type W_1 if u_e^{dp} is less than 16 periods. Therefore, for this employee, a_2 and b_2 are equal to 0, but a_1 and b_1 are equal to l_e^{dp} and u_e^{dp}, respectively. The rest period sequence is produced by production set (4) for the periods throughout the planning horizon that the employee is not assigned to any shifts. Timeslots are generated by the production set (5) using a sequence of tasks (A_a). Production set (6) limits the time allotted for each task's associated activity. A task is created from a sequence of a single activity by the production set (7). Two tasks with a similar activity performed back-to-back are forbidden by the production set (8). Note that all productions from (5) to (8) are based on the activities in the employee's skill set and are proposed by Restrepo et al. (2016).

4.2 Daily Schedules

To make the daily schedules, we can use two methods. The first method is to apply *QR1* to the grammar presented in Section 4.1 to produce the grammar graph for each employee. However, preliminary experiments indicated that this methos is ineffective for setting. The second method has two steps to generate daily schedules. In the first step, feasible timeslots are generated. The second step generates feasible daily shifts based on the generated timeslots. This method is suggested by Gérard et al. (2016) by developing a nested dynamic program and Pan (2018) by developing automata to generate timeslots and daily schedules. We also use the second method to generate daily schedules. Although Pan (2018) used automata, we will employ a CFG in our method. As previously stated, a CFG is superior to an automaton.

To produce all the feasible sequences for the daily schedules, we must first generate all the possible timeslot sequences for each duration. The Algorithm 1 is an illustration of this process. The Chomsky normal form of the production sets (5) through (8) mentioned in Section 4.1 are the grammar we use to generate feasible timeslots. The Chomsky normal form of these productions could be found in Appendix C. A timeslot's lowest possible duration is equal to the minimum lower bound of the activity duration for the activities in the set of employee skills, and its highest possible duration is equal to the maximum of $b_1 - 2$ and $b_2 - 3$ values as explained for productions (2) and (3). To retrieve the grammar graphs for all possible lengths, we do not need to build them separately; instead, we can generate the grammar graph for the longest length and

then identify the descendants of the root node $N(P, 1, d)$ for all d in feasible durations. Indeed, the grammar graphs for the smaller lengths are also constructed simultaneously as the grammar graph for the longest length is built.

Algorithm 1 Algorithm for building the timeslot for each employee e

1: Set the productions (5) through (8) in Chomsky normal form as the grammar;
2: Set $t = \min_{a \in A_a} l_a^{ac}$ and $T = \max(b_1 - 2, b_2 - 3)$ as the min and max possible length for a timeslot duration;
3: Generate the grammar graph using QR1 for the sequence of length T;
4: **for** $d = t, \ldots, T$ **do**
5: Find the descendants of $N(P, 1, d)$;
6: **end for**

Next, we use the constructed feasible timeslot sequences to build every possible daily schedule for every possible duration. Algorithm 2 illustrates the procedure. Productions (2) and (3) mentioned in Section 4.1 in Chomsky normal form, which are related to producing acceptable daily schedules, are considered as two separate grammars in this step. The graphs will be produced using production (2), (3), or both, depending on the l_e^{dp} and u_e^{dp} values for each employee. For instance, if an employee's l_e^{dp} and u_e^{dp} values are 9 and 18, respectively, two distinct graphs utilizing productions (2) and (3) must be created. However, we would only create one graph using production (2) if u_e^{dp} equaled 14. It should be noted that *QR1* cannot directly be utilized to generate the grammar graph for daily schedules. The reason is that the definition of leaf node is different, and all the leaf nodes do not have span equal to 1. Nodes of non-terminal P with any span are considered leaf nodes in the daily schedule graph. Children of leaf nodes are not just terminals but also sequences associated with feasible timeslots. To create daily schedule grammar graphs, we use Algorithm 3, which is a modification of *QR1*. We do not need to separately produce graphs for the different daily schedule durations, as described in Algorithm 1, which constructs timeslots. The graph for the sequences of length T will be generated initially. The graph for each possible duration will then be constructed from the

descendants of $N(P, 1, d)$.

Algorithm 2 Algorithm for building the daily schedules for each employee e

1: Set productions (2) and (3) in Chomsky normal form as separate grammars with W_1 and W_2 starting non-terminals;
2: Set $t = l_e^{dp}$ and $T = u_e^{dp}$ as the min and max possible length for a daily schedule duration;
3: Generate grammar graphs using Algorithm 3 for the sequence of length T;
4: **for** $d = t, \ldots, T$ **do**
5: **if** $d \leq 15$ **then**
6: Find the descendants of $N(W_1, 1, d)$;
7: **else**
8: Find the descendants of $N(W_2, 1, d)$;
9: **end if**
10: **end for**

As previously noted, timeslot sequences may be the children of the daily schedule graphs' leaf nodes. Therefore, lines 4 to 6 in Algorithm 3 are added to add timeslots of length 1 to children of non-terminal P with span 1 if the minimum timeslot duration value is equal to 1. Also, lines 14 to 16 are added to include timeslots of length j as the children of non-terminal P with a span larger than 1, provided that the span is within the employee's acceptable timeslot duration limitations.

Algorithm 3 Algorithm for building the daily schedules graphs

for all non-terminals A **do**
  for $i \in [1, T]$ **do**
    Create node $N(A, i, 1)$;
    if $A = P \ \& \min_{a \in A_a} l_a^{ac} = 1$ **then**
      $Children(N(A, i, 1)) \leftarrow$ timeslots with duration 1;
    end if
    $Children(N(A, i, 1)) \leftarrow \{t | A \rightarrow t \in G\}$
  end for
end for
for $j \in [2, T]$ **do**
  for $i \in [1, T - j + 1]$ **do**
    for all non-terminal A **do**
      Create node $N(A, i, j)$;
      if $A = P \ \& \min_{a \in A_a} l_a^{ac} \leq j \leq max(b_1 - 2, b_2 - 3)$ **then**
        $Children(N(A, i, j)) \leftarrow$ timeslots with duration j;
      end if
      $Children(N(A, i, j)) \leftarrow \{\langle N(B, i, k), N(C, i + k, j - k)\rangle |$
      $k \in [1, j), A \rightarrow BC \in G, Children(N(B, i, k)) \neq \emptyset, Children(N(C, i + k, j - k)) \neq \emptyset\}$;
    end for
  end for
end for

4.3 Weekly Schedules

When creating weekly schedules, weekly limits are considered along with a combination of daily schedules and off days. These restrictions are limits on the maximum number of consecutive working days u^{cd}, weekly working days $[l^d, u^d]$, and weekly working periods $[l^{wp}, u^{wp}]$. Since a CFG cannot account for these restrictions, RCSPP is employed. We solve the RCSPP and generate weekly schedules using the fast exact algorithm developed by Ahmadi et al. (2021). Bidirectional A* search expands partial paths in both directions, from source to target and target to source, to find a complete minimum cost path. Normally, in the RCSPP context, the minimum cost path to each node in each direction is used to solve the problem heuristically. In the approach proposed by Ahmadi et al. (2021), on top of a bidirectional A* algorithm, numerous heuristics are constructed to improve both the initialization process and the core search. Nevertheless, a few adjustments have to be made to match the algorithm to our scheduling problem. For instance, only one resource is considered in Ahmadi et al. (2021)'s approach, and there are no

minimum restrictions on resource usage. However, we have multiple resources and lower bounds on resource usage.

After generating all feasible schedules for each employee for each day, we construct a DAG to provide the ideal weekly schedule, as illustrated in Figure 4.1. Each layer of the graph is associated with one day of the week. Our convention is that all the nodes except the last represent working days and gives a schedule. The last node represents an off day. For example, nodes x_{11} to x_{1n_1-1} represent the working schedules for day 1, and x_{1n_1} is an off day. We begin at source node s and the end is at sink node t. Since there are no constraints on the schedules for any two consecutive days, we can shift from any layer i schedule to any layer $i+1$ schedule. Thus, there are edges connecting every vertex in one layer to every vertex in the next. The aforementioned weekly constraints are the only restrictions on the routes from s to t. Each edge connecting node u to u' consumes resources $p_{u'}$ and $d_{u'}$ where they display the number of periods in node u' and whether the schedule is a working day, respectively. This edge also has a cost equal to the price of the schedule u'. In Section 4.4, we will explain how we arrived at this cost.

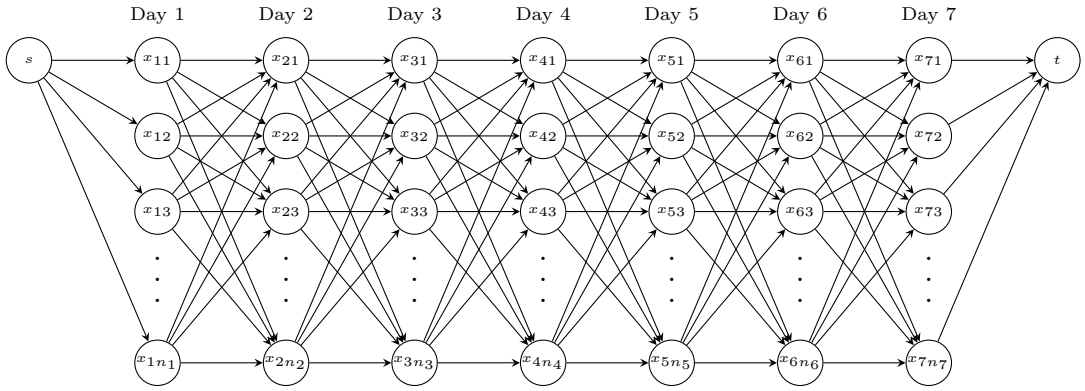

Figure 4.1: The weekly scheduling DAG

Labels are used to store the information of each partial path. Our method defines a *label* as $\{f, g, r, u\}$, where f is the lower bound on the source-target cost, g displays the cost, r displays the set of resource consumptions, and u represents the final node of the partial path. Additionally, the definition of r is $\{WP, DP, CD\}$, where WP stands for the total weekly working periods, DP for the total weekly working days, and CD for the total number of consecutive working days. Moving from node u to node u' will update r as shown below. In the bidirectional method, the

generation of labels start from both the source and the sink node. The labels are then extended forward and backward to the inner layers and combined to create complete routes from the source to the sink node.

$$WP' = WP + p_{u'}$$

$$DP' = DP + d_{u'}$$

$$CD' = \begin{cases} CD + 1 & \text{if } d_{u'} = 1 \\ 0 & \text{else} \end{cases}$$

The initialization phase is the first step in Ahmadi et al. (2021)'s algorithm. This step selects a subset of potential vertices for the following phase, along with obtaining lower bounds on the cost and resource utilization for source-target paths and the initial cost for a complete path. Additionally, the budget factors, which represent the upper limit on resource utilization in each direction, are computed. The direction is indicated by d, which can be either forward (f) or backward (b). The cost of the shortest path from the source or sink node to node u based on the direction is $\underline{g}^d$. This path uses resources in the amount $r_{\underline{g}^d}$. The symbol $\underline{r}^d$ shows a node's resource usage of the resource-optimal path to the node in the direction d, and the cost of this path is shown by the symbol $g^d_{\underline{r}}$. The cost of the best-known solution is denoted by C_0^*. The details of the modified initialisation phase are displayed in Algorithm 4.

Algorithm 4 Initialisation phase

1: Forward Dijkstra on resources WP and DP, find the initial C_0^*
2: Set $\underline{r}^f = \{p_u, d_u\}$ for each node u;
3: Set C_0^* equal to the total cost of all days off;
4: Backward Dijkstra on resources WP and DP
5: Set $\underline{r}^b = 0$ for each node u;
6: Forward Dijkstra on cost, and update C_0^* if possible
7: Run Dijkstra algorithm and find $\underline{g}^f$ for each node;
8: **if** the best-cost path is feasible **then**
9: Update C_0^* and stop;
10: **end if**
11: Backward Dijkstra on cost
12: Set V as the set of expanded nodes;
13: Run Dijkstra algorithm and find $\underline{g}^b$ for each node;
14: **if** $\underline{g}^b + \underline{g}^f \leq C_0^*$ for the node u **then**
15: Add u to V;
16: **end if**
17: Calculate the Budget factors B^f and B^b using nodes V
18: $B^d = 0.5 \times \min(2, \sum_{v \in V} g^d(v) / \sum_{v \in V} g^{d'}(v))$;

The initialization algorithm proposed by Ahmadi et al. (2021) is modified due to the structure of the DAG of the scheduling problem and the lower bound limits on resource usage. In the first stage, we wish to use Dijkstra on each resource independently to obtain $g^f_{\underline{r}}$ for each resource and node. Since no resources are used on an off day on each layer of the DAG, the answer is trivial. Therefore, even without running Dijkstra's algorithm, we can conclude that $g^f_{\underline{r}}$ represents the resource utilization of each node. The resource-optimal path will also consist of only days off during which no resources are consumed. As a result, we assign C_0^* to the cost of all days off schedule. In the second stage, we want to obtain $g^b_{\underline{r}}$ for each resource and node. The answer is trivial, and $g^b_{\underline{r}}$ equals zero for each node for the same reason as previously stated. The third stage employs Dijkstra's algorithm to find $\underline{g}^f$ in the forward direction. The optimal solution is identified if the shortest path is also feasible regarding resource usage. Therefore, we will update C_0^* and then stop. Otherwise, backward Dijkstra's algorithm is used to find $\underline{g}^b$ for each node. According to a lemma put forth by Ahmadi et al. (2021), the search can be terminated on a node if the criteria in line 14 is not met, and we are not required to examine it in the core search.

Therefore, the set of expanded nodes for the main search will only include the nodes that satisfy this requirement. Finally, the budget factors, which represent the upper bound on the maximum resource utilization in each direction, are computed.

The enhanced biased bidirectional A* called RC-EBBA* is the next step associated with the main search. We must change some of its components to adapt this algorithm to a problem with multiple resources and lower bounds on resource usage. Here are the modifications: In line 3, we update the definition of variable $r^d_{min}(v)$ from tracking the minimum resource usage at each node to tracking all resource usage vectors $r^d(v)$. In line 11, in order to prune any partial path with a resource vector similar to a partial path already constructed for that node, we alter the pruning condition to $r \in r^d(v)$. Also, in line 14, the resource vector will be added to the set of resource vectors. Finally, the condition in line 34 needs to be altered to $R_{min} \leq r + r' \leq R$, where R_{min} and R are the lower and upper bound vectors on the resources usage, in order to incorporate the lower bound. Note that there is no limit on the minimum number of consecutive days. So, we set the lower bound for this variable equal to zero. Additionally, this approach makes use of two auxiliary functions. Line 8 for the *Feasible* function will be modified similarly to line 11 for the core search. Lines 4 through 6 for the *EarlyC*Update* will be removed.

4.4 Large Neighborhood Search

The iterative technique known as the LNS, which was first developed by Shaw (1998), starts with an initial solution and improves it by destroying and then repairing a portion of the solution. The neighborhood is defined by the techniques used to destroy and rebuild. This means that all the solutions that can be achieved by using the destroy approach and then the repair method are collectively referred to as the neighborhood $N(x)$ of the solution x.

High-quality local optimum can be discovered by searching large neighborhoods. On the one hand, it takes time to scan a large neighborhood; therefore, applying different filtering methods to make the search focused is essential. On the other hand, unlike a smaller neighborhood, a large neighborhood enables the heuristic to travel the solution space easily, even in the circumstances

of strict boundaries (Pisinger and Ropke 2010). In order to efficiently produce a better solution in each iteration, the neighborhoods must be carefully picked.

In our scheduling problem, the destructor picks employee $e \in E$ and preserves the other employees' schedules while removing the one for employee e. All feasible schedules that can be assigned to this employee are in the neighborhood. The builder searches the neighborhood for the ideal schedule that enhances the solution. Our problem's cost function minimizes the overall over-coverage and under-coverage across all activities and periods. In each LNS iteration, we search for the schedule with the minimum cost. The residual workload is the unmet demand taking into account the partial solution. Considering the residual workload, the builder selects the minimum cost schedule associated with the least amount of overall over-coverage and under-coverage.

Algorithm 5 demonstrates how we apply LNS in our problem. We begin with the initial solution, which consists of a blank set of schedules. The destructor picks the workers in a round-robin manner according to their index in each iteration. The employee's schedule is then deleted. We will start by creating the most cost-effective time slots for each day. The graphs produced in Algorithm 1 and QR2 for the employee are utilized to find the least-cost timeslot for each feasible duration d and each starting period p. When utilizing QR2 to identify the best-cost timeslot, only the residual demand during the intervals p to $p + d - 1$, which establishes the prices related to the graph's leaf nodes, is taken into account. Next, using Algorithm 2 and the modified QR2, Algorithm 6, best-cost daily schedules are created for each feasible duration and starting slot. Because the leaf nodes in daily graphs may also have timeslot children other than terminals, we cannot utilize QR2. Similar to timeslot generation, the best-price daily schedule is determined by solely taking into account the residual demand of the periods associated with each daily schedule span. The cost of the periods that the daily schedules do not cover must thus be included after the daily schedules are generated to obtain the total cost of the schedule for all the periods. Next, we will combine the daily schedules using the RC-EBBA* algorithm to get the optimal weekly schedule. Up until a local optimum is discovered, the LNS continues.

The cost of the optimal daily plan for the given duration d and starting period p is shown in Algorithm 6. A modification of QR2 is used in this algorithm. The daily graphs' leaf nodes

differ from those of the timeslot graphs, as was previously mentioned. In daily graphs, there are two types of leaf nodes. $N(P, i, s)$ is the initial type and can have timeslot-type children. Line 2 entails assigning each leaf node of this type the best-cost timeslot, which has already been constructed for each duration and starting period. The second type, $N(X, i, 1)$, can have children of type terminal b; X is the nonterminal in the Chomsky normal form of the language that yields to break, as is shown in Appendix C. If the schedule for the current employee is removed and they are given a break, the cost of the period i is given in line 5 by the expression $cost(S[i] \backslash \{s_e[i]\} \cup \{b\}, i)$, where $S[i]$ and $s_e[i]$ represent the set of schedules for all employees and employee e in period i, respectively.

Algorithm 5 LNS

1: Set weekly schedule $s_e = \{\},\ \forall e \in E$;
2: **while** local optimum is not found **do**
3: **for** employee $e \in E$ **do**
4: Set $s_e = \{\}$
5: **for** day $i \in I$ **do**
6: Set $t = \min_{a \in A_a} l_a^{ac}$, and $T = \max(b_1 - 2, b_2 - 3)$ as the min and max possible timeslot length;
7: **for** $d = t, ..., T$ **do**
8: **for** starting period $p = 1, ..., p_n - d + 1$ **do**
9: Find the best-cost timeslot using the timeslot graph of duration d built by Algorithm 1 and QR2 for starting period p;
10: **end for**
11: **end for**
12: Set $t = l_e^{dp}$ and $T = u_e^{dp}$ as the min and max possible length for a daily schedule duration;
13: **for** $d = t, ..., T$ **do**
14: **for** starting period $p = 1, ..., p_n - d + 1$ **do**
15: Find the best-cost daily schedule using the daily schedule graph of duration d built by Algorithm 2 and 6 for starting period p;
16: **end for**
17: **end for**
18: **end for**
19: Use Algorithm 4 and the modified RC-EBBA* to generate the best weekly schedule as s_e;
20: **end for**
21: **end while**

Algorithm 6 Algorithm for computing the best schedule cost in a daily schedule graph for duration d and starting period p

for every leaf node $N(P, i, s)$ **do**
    $weight(N(P, i, s)) \leftarrow min(cost(\text{timeslot with starting period } p + i - 1 \text{ and duration } s))$;
end for
for every leaf node $N(X, i, 1)$ **do**
    $weight(N(X, i, 1)) \leftarrow cost(S[i] \backslash \{s_e[i]\} \cup \{b\}, i)$;
end for
for every inner node $N(A, i, j)$ in post-order **do**
    $weight(N(A, i, j)) \leftarrow min\{weight(N(B, x, y) + weight(N(C, w, z)|$
        $\langle N(B, x, y), N(C, w, z)\rangle \in Children(N(A, i, j))\}$;
end for

4.5 Improvements

The most computationally expensive part of our solution methodology is the dynamic programming that produces the weekly schedules. The computation time grows exponentially as the DAG's layer count increases. Since we are focusing on the weekly scheduling problem, there must be at least seven layers in the DAG. Reducing the number of daily schedules in each layer and keeping the most promising ones is a natural idea for managing computation time. In order to simplify our procedure, we can apply the following two heuristics. These methods are a modification of the methods proposed by Pan (2018).

4.5.1 Starting Slots Selection

This heuristic is used to pick a few starting periods and create daily schedules only for those periods. The demand for all the activities that an employee is capable of performing is assessed while choosing the periods for that individual. First, only the periods with increasing demands are selected. All the slots $\{id + s_1 |\ i \in N,\ id + s_1 < s_2\}$ are added to the starting periods set if the time difference between two consecutively picked periods s_1 and s_2 is more than the daily schedule length maximum limit, $d = max(b_1 - 2, b_2 - 3)$. We call this improvement I_1.

4.5.2 Daily Schedules Selection

Even after applying the first improvement to some cases, the number of daily schedules is still substantial. Because of this, the second improvement will be used to generate fewer schedules overall. This approach selects the two best schedules at the lowest cost for each daily schedule duration. We call this improvement I_2.

Chapter 5

Repetition Constraints

Repeating a task with a specific activity during a daily shift might not be ideal for some workplaces. Therefore, in some circumstances, we should prohibit it. This implies that a worker cannot restart a task with the same activity they have already performed. Employees can only perform a task with a specific activity more than once if they perform it just before the break. They can then begin doing it in the first period following the break. Additionally, repetition is possible if the employee possesses one skill. For example, for an employee with skills to perform activities 1 and 2, schedule 112211, where each number shows the activity being performed, is not feasible, whereas 112222 is a feasible schedule. The reason to include this constraint is that the minimum duration limit for some activities is high. So, scheduling a task with this activity for the employee so many times during one shift is not sustainable. Also, fairness is not applied between employees. Each employee should do a task with a specific activity at most once during one day. Another reason is that each activity could be related to different building floors in the retail industry. For example, activity 1 represents activity on the first floor and activity 2 on the second floor. If an employee is assigned to schedule 121212, they must change the floors several times during a shift, which is unreasonable. The proposed tour scheduling LNS algorithm can provide schedules without considering the task repetition constraint. In other words, an employee can repeatedly start and stop performing tasks with similar activities throughout a shift. For the problems with no limit on task repetition, we only need to use the proposed LNS algorithm to solve the problem. Otherwise, we modify the solution methodology. First, we will include the task repetition restriction in the tour scheduling IP suggested in Section 3.2. Then,

we will demonstrate how this constraint should be included in our solution methodology using a task assignment IP. So, for the problems with this constraint, other than the tour scheduling LNS, a task assignment IP is used.

The IP should include the following constraint to prevent repetition. The repetition parameter is r_e, which is equal to 1 if the repetition is permitted for employee e, 0 otherwise. The second term on the left states that an employee may begin a task after a break ends with the same activity as the task completed just before the break, even if the repetition is zero.

$$\sum_{p=1}^{p_n - l_a^{ac}+1} t_{e,i,a,p} - \sum_{p=1}^{p_n - l_a^{ac}+1} \sum_{b} t_{e,i,a,p+b} \times x_{e,i,a,p-1} \times y_{e,i,a,p} \leq 2p_n r_e + 1 \qquad \forall e \in E, i \in I, a \in A$$

Next, we will suggest a strategy for incorporating this constraint into the LNS search. Importantly, a CFG is not able to capture the repetition limitations, since a CFG cannot track the number of non-terminals or non-consecutive terminals in a sequence. We utilized RCSPP for generating weekly schedules for the same reason since a CFG could not model weekly restrictions. We propose the following substitute way to include such constraints. First, we will execute the weekly scheduling LNS algorithm to obtain weekly schedules for employees. The output of the algorithm will be used as an initial solution, and we will extract the set of working days, the shift start time, shift end time, break periods, and the set of tasks performed by each employee on each day. We will use the algorithm's output as a starting point and then extract the set of working days, the start and end periods of each employee's shifts, their break times, and the activities they performed each day. This data will be used later as input for a task assignment IP. Based on the shifts and set of activities already assigned to the employees in the initial solution, this IP allocates tasks to the employees, including the new constraint. This IP will be considerably smaller than the original one because the majority of the variables in the original IP are now parameters. Table 5.1 defines the decision variables, parameters, and indices of this task assignment IP.

Appendix D contains the model formulation. The objective function seeks to reduce the

Table 5.1: Sets, indices, parameters, and decision variables for the task assignment IP

Sets	
I	set of days in the planning horizon
P	set of periods
A	set of activities
E	set of employees
$B_{e,i}$	set of break periods for employee e on day i
W_e	set of working days for employee e
$A_{e,i}$	set of activities for employee e on day i
Indices	
i	index of days, where $i \in I$
p	index of periods, where $p \in P$; period p is defined as the interval $[p, p+1)$
a	index of activities, where $a \in A$
e	index of employees, where $e \in E$
Parameters	
p_n	number of periods in each day
c_s	shortage cost for each period and activity
c_e	excess cost for each period and activity
$d_{i,p,a}$	number of workers required on day i, in period p, for activity a
$ss_{e,i}$	shift start time for employee e on day i
$se_{e,i}$	shift end time for employee e on day i
$[l_a^{ac}, u_a^{ac}]$	bounds on the duration of activity a
$r_{e,i}$	is 1 if employee e can repeat an activity on day i, otherwise 0 (equals 1 if and only if there is just one activity in $A_{e,i}$)
Binary Decision Variables	
$t_{e,i,a,p}$	is 1 if and only employee e on day i starts to do activity a in period p
$x_{e,i,a,p}$	is 1 if and only employee e on day i performs activity a in period p
Integer Decision Variables	
$\underline{u}_{i,p,a}$	shortage on day i in period p for activity a
$\overline{u}_{i,p,a}$	excess on day i in period p for activity a

weighted total of shortage and excess for all activities and periods. The idea is to keep the overall cost as low as possible since it leads to costs for the system. For the activities that are not in the set of skills for every employee every day, constraints (1) and (2) set the variables x and t equal to 0, respectively. For each employee's off days, constraint (3) sets the variable x equal to 0. For the periods not in the shift assigned to each employee each day, the variables x and t are set to equal 0 by constraints (4) and (5), respectively. The shortage and the excess amount for each task in each period are determined using the constraint set (6). Therefore, the total excess and required staff for that task equals the number of people allocated to accomplish

it and the shortage during that period. Except for break times, constraint set (7) assigns each employee exactly one activity per period for the duration of their shift. If an employee is already working on an activity in period $p-1$, constraint set (8) prohibits starting that activity in period p. That is done to avoid going over the permitted activity duration.The reason that p is in the region $[2, p_n - l_a^{ac} + 1]$ is the requirement that starting activity a must be in a period that permits continuing it at least for l_a^{ac} periods. Constraint set (9) ensures that if a task starts in a period, the variable x equals 1 in that period. According to constraint set (10), if an employee is performing an activity during a period, they either perform the same activity during the previous period or begin the task within the present period. Constraint set (11) ensures that if an employee is doing an activity in the first period, they have started doing the activity in the first period. Constraint set (12) sets the minimum activity duration limit. So, if an activity is started in a period, the employee must execute it in all l_a^{ac} periods. The most extensive activity duration is constrained by the constraint set (13). Constraint sets (14) to (16) are used for forbidding task repetition with the same activity. As a result, starting from period p, an activity a could be allocated to an employee for up to u_a^{ac} periods in the subsequent $u_a^{ac} + 1$ periods. The decision variables are defined by the constraint sets (17) to (21). An example of the task assignment IP can be found in Appendix E.

Chapter 6

Computational Experiments

In this section, the performance of the tour scheduling IP and the proposed algorithm is assessed taking into account the additional constraints related to task repetition using two datasets, i.e., the industrial instances and the randomly generated instances. We include the task repetition constraint because it adds another step, task assignment IP, to our method. Hence, it is more difficult to solve the problem. If we do not include this constraint, we only need to use the tour scheduling LNS to solve the problem. The weekly scheduling DAG can hold up to 500 feasible schedules in some instances with flexible daily period boundaries. In order to decrease the number of schedules, we employ I_1 and I_2 improvements, which are mentioned in Section 4.5. We have applied both of these improvements in the later instances where the upper and lower bounds are unequal, meaning employees can be assigned to shifts of different lengths. The tour scheduling IP model is implemented in Julia 1.8.2 and solved by CPLEX 22.1.0. For the proposed algorithm, C++ and Julia are used for the tour scheduling LNS algorithm and task assignment IP implementation, respectively. The task assignment IP model is solved by HiGHS 1.3.0, a solver proposed by Huangfu and Hall (2018).

6.1 Industrial Instances

The industrial data comes from a company that develops workforce scheduling software. The instances relate to clients in the service sector, where demand varies throughout the day. Hence, considering the flexibility in the scheduling can lead to more profitability. The parameters for

Table 6.1: Parameters of industrial instances

Instance	$\|E\|$	$\|A\|$	Avg. Skills	$[l_a^{ac}, u_a^{ac}]$	$[l_e^{dp}, u_e^{dp}]$	$[l^{wp}, u^{wp}]$	$[l^d, u^d]$	u^{cd}
C1	10	4	2.30	[3,16]	[16,16]	[0,80]	[0,5]	5
C2	29	6	2.86	[3,16]	[16,16]	[0,80]	[0,5]	4
C3	102	19	7.18	[1,16]	[16,16]	[0,80]	[4,5]	5
C4	18	14	5.77	[1,16]	[8,22]	[0,132]	[6,6]	6
C5	28	10	1.25	[1,16]	[8,22]	[0,110]	[0,5]	5

each instance are displayed in Table 6.1. The second and third column represent the number of employees and activities, respectively. The fourth column indicates the average number of activities in each employee's skill set. Columns five through nine show the constraints for activity, daily, and weekly parameters. As explained, due to the high flexibility of daily schedules duration for instances $C4$ and $C5$, improvements I_1 and I_2 are employed for solving these instances.

The Table 6.2 shows the running times and objective costs of the proposed algorithm and CPLEX for these industrial instances. The first column shows the instance index. The second column indicates the initial cost which means the total shortage when no employee is assigned to any schedule. This is the initial solution for the proposed algorithm and the CPLEX solver. Columns three and four display the running times of the proposed algorithm, for the tour scheduling LNS and the task assignment IP, respectively. For each of these stages, a time limit of 30 minutes is used. The fifth column is the objective cost of the proposed algorithm. The last two columns are the objective costs of the solutions found by CPLEX for two test setups. First, we use the solution time of our proposed algorithm as the limit for the CPLEX solver (variable limit). For example, in C1, the time limit is 175.41 seconds, equal to the summation of the third and fourth columns. Second, we simply set a 1-hour time limit for the CPLEX solver.

CPLEX failed to find the optimal solution within the allotted time for all the industrial instances. Hence, the objective cost of the best solution found is reported. By comparing the results of the two methods, i.e., the proposed algorithm and CPLEX, the faster method and lower objective cost are *bolded* in the table for the running time and objective cost, respectively. As CPLEX did not stop solving before the time limit, the proposed algorithm was faster since it solved all instances within the limited time. In addition, the findings for the industrial instances demonstrate better objective costs across all instances. We observe that in C3, CPLEX cannot

Table 6.2: Running times (in seconds) & objective cost of the proposed algorithm and CPLEX solver

		Running Time		Obj. Cost		
					CPLEX	
Instance	Init. Cost	Tour Sch. LNS	Task As. IP	Pro. Alg.	Var. Lim.	1h Lim.
C1	707	**168.27**	**7.14**	**199**	707	293
C2	3994	**32.62**	**6.07**	**2214**	3994	2246
C3	5345	**908.87**	**229.05**	**2189**	5345	5345
C4	1816	**1195.56**	**71.92**	**507**	663	663
C5	2205	**137.92**	**6.31**	**710**	2205	770

discover a better solution than the initial one in one hour due to a large number of employees. Except for instance C4, CPLEX could not find a better solution than the initial solution in the time limit equal to our proposed algorithm running time. Instance C4 was not solved by our proposed algorithm as quickly as the other instances. The reasons are the heterogeneous employees with a high average number of skills and flexible daily period boundaries that yields high computational time in our proposed algorithm for creating the grammar graphs and generating schedules at each iteration of the tour scheduling LNS.

6.2 Random Instances

To further study the influence of parameters on the running time and objective cost for the proposed algorithm, we tested randomly generated instances. After performing initial tests, we concluded that parameters could be divided into four groups and algorithm performance could be tested by changing the flexibility degree of each group. These groups include employee number, activity, daily, and weekly parameters. For the employee number, we generate instances with 10, 45, and 80 employees. For each of the remaining groups, we consider two possibilities of high flexibility (H) and low flexibility (L).

For instance with H activity parameters flexibility, $[l_a^{ac}, u_a^{ac}] = [1, 16]$ and the number of activities, $|A|$, is equal to $\frac{|E|}{5} + 2$, where $|E|$ is the number of employees. Also, a random variable from uniform distribution $U\{1, \lceil \frac{|A|}{3} \rceil\}$ is generated for each employee that indicates the number of activities in the skill set of the employee. For flexibility L, $[l_a^{ac}, u_a^{ac}] = [6, 12]$, and $|A|$ is equal to $\frac{|E|}{10}$. Also, uniform distribution $U\{1, \lceil \frac{|A|}{5} \rceil\}$ is used to generate random variables. Finally,

based on the assigned numbers, activities are randomly selected as the skill set of each employee.

The third set of parameters is the daily flexibility set. In case the flexibility is H, $[l_e^{dp}, u_e^{dp}]$ is equal to $[8, 24]$ for all the employees, otherwise $[16, 16]$. The reason that we use I_1 improvement and I_2 improvement when the daily flexibility is H is that the number of feasible daily schedules increases. By using I_1 improvement, we cam decrease the daily schedule generation time, and I_2 improvement decreases the weekly schedule generation times.

The weekly flexibility parameters set is the fourth set we study in the experiments. If the flexibility is H, $[l^{wp}, u^{wp}]$, $[l^d, u^d]$, and u^{cd} are $[0, 7u_e^{dp}]$, $[0, 7]$, and 7, respectively. Else, they will be equal to $[4l_e^{dp}, 5u_e^{dp}]$, $[4, 5]$, and 4. The same demand curves as the industrial instances are used for these instances.

6.2.1 Low Daily Flexibility

The scheduling instances with daily flexibility L will be tested first. In this case, we do not use any of the improvements proposed in Section 4.5. Table 6.3 shows the parameters for each instance. The second and third columns, respectively, list the number of employees and activities. The number of activities is a function of the number of employees and activity parameters' flexibility. The following columns display how flexible each set of parameters is. The instances are generated in a way that as the instance number increases, the number of variables and constraints increases as well. For example, the number of variables and constraints for $R2$ are 59182 and 11780 while they are 102568 and 49540 for $R3$.

Tables 6.4 and 6.5 show the running times and objective cost of our algorithm and CPLEX for the random instances, respectively. In Table 6.4, if the CPLEX solver cannot find the optimal solution within the specified time limit, we use "T". The experiment setup is exactly the same as the industrial instances. The results show that for small instances, such as $R1$ and $R2$, with 10 employees and 1 activity both methods could find the optimal solution. However, CPLEX was much faster. For the instances $R3$ to $R5$ our method could solve the problem in a shorter time. However, due to the heuristic nature of LNS, it finds a local optimal solution and stops at that point. As the number of employees and flexibility increases, CPLEX fails to beat our algorithm

Table 6.3: Randomly generated low daily flexibility instances

Instance	$\|E\|$	$\|A\|$	Activity	Weekly	Daily
R1	10	1	L	H	L
R2	10	1	L	L	L
R3	10	4	H	H	L
R4	10	4	H	L	L
R5	45	4	L	H	L
R6	45	4	L	L	L
R7	45	11	H	H	L
R8	45	11	H	L	L
R9	80	8	L	H	L
R10	80	8	L	L	L
R11	80	18	H	H	L
R12	80	18	H	L	L

Table 6.4: Running times (in seconds) of the proposed algorithm and CPLEX solver for the low daily flexibility random instances

	Pro. Alg.		CPLEX	
Instance	Tour Sch. LNS	Task As. IP	Var. Lim.	1h Lim.
R1	124.9	0	**7.31**	**7.31**
R2	85.98	0	**8.67**	**8.67**
R3	**100.18**	**7.01**	T	T
R4	**125.65**	**3.29**	T	T
R5	**253.23**	**0**	T	T
R6	**193.11**	**0**	T	T
R7	**350.74**	**62.92**	T	T
R8	**739.72**	**73.69**	T	T
R9	**576.62**	**15.67**	T	T
R10	**848.64**	**14.89**	T	T
R11	**T**	**933.09**	T	T
R12	T	T	T	T

in terms of both speed and finding a solution in a reasonable time. As the results show, CPLEX could not find a solution for the instances $R8$ to $R12$ which have a higher number of employees. Another finding by comparing the instances $R5$ and $R6$ is that when the weekly flexibility is L and we have more strict constraints on the weekly parameters, the problem becomes much harder for CPLEX to solve.

Table 6.5: Objective cost of the proposed algorithm and CPLEX solver for the low daily flexibility random instances

			CPLEX	
Instance	Init. Cost	Pro. Alg.	Var. Lim.	1h Lim.
R1	1850	**870**	**870**	**870**
R2	1850	**1150**	**1150**	**1150**
R3	707	245	173	**171**
R4	707	257	513	**189**
R5	2828	980	1860	**614**
R6	2828	**836**	2772	1152
R7	4662	**1322**	4662	1390
R8	3660	**962**	3660	3360
R9	5656	**1414**	5656	5656
R10	6160	**580**	6160	6160
R11	8554	**2780**	6446	6446
R12	6160	**2098**	6160	6160

Table 6.6: Randomly generated high daily flexibility instances

Instance	$\|E\|$	$\|A\|$	Activity	Weekly	Daily
R13	10	1	L	H	H
R14	10	1	L	L	H
R15	10	4	H	H	H
R16	10	4	H	L	H
R17	45	4	L	H	H
R18	45	4	L	L	H
R19	45	11	H	H	H
R20	45	11	H	L	H
R21	80	8	L	H	H
R22	80	8	L	L	H
R23	80	18	H	H	H
R24	80	18	H	L	H

6.2.2 High Daily Flexibility

Next, the scheduling instances with daily flexibility H will be discussed, where the improvements I_1 and I_2 proposed in Section 4.5 will be used in the proposed algorithm. The Table 6.6 shows the parameters for each instance. Similar to Section 6.2.1 the instances are generated in a way that as the instance number increases, the number of variables and constraints increases as well.

The Tables 6.7 and 6.8 show the running time and objective cost of our algorithm and CPLEX for the random instances. The results show that in small instances, such as $R13$ to $R17$, with 10 and 40 employees and 1 activity, CPLEX could get a better solution in 1 hour

Table 6.7: Running times (in seconds) of the proposed algorithm and CPLEX solver for the high daily flexibility random instances

	Pro. Alg.		CPLEX	
Instance	Tour Sch. LNS	Task As. IP	Var. Lim.	1h Lim.
R13	**100.84**	**0**	T	T
R14	**55.3**	**0**	T	T
R15	**90.97**	**3.86**	T	275.64
R16	**122.98**	**8.02**	T	T
R17	**93.46**	**8.25**	T	T
R18	**126.41**	**6.37**	T	T
R19	**T**	**1724.79**	T	T
R20	**T**	**154.63**	T	T
R21	**1489.34**	**140.18**	T	T
R22	**T**	**82.52**	T	T
R23	M	-	T	T
R24	M	-	T	T

although our method was faster for most of the cases. The reason is that in addition to the heuristic characteristic of LNS, we use two improvement methods which remove some of the less promising feasible solutions. So, there is a higher chance of getting an optimal local solution. However, we can see the difference between the objective costs gets smaller as the size of the instance increases. This gap is much smaller for instances $R16$ and $R17$. For the remaining instances, i.e., R18 to R22, our algorithm could get a better solution faster. Also, CPLEX could not find a solution for the instances $R19$ to $R24$, where the number of employees and activities is large. The last two samples could not be solved by either method. The proposed algorithm faced memory issues when generating the grammar graphs for the employees. Due to a high number of employees with different skills, as the employees were generated randomly, the method will generate different graphs for each employee as none of them could be used by more than one employee. Also, the high daily flexibility makes the number of graphs larger as the method generates different graphs for each feasible daily duration.

Table 6.8: Objective cost of the proposed algorithm and CPLEX solver for the high daily flexibility random instances

			CPLEX	
Instance	Init. Cost	Pro. Alg.	Var. Lim.	1h Lim.
R13	704	128	704	**28**
R14	704	96	704	**48**
R15	707	133	125	**56**
R16	707	132	205	**128**
R17	6168	531	6168	**499**
R18	2816	**373**	2816	640
R19	6981	**2425**	6981	6981
R20	5065	**2322**	5065	5065
R21	11702	**1431**	11702	11702
R22	8840	**1117**	8840	8440
R23	26360	-	26360	26360
R24	26360	-	26360	26360

Chapter 7

Conclusions and Future Research Directions

7.1 Summary

This study explores multi-activity tour scheduling problems with flexible shifts and heterogeneous employees. The shifts in this problem can begin at any period, with upper and lower bounds of 8 and 24 periods, respectively, and a duration of 30 minutes for each period. The length of the shift affects how many break periods there will be. The shifts with fewer than 16 periods have one break, whereas the others have two. The start time of the break is flexible, but it must begin at least six periods after the start of the shift and must end before the last period of the shift. Also, there are restrictions on the duration of the activity, the daily and weekly working hours, and the consecutive working days. To solve the model to optimality, a tour scheduling IP model is first suggested in Chapter 3.

In Chapter 4, a heuristic method based on LNS has been created because the IP model cannot efficiently solve large-scale problems. Three steps are followed to create a feasible daily schedule. First, a CFG is used to create feasible timeslots. Grammar graphs are then used to mix timeslots and breaks to create feasible daily shifts. Finally, using RCSPP, weekly schedules are created from the daily schedules that have previously been created considering the weekly constraints as resources. In some cases with flexible working hours per day, the number of feasible daily shifts

could reach 500. Two heuristics are defined to select the shifts' most promising starting times first, and then select the most promising schedules for each possible length, in order to reduce computing time. After creating feasible weekly schedules, LNS is used to iteratively allocate schedules to the employees and refine the problem until a local optimal is reached.

In addition, a restriction that prohibits the repetition of tasks is added to the problem in Chapter 5. It is more practical for some working industries to assign people to each type of task once each day rather than restarting it numerous times. First, this condition is added to the suggested tour scheduling IP. The developed heuristic algorithm includes an extension because this constraint cannot be stated by a CFG. We fix the start and finish times of each employee's shifts, as well as the length of breaks and the set of activities that can be performed for each day, based on the local optimal solution from the LNS. Then, taking into account the constraint of work repetition, a new IP model is utilized to assign tasks to employees.

In Chapter 6, both customer instances and randomly generated instances are used in computational research. In our algorithm, each LNS and task assignment IP is given a 30-minute time limit, and CPLEX solves the tour scheduling IP within a 1-hour time limit. The findings demonstrate that as the flexibility and size of the instance increase, the developed heuristic approach performs superior to the tour scheduling IP and produces better results in terms of solution time and solution quality. Indeed, for the industrial instances, our algorithm is faster and in all instances we could find much better solutions than CPLEX. For the random instances, for some smaller instances with fewer employees and activities, CPLEX can obtain a better objective cost within the allotted time frame. While our algorithm can find a high-quality solution, CPLEX struggles to do so when the number of employees and activities becomes large.

7.2 Future Research Directions

When a local optimal solution is identified or the time limit is reached, the search in the developed algorithm is over. The initial solution at the beginning of the search determines the local optimal the algorithm can reach. In our method, we start searching from an empty solution where no one is assigned to any schedule. By employing a technique to produce a better initial solution for

the search, a better objective cost could be reached. Additionally, since the developed algorithm is fast, it stops even before the considered time limit for most of the instances. Each time the search finds a locally optimal solution, we can restart the search to look for a better solution. A different initial solution should be taken into account each time the search is restarted. By making use of these advances, we could find solutions closer to the global optimal.

The improvements we outlined in Section 4.5 and use for high daily flexibility instances could be an aspect to investigate for better performance. Using the first improvement, we skip generating some daily schedules with specific starting times. We delete the less promising daily schedules using the second improvement before generating the weekly schedules. These improvements decrease both the computational time and solution quality. We can reduce the time required for daily schedule generation and RCSPP while maintaining the solution quality by suggesting better improvements than the ones already proposed.

Last but not least, a machine learning technique may be used to forecast the schedule generation and RCSPP time based on the problem's parameters and guide us in selecting from the developed improvements to reduce the computing effort while maintaining a good level of solution quality.

Appendix A

Chapter 3 Supplement

Tour Scheduling IP

$$\min \quad c_s \sum_{i \in I} \sum_{p \in P} \sum_{a \in A} \underline{u}_{i,p,a} + c_e \sum_{i \in I} \sum_{p \in P} \sum_{a \in A} \overline{u}_{i,p,a}$$

$$\text{s.t.} \quad \sum_{i \in I} w_{e,i} \leq u^d h_e \qquad \forall e \in E \ (1)$$

$$l^d h_e \leq \sum_{i \in I} w_{e,i} \qquad \forall e \in E \ (2)$$

$$w_{e,i} = \sum_{s \in S} z_{e,s,i} \qquad \forall i \in I, e \in E \ (3)$$

$$\sum_{i'=i-u^{cd}}^{i} w_{e,i'} \leq u^{cd} \qquad \forall i \in I, e \in E \ (4)$$

$$\sum_{p\in P}\sum_{i\in I} x_{e,i,a,p} = 0 \qquad \forall e \in E, a \notin A_e \ (5)$$

$$z_{e,s,i} \leq \sum_{p=\underline{l}_s+6}^{\underline{l}_s-lb_s} y_{e,i,p,lb_s} \qquad \forall i \in I, e \in E, s \in S \ (6)$$

$$\sum_{p\in P}\sum_{b\in B} y_{e,i,p,b} \leq 1 \qquad \forall i \in I, e \in E \ (7)$$

$$\sum_{k\in K} x_{e,i,a,p} + \underline{u}_{i,p,a} = \overline{u}_{i,p,a} + d_{i,p,a} \qquad \forall i \in I, a \in A, p \in P \ (8)$$

$$\sum_{a\in A} x_{e,i,a,p} = \sum_{s\in S} wp_{p,s} z_{e,s,i} - \sum_{b\in B}\sum_{d=1}^{min(b,p)} y_{e,i,p-d+1,b} \qquad \forall i \in I, e \in E, p \in P \ (9)$$

$$t_{e,i,a,p} + x_{e,i,a,p-1} \leq 1 \qquad \forall i \in I, e \in E, a \in A, p \in [2, p_n - l_a^{ac} + 1] \ (10)$$

$$t_{e,i,a,p} - x_{e,i,a,p} \leq 0 \qquad \forall i \in I, e \in E, a \in A, p \in [1, p_n - l_a^{ac} + 1] \ (11)$$

$$x_{e,i,a,p} - x_{e,i,a,p-1} - t_{e,i,a,p} \leq 0 \qquad \forall i \in I, e \in E, a \in A, p \in [2, p_n - l_a^{ac} + 1] \ (12)$$

$$x_{e,i,a,1} - t_{e,i,a,1} \leq 0 \qquad \forall i \in I, e \in E, a \in A \ (13)$$

$$l_a^{ac} t_{e,i,a,p} - \sum_{p'=p}^{p+l_a^{ac}-1} x_{e,i,a,p'} \leq 0 \qquad \forall i \in I, e \in E, a \in A, p \in [1, p_n - l_a^{ac} + 1] \ (14)$$

$$\sum_{p'=p}^{p+u_a^{ac}} x_{e,i,a,p'} \leq u_a^{ac} \qquad \forall i \in I, e \in E, a \in A, p \in [1 : p_n - u_a^{ac}] \ (15)$$

$$\sum_{i\in I}\sum_{s\in S} l_s z_{e,s,i} \leq u^{wp} \qquad \forall e \in E \ (16)$$

$$l^{wp} h_e \leq \sum_{i\in I}\sum_{s\in S} l_s z_{e,s,i} \qquad \forall e \in E \ (17)$$

$$l_e^{dp} w_{e,i} \leq \sum_{s\in S} l_s z_{e,s,i} \qquad \forall i \in I, e \in E \ (18)$$

$$\sum_{s \in S} l_s z_{e,s,i} \leq l_e^{dp} \quad \forall i \in I, e \in E \ (19)$$

$$z_{e,s,i} \in \{0,1\} \quad \forall i \in I, e \in E, s \in S \ (20)$$

$$w_{e,i} \in \{0,1\} \quad \forall i \in I, e \in E \ (21)$$

$$y_{e,i,p,b} \in \{0,1\} \quad \forall i \in I, e \in E, p \in P, b \in B \ (22)$$

$$t_{e,i,a,p} \in \{0,1\} \quad \forall i \in I, e \in E, a \in A, p \in [1 : p_n - l_a^{ac} + 1] \ (23)$$

$$x_{e,i,a,p} \in \{0,1\} \quad \forall i \in I, e \in E, a \in A, p \in P \ (24)$$

$$h_e \in \{0,1\} \quad \forall e \in E \ (25)$$

$$\underline{u}_{i,p,a} \in Z^+ \quad \forall i \in I, e \in E, p \in P \ (26)$$

$$\overline{u}_{i,p,a} \in Z^+ \quad \forall i \in I, e \in E, p \in P \ (27)$$

Appendix B

Chapter 3 Supplement

Example 1

Consider a three-employee tour scheduling problem. Assume that the store is open for 8 hours. So, $h \in \{1, 2, ..., 18\}$. Additionally, this store has two activities that each have a minimum and maximum time limit of 2 and 4 periods, respectively. Assume that employee 1 is skilled for activity 1, employee 2 is skilled for activity 2, and employee 3 is skilled for both activities. The minimum and maximum working periods per day are 8 and 24, the minimum and maximum working periods per week are 16 and 120, and the minimum and maximum working days per week are 2 and 5. There can be no more than five consecutive days in a week. Assume that none of the employees worked the week before. All 45 eligible shifts with a duration of 8 to 16 periods are listed in Table A2.1. Table A2.2 lists the sets and parameters.

Table A2.1: Parameters for each enumerated shift for each duration between 8 to 16 periods and each start period

Enumerated Shift	Start Period	End Period	Length (periods)	Break Length (period)
1	1	8	8	1
2	2	9	8	1
⋮	⋮	⋮	⋮	⋮
9	9	16	8	1
10	1	9	9	1
11	2	10	9	1
⋮	⋮	⋮	⋮	⋮
17	8	16	9	1
⋮	⋮	⋮	⋮	⋮
45	1	16	16	2

Table A2.2: Sets and parameters for the IP model

Sets	
I	$\{1, 2, \ldots, 7\}$
S	$\{1, 2, \ldots, 45\}$
P	$\{1, 2, \ldots, 16\}$
A	$\{1, 2\}$
A_e	$A_1 = \{1\}$ $A_2 = \{2\}$ $A_3 = \{1, 2\}$
E	$\{1, 2, 3\}$
B	$\{1, 2\}$
Parameters	
p_n	16
$wp_{p,s}$	e.g. $wp_{p,1}$ is 1 $\forall p \in \{1, 2, ..., 8\}$, otherwise 0
$[l_a^{ac}, u_a^{ac}]$	$[2, 4]$ $\forall a \in A$
$[l_e^{dp}, u_e^{dp}]$	$[8, 24]$ $\forall e \in E$
$[l^{wp}, u^{wp}]$	$[16, 120]$
$[l^d, u^d]$	$[2, 5]$
u^{cd}	5
$w_{e,i}$	0 $\forall e \in E, i \in \{-4, -3, \ldots, 0\}$

Appendix C

Chapter 4 Supplement

Chumsky Normal Form

$$S \implies W_1R|RW_1|RZ_1|RW_2|W_2R|RZ_2|RR \tag{1}$$

$$Z_1 \implies W_1R \tag{2}$$

$$W_{1[a_1,b_1]} \implies P_{[6,b_1-2]}Y_1 \tag{3}$$

$$Y_1 \implies XP \tag{4}$$

$$Z_2 \implies W_2R \tag{5}$$

$$W_{2[a_2,b_2]} \implies P_{[6,b_2-3]}Y_2 \tag{6}$$

$$Y_2 \implies LP \tag{7}$$

$$L \implies XX \tag{8}$$

$$X \implies b \tag{9}$$

$$R \implies RR|r \tag{10}$$

$$P \implies A'_{a[l_a^{ac},u_a^{ac}]}P'_a|A'_{a[l_a^{ac}-1,u_a^{ac}-1]}A'_{a[1,1]} \qquad \forall a \in A_e \tag{11}$$

$$P \implies a \qquad \forall a \in \{a|\ l_a^{ac} = 1, a \in A_e\} \tag{12}$$

$$A'_a \implies A'_aA'_a|a \qquad \forall a \in A_e \tag{13}$$

$$P'_a \implies A'_{a'[l_{a'}^{ac},u_{a'}^{ac}]}P'_{a'}|A'_{a'[l_{a'}^{ac}-1,u_{a'}^{ac}-1]}A'_{a'[1,1]} \qquad \forall a, a' \in A_e,\ a' \neq a \tag{14}$$

$$P'_a \implies a' \qquad \forall a, a' \in A_e,\ a' \neq a,\ a' \in \{a' |\ l^{ac}_{a'} = 1\} \quad (15)$$

Appendix D

Chapter 5 Supplement

Task Assignment IP

$$\min \quad c_s \sum_{i\in I}\sum_{p\in P}\sum_{a\in A} \underline{u}_{i,p,a} + c_e \sum_{i\in I}\sum_{p\in P}\sum_{a\in A} \overline{u}_{i,p,a}$$

$$\text{s.t.} \quad \sum_{e\in E}\sum_{p\in P}\sum_{i\in I}\sum_{a\notin A_{e,i}} x_{e,i,a,p} = 0 \tag{1}$$

$$\sum_{k}\sum_{i\notin W_e}\sum_{a\notin A_{e,i}}\sum_{p=1}^{p_n - l_a^{ac}+1} t_{e,i,a,p} = 0 \tag{2}$$

$$\sum_{e\in E}\sum_{i\notin W_e}\sum_{a\in A}\sum_{p\in P} x_{e,i,a,p} = 0 \tag{3}$$

$$\sum_{e\in E}\sum_{i\in I}\sum_{a\in A}\Big(\sum_{p=1}^{ss_{e,i}-1} x_{e,i,a,p} + \sum_{p=se_{e,i}+1}^{p_n} x_{e,i,a,p} + \sum_{p\in B_{e,i}} x_{e,i,a,p}\Big) = 0 \tag{4}$$

$$\sum_{e\in E}\sum_{i\in W_e}\sum_{a\in A}\Big(\sum_{p=1}^{min(ss_{e,i}-1,p_n-l_a^{ac}+1)} t_{e,i,a,p} + \sum_{p=se_{e,i}+1}^{p_n-l_a^{ac}+1} t_{e,i,a,p} + \sum_{p\in B_{e,i},\ p\le p_n-l_a^{ac}+1} t_{e,i,a,p}\Big) = 0 \tag{5}$$

$$\sum_{e\in E} x_{e,i,a,p} + \underline{u}_{i,p,a} = \overline{u}_{i,p,a} + d_{i,a,p} \qquad \forall i \in I, p \in P, a \in A \ (6)$$

$$\sum_{j} x_{e,i,a,p} = 1 \qquad \forall e \in E, i \in W_e, p \in [ss_{e,i} : se_{e,i}], p \notin B_{e,i} \ (7)$$

$$t_{e,i,a,p} + x_{e,i,a,p-1} \le 1 \qquad \forall e \in E, i \in W_e, a \in A, p \in [2 : p_n - l_a^{ac} + 1] \ (8)$$

$$t_{e,i,a,p} - x_{e,i,a,p} \le 0 \qquad \forall e \in E, i \in W_e, a \in A, p \in [1 : p_n - l_a^{ac} + 1] \ (9)$$

$$x_{e,i,a,p-1} + t_{e,i,a,p} - x_{e,i,a,p} \ge 0 \qquad \forall e \in E, i \in W_e, a \in A, p \in [2 : p_n - l_a^{ac} + 1] \ (10)$$

$$t_{e,i,a,1} - x_{e,i,a,1} \ge 0 \qquad \forall e \in E, i \in W_e, a \in A \ (11)$$

$$l_a^{ac} t_{e,i,a,p} - \sum_{p'=p}^{p+l_a^{ac}-1} x_{e,i,a,p'} \le 0 \qquad \forall e \in E, i \in W_e, a \in A, p \in [1 : p_n - l_a^{ac} + 1] \ (12)$$

$$\sum_{p'=p}^{p+u_a^{ac}} x_{e,i,a,p'} \le u_a^{ac} \qquad \forall e \in E, i \in W_e, a \in A, p \in [1 : p_n - u_a^{ac}] \ (13)$$

$$x_{e,i,a,B_{e,i}(1)-1} \ge c_{e,i,a} \qquad \forall e \in E, i \in W_e, a \in A \ (14)$$

$$t_{e,i,a,B_{e,i}(-1)+1} \ge c_{e,i,a} \qquad \forall e \in E, i \in W_e, a \in A \ (15)$$

$$\sum_{p=1}^{p_n-l_a^{ac}+1} t_{e,i,a,p} - c_{e,i,a} \le 2p_n \times r_{e,i} + 1 \qquad \forall e \in E, i \in W_e, a \in A \ (16)$$

$$t_{e,i,a,p} \in \{0,1\} \qquad \forall e \in E, i \in W_e, a \in A, p \in [1 : p_n - l_a^{ac} + 1] \ (17)$$

$$\forall e \in E, i \in W_e, a \in A, p \in P \quad (18)$$

$$\forall i \in W_e, a \in A, p \in P \quad (19)$$

$$\forall i \in W_e, a \in A, p \in P \quad (20)$$

$$\forall e \in E, i \in W_e, a \in A \quad (21)$$

$$x_{e,i,a,p} \in \{0,1\}$$

$$\underline{u}_{i,p,a} \in Z^+$$

$$\overline{u}_{i,p,a} \in Z^+$$

$$c_{e,i,a} \in \{0,1\}$$

Appendix E

Chapter 5 Supplement

Example 2

Take into account a weekly scheduling problem with two employees. The store will be open for 8 hours. So, $p \in \{1, 2, ..., 16\}$. Additionally, this store has two activities, each with minimum and maximum durations of 2 and 4 periods, respectively. Assume that employee one is skilled in activity 1 and that employee two is skilled in activities 1 and 2. The following schedules are the outputs of the LNS algorithm used to solve the weekly scheduling problem. Employee 1 works the first 5 days from periods 1 to 11, and the break period is 7 for each of the 5 days. Employee 2 works periods 2 through 14 on days 6 and 7, and the break period is 8 for both days. This employee completes task 1 prior to the break and 2 following it. The parameters and indices are indicated in Table A5.1 for the task assignment IP model.

Table A5.1: Sets, indices, parameters, and decision variables for the task assignment IP

Sets	
I	$\{1, 2, ..., 7\}$
P	$\{1, 2, ..., 16\}$
A	$\{1, 2\}$
E	$\{1, 2\}$
W_e	$W_1 = \{1, 2, ..., 5\}$ $W_2 = \{6, 7\}$
$B_{e,i}$	$B_{1,i} = \{7\}\ \forall i \in W_1$ $B_{2,i} = \{8\}\ \forall i \in W_2$
$A_{e,i}$	$A_{1,i} = \{1\}\ \forall i \in W_1$ $A_{2,i} = \{1, 2\}\ \forall i \in W_2$
Parameters	
p_n	16
$ss_{e,i}$	$ss_{1,i} = 1\ \forall i \in W_1$ $ss_{2,i} = 2\ \forall i \in W_2$
$se_{e,i}$	$se_{1,i} = 11\ \forall i \in W_1$ $se_{2,i} = 14\ \forall i \in W_2$
$[l_a^{ac}, u_a^{ac}]$	$[2, 4]\ \forall a \in A$
$r_{e,i}$	$0\ \forall e \in E, i \in I$

Bibliography

Ahmadi, S., Tack, G., Harabor, D. D., and Kilby, P. (2021). A fast exact algorithm for the resource constrained shortest path problem. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 14, 12217–12224.

Aykin, T. (1996). Optimal shift scheduling with multiple break windows. *Management Science* 42(4), 591–602.

Aykin, T. (1998). A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *Journal of the Operational Research Society* 49(6), 603–615.

Bailey, J. (1985). Integrated days off and shift personnel scheduling. *Computers & Industrial Engineering* 9(4), 395–404.

Baker, K. R. (1976). Workforce allocation in cyclical scheduling problems: A survey. *Journal of the Operational Research Society* 27(1), 155–167.

Bechtold, S. E. and Jacobs, L. W. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science* 36(11), 1339–1351.

Bechtold, S. E. and Jacobs, L. W. (1996). The equivalence of general set-covering and implicit integer programming formulations for shift scheduling. *Naval Research Logistics* 43(2), 233–249.

Bhulai, S., Koole, G., and Pot, A. (2008). Simple methods for shift scheduling in multiskill call centers. *Manufacturing & Service Operations Management* 10(3), 411–420.

Billionnet, A. (1999). Integer programming to schedule a hierarchical workforce with variable demands. *European Journal of Operational Research* 114(1), 105–114.

Bonutti, A., Ceschia, S., De Cesco, F., Musliu, N., and Schaerf, A. (2017). Modeling and solving a real-life multi-skill shift design problem. *Annals of Operations Research* 252(2), 365–382.

Boyer, V., Gendron, B., and Rousseau, L.-M. (2014). A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. *Journal of Scheduling* 17(2), 185–197.

Brunner, J. O. and Bard, J. F. (2013). Flexible weekly tour scheduling for postal service workers using a branch and price. *Journal of scheduling* 16(1), 129–149.

Brusco, M. J. and Jacobs, L. W. (1993). A simulated annealing approach to the solution of flexible labour scheduling problems. *Journal of the Operational Research Society* 44(12), 1191–1200.

Brusco, M. J. and Jacobs, L. W. (2000). Optimal models for meal-break and start-time flexibility in continuous tour scheduling. *Management Science* 46(12), 1630–1641.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control* 2(2), 137–167.

Cocke, J. (1969). *Programming languages and their compilers: Preliminary notes*. New York University.

Côté, M.-C., Gendron, B., Quimper, C.-G., and Rousseau, L.-M. (2011a). Formal languages for integer programming modeling of shift scheduling problems. *Constraints* 16(1), 54–76.

Côté, M.-C., Gendron, B., and Rousseau, L.-M. (2011b). Grammar-based integer programming models for multiactivity shift scheduling. *Management Science* 57(1), 151–163.

Côté, M.-C., Gendron, B., and Rousseau, L.-M. (2013). Grammar-based column generation for personalized multi-activity shift scheduling. *INFORMS Journal on Computing* 25(3), 461–474.

Dantzig, G. B. (1954). A comment on Edie's "Traffic delays at toll booths". *Journal of the Operations Research Society of America* 2(3), 339–341.

Demassey, S., Pesant, G., and Rousseau, L.-M. (2005). Constraint programming based column generation for employee timetabling. In: *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer, 140–154.

Demassey, S., Pesant, G., and Rousseau, L.-M. (2006). A cost-regular based hybrid column generation approach. *Constraints* 11(4), 315–333.

Detienne, B., Péridy, L., Pinson, É., and Rivreau, D. (2009). Cut generation for an employee timetabling problem. *European Journal of Operational Research* 197(3), 1178–1184.

Easton, F. F. and Rossin, D. F. (1991). Sufficient working subsets for the tour scheduling problem. *Management Science* 37(11), 1441–1451.

Edie, L. C. (1954). Traffic delays at toll booths. *Journal of the Operations Research Society of America* 2(2), 107–138.

Gaspero, L. D., Gärtner, J., Musliu, N., Schaerf, A., Schafhauser, W., and Slany, W. (2010). A hybrid LS-CP solver for the shifts and breaks design problem. In: *International Workshop on Hybrid Metaheuristics*. Springer, 46–61.

Gaspero, L. D., Gärtner, J., Musliu, N., Schaerf, A., Schafhauser, W., and Slany, W. (2013). Automated shift design and break scheduling. In: *Automated Scheduling and Planning*. Springer, 109–127.

Gérard, M., Clautiaux, F., and Sadykov, R. (2016). Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce. *European Journal of Operational Research* 252(3), 1019–1030.

Hernández-Leandro, N. A., Boyer, V., Salazar-Aguilar, M. A., and Rousseau, L.-M. (2019). A matheuristic based on Lagrangian relaxation for the multi-activity shift scheduling problem. *European Journal of Operational Research* 272(3), 859–867.

Hojati, M. and Patil, A. S. (2011). An integer linear programming-based heuristic for scheduling heterogeneous, part-time service employees. *European Journal of Operational Research* 209(1), 37–50.

Huangfu, Q. and Hall, J. J. (2018). Parallelizing the dual revised simplex method. *Mathematical Programming Computation* 10(1), 119–142.

Jacobs, L. W. and Brusco, M. J. (1996). Overlapping start-time bands in implicit tour scheduling. *Management Science* 42(9), 1247–1259.

Jones, C. and Nolde, K. (2013). Demand driven employee scheduling for the swiss market. *Technical Report. Automatic Control Laboratory, EPFL*.

Kabak, Ö., Ülengin, F., Aktaş, E., Önsel, Ş., and Topcu, Y. I. (2008). Efficient shift scheduling in the retail sector through two-stage optimization. *European Journal of Operational Research* 184(1), 76–90.

Kasami, T. (1966). An efficient recognition and syntax-analysis algorithm for context-free languages. *University of Illinois at Urbana-Champaign Coordinated Science Laboratory Report no. R-257*.

Mehrotra, A., Murphy, K. E., and Trick, M. A. (2000). Optimal shift scheduling: A branch-and-price approach. *Naval Research Logistics* 47(3), 185–200.

Meisels, A. and Schaerf, A. (2003). Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence* 39(1), 41–59.

Moondra, S. L. (1976). An LP model for work force scheduling for banks. *Journal of Bank Research* 7(4), 299–301.

Musliu, N., Schaerf, A., and Slany, W. (2004). Local search for shift design. *European Journal of Operational Research* 153(1), 51–64.

Pan, S., Akplogan, M., Létocart, L., Touati, N., and Calvo, R. W. (2016). Solving a multi-activity shift scheduling problem with a TABU search heuristic. In: *PATAT 2016: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling*, 317–326.

Pan, S. (2018). Exact and Heuristic Methods for Multi-Activity Tour Scheduling Problems. PhD thesis. Sorbonne Paris Cité.

Pan, S., Akplogan, M., Touati, N., Létocart, L., Calvo, R. W., and Rousseau, L.-M. (2018). A hybrid heuristic for the multi-activity tour scheduling problem. *Electronic Notes in Discrete Mathematics* 69, 333–340.

Pastor, R. and Olivella, J. (2008). Selecting and adapting weekly work schedules with working time accounts: A case of a retail clothing chain. *European Journal of Operational Research* 184(1), 1–12.

Pesant, G. (2004). A regular language membership constraint for finite sequences of variables. In: *International Conference on Principles and Practice of Constraint Programming*. Springer, 482–495.

Pisinger, D. and Ropke, S. (2010). Large neighborhood search. In: *Handbook of metaheuristics*. Springer, 399–419.

Qu, Y. and Curtois, T. (2020). Solving the multi-activity shift scheduling problem using variable neighbourhood search. In: *Proceedings of the 9th International Conference on Operations*

Research and Enterprise Systems - ICORES, INSTICC. SciTePress, 227–232. ISBN: 978-989-758-396-4.

Quimper, C.-G. and Rousseau, L.-M. (2010). A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics* 16(3), 373–392.

Quimper, C.-G. and Walsh, T. (2007). Decomposing global grammar constraints. In: *International Conference on Principles and Practice of Constraint Programming*. Springer, 590–604.

Restrepo, M. I., Gendron, B., and Rousseau, L.-M. (2016). Branch-and-price for personalized multiactivity tour scheduling. *INFORMS Journal on Computing* 28(2), 334–350.

Restrepo, M. I., Gendron, B., and Rousseau, L.-M. (2018). Combining Benders decomposition and column generation for multi-activity tour scheduling. *Computers & Operations Research* 93, 151–165.

Restrepo, M. I., Lozano, L., and Medaglia, A. L. (2012). Constrained network-based column generation for the multi-activity shift scheduling problem. *International Journal of Production Economics* 140(1), 466–472.

Seçkiner, S. U., Gökçen, H., and Kurt, M. (2007). An integer programming model for hierarchical workforce scheduling problem. *European Journal of Operational Research* 183(2), 694–699.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *International Conference on Principles and Practice of Constraint Programming*. Springer, 417–431.

Talarico, L. and Duque, P. A. M. (2015). An optimization algorithm for the workforce management in a retail chain. *Computers & Industrial Engineering* 82, 65–77.

Thompson, G. M. (1995). Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science* 41(4), 595–607.

Al-Yakoob, S. M. and Sherali, H. D. (2008). A column generation approach for an employee scheduling problem with multiple shifts and work locations. *Journal of the Operational Research Society* 59(1), 34–43.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and Control* 10(2), 189–208.