Optimize Coverage for Employee Rostering Problem

Assignment	Final Design Specification (FDS)
Team Designation	CD1
Client	Ceridian / Eddie Hsu
Supervisor	Professor Merve Bodur
Team Members	Jiahua Chen (1003897955)
	Jiaru Li (1003791523)
	Sijia Li (1004073904)
	Mingkun Wang (1003803404)
Date	April 1, 2022

Contents

1 Executive Summary

Ceridian (the client) is a global human capital management (HCM) software company that provides work intelligence solutions and services for all kinds of organizations. It currently uses a modification of Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristics to solve employee rostering problems. The current algorithm is capable of generating feasible schedules, but it has no guarantee of solution quality. Therefore, the client is looking to explore alternative algorithms that can either outperform GRASP in solution quality within a reasonable amount of time or be embedded to help improve GRASP (the baseline).

The goal of the project is to explore, research, and develop a rostering algorithm that is able to give a better solution quality. The ideal output should be a roster developed over a 7-day planning horizon, based on employee availabilities, employee skill sets, appropriate shift lengths, number of consecutive work days, and so forth.

While performing the literature review, the capstone team explored a variety of methodologies used to solve rostering problems, such as mathematical programming, constraint programming, and multi-agent approaches. One or a combination of these methodologies may be adapted and incorporated into the design of new employee rostering algorithm(s).

An integer programming (IP) model is formulated to create seven-day rostering schedules. The model aims to minimize the demand coverage penalty (from overstaffing and understaffing) while enforcing a set of selected fundamental scheduling constraints, such as minimum time between shifts and maximum daily hours.

Two synthetic local diner datasets and one real cosmetics store dataset are used to generate rostering schedules and evaluate the IP model. Compared to the baseline model (GRASP), the IP model outperforms GRASP in terms of roster quality, guaranteeing optimal rosters created. Although similar performances are achieved on smaller problem instances in terms of computational speed for both GRASP and IP model, GRASP outperforms the IP model for larger problem instances. This makes sense as GRASP is a heuristic algorithm that searches for a local optimal value in a small neighbourhood, whereas the IP model has a much larger search space and finds a global optimal value.

In conclusion, the IP model does not outperform GRASP completely. The capstone team recommends leveraging the IP model for smaller problem instances and GRASP for larger and more complex problem instances. Therefore, decision makers have to make a trade-off between solution quality and computational speed depending on problem sizes.

For future work, the IP model could be embedded into the GRASP algorithm in the client's platform to serve as a warm start. Additionally, some hybrid models could be investigated to solve the optimization problem on relaxed settings followed by a heuristic algorithm to improve the obtained solution. More work could be done to take employee preferences into account when generating rosters to maximize employee satisfaction, as well as leveraging inverse optimization to solve the rostering problem from the end-user's perspective.

2 Relevant Terminologies

Some terminologies used in the capstone project document are defined below.

- Constraints (in optimization): The conditions that must be met in the possess of solving the optimization problem.
- Coverage: The extent to which the number of employees assigned to a specific shift meets the staffing levels. If there are more employees assigned than the staffing levels required, it is considered as over-staffing; if fewer are assigned, it is considered as under-staffing.
- Decision Variables: Unknown quantities defined by the decision makers to suggest decisions in the problem.
- Integer Programming: An optimization problem with linear objective functions subject to a set of linear constraints over integer decision variables only.
- Linear Programming: A method to achieve the optimal solution using a mathematical model which represents relationships using linear functions.
- Multi-agent: A system composed of multiple interacting intelligent agents, which may solve problems that are difficult for a single agent.
- Rostering: The process of assigning employees to **flexible** shifts that meet the staffing levels. This includes specific date, **flexible** start and end times, and task(s) in duration of the shift.
- Scheduling: The process of assigning employees to **pre-defined** shifts that meet staffing levels. This includes specific date, **fixed** start and end times, and task(s) in duration of the shift.
- Objective (or Objective Function): The mathematical equation that describes the target of the problem, either to maximize or to minimize a concept defined by a numerical value and decision variables. The value of the equation is called Objective Function Value.

3 Introduction

3.1 Background and Problem Statement

Ceridian (the client) is a global human capital management (HCM) software company that provides work intelligence solutions and services for organizations of all sizes and from multiple industries, such as retail, manufacturing, and healthcare.

One of the client's well-known Enterprise HCM Software is Dayforce. It eases workforce management, increases operational efficiency, and helps improve compliance with complex labour rules [1]. Customized to each customer's needs, Dayforce allows its customers to specify their own rostering rules, either by selecting from an existing list of rules or by creating their own. Depending on the rules, values for attributes such as evaluation period, minimum/maximum threshold, and pay adjustments are entered. After all rules are defined, the application will generate a schedule per employee across the specified planning horizon based on employee availability and capability.

Currently, the rostering algorithm employed in Dayforce is a modification of the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristics. It is the base form of multi-start metaheuristic for combinatorial optimization problems, in which each iteration consists of three phases: Construction, Enforce Minimums, and Improvement (elaborated in Section 5.1) [2].

Although GRASP is capable of generating feasible schedules, it can be slow, computationally expensive, and has no guarantee of solution quality. Therefore, the client is looking to explore alternative algorithms that can potentially replace GRASP partially or completely.

The aim of the project is to explore, research, and develop a rostering algorithm used to plan human resources that is able to give a better solution quality. The proposed solution will aim to minimize the roster's demand coverage penalty (overstaffing and understaffing) and either outperform GRASP in quality within a reasonable amount of time or be embedded to help improve GRASP (the baseline).

Roster optimization will be made based on the shift demands from the client's customers. In addition, a set of selected fundamental rostering constraints including employees' time availabilities, employees' skillsets, legal constraints (e.g., maximum employee working hours), and union regulations (e.g., minimum time between two shifts) will be enforced (elaborated in Section 5.2.3).

The ideal output should be a roster developed for a 7-day planning horizon. The roster shall indicate the start time and end time of each shift, work to be done (given by the organizations), and which employee will be responsible (generated) on each day in the planning horizon.

As this is more of an exploratory work, the main focus of the project is on researching, developing, and testing new rostering algorithms. Integrating the final algorithm(s) to the client's existing platform is considered out-of-scope as the client is not looking for a production ready solution.

The real-world impact of the project is to discover potential rostering algorithms that could either replace or help improve the current in-production algorithm so that the client does not need to invest significant human resources, money, and time themselves in exploring alternative methods. High quality rosters can have major impacts on the company's productivity: ensured coverage, reduced costs, and high employees' satisfaction [3]. Furthermore, rostering plays an important role in many areas, for example, hospitals, schools, and convenience stores. In addition, human capital management software companies, like Ceridian, and their customers would benefit from our exploration results. The results could provide insights into utilizing new technologies to further improve their product software.

3.2 Changes from Project Requirements

The team changed focus from exploring the multi-agent heuristic approach to exploring the integer programming model. The original objective was to tackle and improve all three phases of GRASP using the multi-agent heuristic approach. However, it was later deemed that the project scope was too large for a capstone project given the eight-month time limitation, and the project complexity was high due to it being a blackbox optimization problem. In the original setting, the constraints were a blackbox (unknown) to the algorithm as there were too many of them, so the algorithm had to interrogate a compliance engine to check if they were compliant.

A mutual agreement was reached by the client, the professor, and the team to narrow the scope to improving the schedule created in the first phase of GRASP (Construction). Accordingly, the project objective has changed to developing an alternative algorithm that can either outperform GRASP in quality within a reasonable amount of time or be embedded to help improve GRASP. In addition, instead of tackling all constraints, only a set of fundamental rostering constraints are selected to be enforced in the algorithm as a proof of concept.

4 Solutions Selection

4.1 Literature Review

Literature review is conducted to acknowledge methods employed in relevant areas and generate ideas for this project. Due to the computational nature of the problem and the recommended heuristic approach by the client, the focus will be primarily on mathematical programming (MP), constraint programming (CP), artificial intelligence approaches (AI), and metaheuristics (MH). Practices by competitors are also explored to ideate more solution alternatives to tackle this problem. The pros and cons of each are shown in Table 1 [4, 5, 6, 7, 8, 9].

Approach	Pros	Cons
Mathematical Programming (MP)	• Guaranteed optimal solution • Advanced solvers available	• Context-specific, generalized robust model hard to find • Computationally expensive in resource required and time • Risk of no solutions
Constraint Programming (CP)	• Promising in highly fixed and restricted rostering work • Can produce feasible solutions	• Feasible but inefficient solutions • Not generalizable to different contexts
Artificial Intelligence (AI)	• Traditional AI: incorporate human factors in rostering • Latest AI techniques (e.g. Reinforcement Learning): model robustness	• No guarantee on solution quality • Can have high computational time
Metaheuristics (MH)	• Great solving ability for all levels of optimization problems • Model robustness and easy implementation • Relatively fast computational time	• No guarantee on solution quality • Computational time prone to problem size

Table 1: Summary of Potential Approaches

In rostering problems, MP is guaranteed to produce an optimal solution. It usually formulates the problem into linear programming (LP) and mixed integer programming (MIP). The problem is formulated with rostering targets defined by decision variables as objective function, as well as rostering requirements as boundary constraints of decision variables. The MP algorithm then starts to search for feasible solution point(s) in this defined feasible region and try to find the optimal solution based on defined objective function values. This approach generally gives an optimal result, but suffers from great difficulties from two major perspectives. First, MP is computationally expensive and time consuming with the risk of no solutions. With all the boundary constraints, there exists the possibility that there is no feasible data point that can be employed as a solution. On the other hand, the problem formulation can be context specific and inflexible, making a generalized robust formulation of rostering problems impossible [10]. Practitioners would have to often adjust the model formulations to fit it into different scenarios. To address this problem, a simplified implementation approach is often employed to produce optimal results for comparison [10].

CP is particularly promising in highly constrained rostering contexts. With the start time, end time, and type of job fixed, the algorithm simply assigns work shifts to available employees to fill task demands. However, compared to MP, CP usually produces feasible but inefficient (suboptimal) solutions. Moreover, it loses the generalizability and flexibility in problem formulation, which makes it impossible to become a robust rostering approach to all types of rostering tasks.

Traditional AI approaches seen in rostering problems are in the form of decision support systems, with degrees of control ranging from manual rostering to full automation [11] [12]. These unique, flexible approaches can incorporate human factors considerations, such as individual preferences, which are difficult to be hard-coded in software solutions. It can also help increase decision makers' trust in the model. However, as the degree of automation increases, the computational time and the generalizability of traditional AI approach decreases. More recent AI approaches try to utilize more advanced modelling techniques such as reinforcement learning, and they have shown great prospects in terms of model robustness [8]. Though recent AI approaches seem promising, related research work only focuses on specific rostering or scheduling problems, making the existence of generalizable AI rostering approaches questionable.

MH, typically hybrids of heuristic algorithms, make up an important category of approaches used in highly complex optimization problems that cannot be solved by conventional heuristics. Examples of MH include simulated annealing, genetic algorithm, multi-agent heuristic, and GRASP (the current algorithm in use). MH combine the great solving ability for all levels of optimization problems, model robustness across different contexts, and easy implementation while having a relatively fast computational speed [10]. However, besides multi-agent heuristics, all other MH use a centralized single agent to tackle problems, which tend to be time-consuming when the size of problem instances increases. In comparison, the multi-agent approach solves the problem by allowing multiple agents to search for their own local best solutions, and a central entity will act as a coordinator to help facilitate this negotiation process and achieve the overall best solution [13]. Though there is little academic work, let alone commercial software, focusing on multi-agent rostering, promising applications of multi-agent approaches in staff rostering can be inferred from other successful implementations in similar problem contexts such as course timetabling [14]. The distributed modelling approach with parallel computation can achieve fast computation and good quality (near-optimal results).

Information about the specific algorithms used in competing commercial scheduling software is not publicly accessible due to confidentiality. However, the client mentions several algorithms that the competitors may be using: 1) MP, 2) Heuristic approach, and 3) Artificial Intelligence/Machine Learning Approach [2]. MP and heuristic approaches are widely used in commercial scheduling software while the Artificial Intelligence/Machine Learning approach remains unclear in terms of actual usage in software [2].

4.2 Ideation Process

The focus of the project is to develop a rostering solution approach to produce better rosters while complying with selected fundamental rostering constraints. As discussed in Section 4.1, MP is widely used to solve rostering problems for optimal solutions, and advanced optimization solvers can now solve large scale problems within a reasonable amount of time. Unlike the classical nurse scheduling MP problem, the client's customers are from various industries that may have flexible shift start and end times [15]. This makes the rostering problem much more complicated and implies longer time to solve.

Since CP and AI are relatively less developed and context specific, the team decides to focus on the MH and MP approaches instead. In the selection process, we have two major factors to consider: solution quality and computational time.

From the perspective of solution quality, MP undoubtedly produces the best solution compared to other approaches that do not guarantee an optimal solution. Similar to MH, MP can also be applied in different problem contexts. However, it does suffer from the potential risk of no solutions while others can at least provide some feasible ones. If we were to only consider solution optimality, a natural choice will be the MP approach since it ensures an optimal solution.

On the other hand, for computational time, MH has a relatively faster computational time than MP across different scenarios and problem contexts. However, as mentioned earlier, advanced optimization solvers can now solve large-scale problems within a reasonable amount of time. Thus, the team decides to conduct computational experiments on problem instances of different sizes to compare the two algorithms thoroughly.

5 Methodologies

5.1 Baseline GRASP Model

The baseline model is the GRASP metaheuristics, which is currently employed by the client. The algorithm consists of three phases: Construction, Enforce Minimums, and Improvement as shown in Figure 1 [2]. In the Construction phase, the algorithm will score each combination (employee, day, length of time interval, task) and assign shifts to employees to cover the demand. Shifts will be added until there is no more good combination left. During the Enforce Minimums phase, extra shifts will be added or existing shifts will be extended while the shifts are still able to cover the demand. It is also an iterative process until the minimum hours rule is met. In the final Improvement phase, the shifts will be shrunk in order to reduce overstaffing or be extended in order to reduce understaffing. Changes in the tasks performed for each shift can also be made in this phase. The major objective for the final stage is to improve the overall coverage.

Since the GRASP heuristic is available in the client's environment, the team can customize rostering constraints and modify numerical value setups in the database to generate rosters using GRASP for different datasets. The constraints can be set up by changing the parameters on the platform (Dayforce). The overstaffing and understaffing values can be calculated using the differences between labour force supplies from the generated GRASP schedule and the input labour force demands. The overall penalties of rosters are then calculated based on the overstaffing and understaffing values.

The GRASP model sets a baseline for the rosters produced by the Integer Programming (IP) model. The same problem instances will be tested using the GRASP model and the proposed IP model and the performances (solution quality and computational time) of the two methodologies will be compared in the following sections.

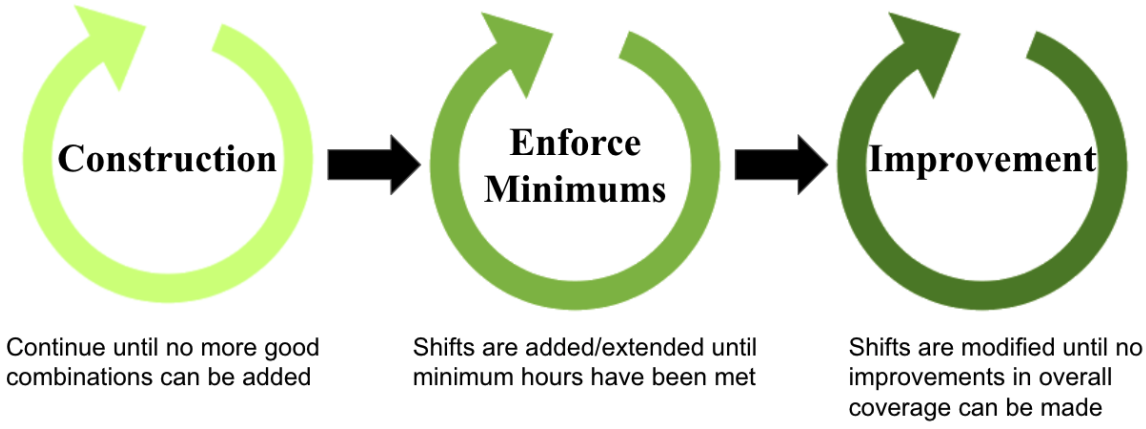

Figure 1: Illustration of GRASP Metaheuristics

5.2 Integer Programming (IP) Model

An Integer Programming (IP) problem is a type of MP that finds an optimal value of a linear function subject to a set of linear constraints over integer variables [16]. The proposed solution is an IP model that creates a seven-day rostering planning horizon. In the objective function, given the demand for a seven-day planning horizon, the goal is to find a feasible roster that minimizes the overall overstaffing (too many employees scheduled) and understaffing (too few employees scheduled) penalties for all employees over all time slots and days. For each time slot k on each day j for each task ℓ, two penalty terms are defined: one for overstaffing ($p^o_{jk\ell}$) and another for understaffing ($p^u_{jk\ell}$). Intuitively, a larger penalty would be applied to understaffing during busy hours than regular hours in retail stores or restaurants, and a larger penalty would be applied to overstaffing during regular hours than busy hours. The penalty terms are multiplied by the amount of overstaffing and understaffing defined by two decision variables $s^o_{jk\ell}$ and $s^u_{jk\ell}$, respectively. The overall penalties are calculated by taking the sum over the entire planning horizon as given by the equation $\sum_{j\in J}\sum_{k\in K}\sum_{\ell\in L}$ $(p^o_{jk\ell}s^o_{jk\ell}+ p^u_{jk\ell}s^u_{jk\ell})$. In addition to the objective function, the schedule produced must also satisfy a set of scheduling requirements. There are 10 main types of constraints, which are demand balance, minimum demand coverage, employee availability, maximum consecutive days, minimum time between shifts, maximum shifts per day, maximum shifts per week, minimum and maximum daily hours, minimum and maximum weekly hours, and maximum and maximum shift length. The detailed IP model formulation are included in the following subsections: Section 5.2.1 defines the IP model nomenclature, Section 5.2.2 presents the complete IP model formulation, and Section 5.2.3 provides a description of the objective value function and constraints in the IP model.

5.2.1 IP Model Nomenclature

Indices

ℓ Task index

i Employee index

j Day index

k Slot index
(24x4 = 96 15-minute slots in a day If an employee is assigned to a shift from 8 - 8:30, then for 8:00 - 8:15, the slot index would be 33; for 8:15 - 8:30, the slot index would be 34.) A shift consists of slots.)

Sets

I Employee indices, 1, ..., $|I|$

J Day indices, 1, ..., $|J|$

K Slot indices, 1, ..., $|K|$

L Task indices, 1, ..., $|L|$

Parameters

$|I|$ Number of employees in the scheduling unit

$|J|$ Number of days in the scheduling horizon (i.e. 7 days in one week)

$|K|$ Number of slots in a day (i.e., 1 to 96 15min slots in a day)

$|L|$ Number of task types, (e.g., cashier, cook)

$d_{jk\ell}$ Demand on day j, slot k, for task ℓ
When demand = 0, there is no demand for the given slot;
When demand is a positive integer, there is demand for the given slot.

$a_{ijk\ell}$ 1 if employee i is available on day j, slot k, for task ℓ; 0 otherwise

g^{min} Minimum number of hours an employee can be assigned in a day

g^{max} Maximum number of hours an employee can be assigned in a day

h^{min} Minimum number of hours an employee can be assigned in a week

h^{max} Maximum number of hours an employee can be assigned in a week

m^{day} Maximum number of shifts an employee can be assigned in a day

m^{week} Maximum number of shifts an employee can be assigned in a week

$p^u_{jk\ell}$ Penalty for understaffing on day j, slot k, for task ℓ

$p^o_{jk\ell}$ Penalty for overstaffing on day j, slot k, for task ℓ

$C_{jk\ell}$ Minimum coverage requirement on day j, slot k for task ℓ (i.e., a percentage, e.g. 70% coverage)

Decision Variables

$x_{ijk\ell}$ 1 if employee i is scheduled on day j, slot k, for task ℓ; 0 otherwise

$b_{ijk\ell}$ 1 if employee i starts on day j, slot k, for task ℓ; 0 otherwise
For example, if employee i starts to work on 8AM, then there should be one constraint enforcing $b_{i,j,33,\ell} = 1$ when $x_{i,j,33,\ell} = 1$ and $x_{i,j,32,\ell} = 0$

$e_{ijk\ell}$ 1 if employee i ends on day j, slot k (exclusive), for task ℓ; 0 otherwise
For example, if employee i ends work on 5PM, then there should be one constraint enforcing $e_{i,j,69,\ell} = 1$ when $x_{i,j,68,\ell} = 1$ and $x_{i,j,69,\ell} = 0$

$s^o_{jk\ell}$ Number of overstaffing on day j, slot k, task ℓ

$s^u_{jk\ell}$ Number of understaffing on day j, slot k, task ℓ

W_{ij} 1 if employee i is scheduled on day j; 0 otherwise

5.2.2 Objective and Constraints

$$\min \quad \sum_{j\in J}\sum_{k\in K}\sum_{\ell\in L}(p^o_{jk\ell}s^o_{jk\ell} + p^u_{jk\ell}s^u_{jk\ell}) \tag{1}$$

$$\sum_{i\in I} x_{ijk\ell} - s^o_{jk\ell} + s^u_{jk\ell} = d_{jk\ell} \qquad \forall j,k,\ell \tag{2}$$

$$\sum_{i\in I} x_{ijk\ell} \geq C_{jk\ell} \times d_{jk\ell} \qquad \forall j,k,\ell \tag{3}$$

$$x_{ijk\ell} \leq a_{ijk\ell} \qquad \forall i,j,k,\ell \tag{4}$$

$$\sum_{j'=j-5}^{j} W_{ij'} \leq 5 \qquad \forall i, j \geq 6 \tag{5}$$

$$\sum_{k\in K}\sum_{\ell\in L} x_{ijk\ell} \geq W_{ij} \qquad \forall i,j \tag{6}$$

$$\sum_{k\in K}\sum_{\ell\in L} x_{ijk\ell} \leq M(W_{ij}) \qquad \forall i,j \tag{7}$$

$$I_{(k\leq 64)}(\sum_{l\in L} e_{ijk\ell} + \sum_{k'=k}^{k+32}\sum_{\ell\in L} b_{ijk'\ell}) \leq 1 \qquad \forall i,j,k \tag{8}$$

$$I_{(k>64)}(\sum_{l\in L} e_{ijk\ell} + \sum_{k'=k}^{96}\sum_{\ell\in L} b_{ijk'\ell} + \sum_{k''=1}^{32-(96-k)}\sum_{\ell\in L} b_{i,j+1,k'',\ell}) \leq 1 \qquad \forall i,j,k \tag{9}$$

$$\sum_{k\in K}\sum_{\ell\in L} x_{ijk\ell} \geq 4g^{\min}W_{ij} \qquad \forall i,j \tag{10}$$

$$\sum_{k\in K}\sum_{\ell\in L} x_{ijk\ell} \leq 4g^{\max}W_{ij} \qquad \forall i,j \tag{11}$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{\ell\in L} x_{ijk\ell} \geq 4h^{\min} \qquad \forall i \tag{12}$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{\ell\in L} x_{ijk\ell} \leq 4h^{\max} \qquad \forall i \tag{13}$$

$$\sum_{k\in K}\sum_{\ell\in L} b_{ijk\ell} \leq m^{\text{day}} \qquad \forall i,j \tag{14}$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{\ell\in L} b_{ijk\ell} \leq m^{\text{week}} \qquad \forall i \tag{15}$$

$$I_{(k\leq 80)}(\sum_{l\in L} b_{ijk\ell} + \sum_{k'=k}^{k+16}\sum_{l\in L} e_{ijk'\ell}) \leq 1 \qquad \forall i,j,k \tag{16}$$

$$I_{(k>80)}(\sum_{l\in L} b_{ijkl} + \sum_{k'=k}^{96}\sum_{\ell\in L} e_{ijk'\ell} + \sum_{k''=1}^{16-(96-k)}\sum_{\ell\in L} e_{i,j+1,k'',\ell}) \leq 1 \qquad \forall i,j,k \tag{17}$$

$$I_{(k\leq 64)}(\sum_{k'=k}^{k+32} e_{ijk'\ell} \geq b_{ijk\ell}) \qquad \forall i,j,k,l \qquad (18)$$

$$I_{(k>64)}(\sum_{k'=k}^{96} e_{ijk'\ell} + \sum_{k''=1}^{32-(96-k)} e_{i,j+1,k''\ell} \geq b_{ijk\ell}) \forall i,j,k,l \qquad (19)$$

$$b_{ijk\ell} \leq x_{ijk\ell} \qquad \forall i,j,k,\ell \qquad (20)$$

$$b_{ijk\ell} \leq 1 - x_{i,j,k-1,\ell} \qquad \forall i,j,k,\ell \qquad (21)$$

$$x_{i,j,k-1,\ell} + b_{ijk\ell} \geq x_{ijk\ell} \qquad \forall i,j,k,\ell \qquad (22)$$

$$e_{ijk\ell} \leq x_{i,j,k-1,\ell} \qquad \forall i,j,k,\ell \qquad (23)$$

$$e_{ijk\ell} \leq 1 - x_{ijk\ell} \qquad \forall i,j,k,\ell \qquad (24)$$

$$x_{i,j,k-1,\ell} - x_{ijk\ell} \leq e_{ijk\ell} \qquad \forall i,j,k,\ell \qquad (25)$$

$$b_{i,j+1,1,\ell} \leq 1 - x_{i,j,96,\ell} \qquad \forall i,j \leq 6,\ell \qquad (26)$$

$$x_{i,j,96,\ell} + b_{i,j+1,1,\ell} \geq x_{i,j+1,1,\ell} \qquad \forall i,j \leq 6,\ell \qquad (27)$$

$$e_{i,j+1,1,\ell} \leq x_{i,j,96,\ell} \qquad \forall i,j \leq 6,\ell \qquad (28)$$

$$x_{i,j,96,\ell} - x_{i,j+1,1,\ell} \leq e_{i,j+1,1,\ell} \qquad \forall i,j \leq 6,\ell \qquad (29)$$

$$\sum_{\ell \in L} x_{ijk\ell} \leq 1 \qquad \forall i,j,k \qquad (30)$$

$$x_{ijk\ell}, b_{ijk\ell}, e_{ijk\ell}, W_{ij} \in \{0,1\} \qquad \forall i,j,k,\ell \qquad (31)$$

$$s^o_{jk\ell}, s^u_{jk\ell} \in \mathbb{Z}^* \qquad \forall j,k,\ell \qquad (32)$$

5.2.3 Objective Function and Constraints Descriptions

The following is a description for the objective function and a description for each scheduling constraint.

- (1) **Objective Function:** The total penalty from overstaffing and understaffing over the rostering planning horizon.
- (2) **Demand Balance:** For each time slot k on each day j for each task ℓ the total number of employees scheduled minus the amount of overstaffing plus the amount of understaffing should equal to the given demand.
- (3) **Minimum Coverage:** For each time slot k on each day j for each task l, the total number of employees scheduled must be greater than a given minimum level of coverage (e.g., satisfy at least 80% of the given demand).
- (4) **Availability:** An employee can only be scheduled for time slots s/he is available for.
- (5) **Consecutive Days Rule:** An employee cannot be scheduled for more than 5 consecutive days.
- (6) and (7) **Relationship between x_{ijkl} and W_{ij}:** An employee is scheduled to work on a day if the employee has time slots assigned in that day.

- (8) and (9) **Time Between Shifts Rule:** When an employee ends a shift (within the first 16 hours of a day), there must be at least 8 hours before the employee can start another shift.
- (10) **Daily Hours Rule (Minimum):** An employee cannot be assigned less than g^{min} hours in a day.
- (11) **Daily Hours Rule (Maximum):** An employee cannot be assigned more than g^{max} hours in a day.
- (12) **Weekly Hours Rule (Minimum):** An employee cannot be assigned less than h^{min} hours in a week.
- (13) **Weekly Hours Rule (Maximum):** An employee cannot be assigned more than h^{max} hours in a week.
- (14) **Shifts Per Day Rule:** An employee cannot be assigned more than m^{day} shifts in a day.
- (15) **Shifts Per Week Rule:** An employee cannot be assigned more than m^{week} shifts in a week.
- (16) and (17) **Shift Length Rule (Minimum):** Each shift must be at least 4 hours long.
- (18) and (19) **Shift Length Rule (Maximum):** Each shift cannot be more than 8 hours long.
- (20), (21), and (22) **Relationship between x and b variables:** ($x_{i,j,k-1,l} = 0$ and $x_{ijkl} = 1$) if and only if $b_{ijkl} = 1$
- (23), (24), and (25) **Relationship between x and e variables:** ($x_{i,j,k-1,l} = 1$ and $x_{ijkl} = 0$) if and only if $e_{ijkl} = 1$
- (26) and (27) **Relationship between x and b variables - Complementary:** It is for the cross-day corner case, ($x_{i,j,96,l} = 0$ and $x_{i,j+1,1,l} = 1$) if and only if $b_{i,j+1,1,l} = 1$. Similar to (20)-(22), but (20) already cover all cases.
- (28) and (29) **Relationship between x and e variables - Complementary:** It is for the cross-day corner case, ($x_{i,j,96,l} = 1$ and $x_{i,j+1,1,l} = 0$) if and only if $e_{i,j+1,1,l} = 1$. Similar to (23)-(25), but (23) already cover all cases.
- (30) **One Task At a Time:** An employee i can only be assigned to one type of task l on day j and slot k.
- (31) **Decision Variable Domains:** $x_{ijkl}, b_{ijkl}, e_{ijkl}, W_{ij}$ are binary variables.
- (32) **Decision Variable Domains:** s^o_{jkl}, s^u_{jkl} are non-negative integers.

6 Application to Local Diner and Cosmetics Store

6.1 Understaffing and Overstaffing Penalty Definitions

The proposed model is applied to three examples: 2 synthetic examples of a local diner and 1 real example from a cosmetics store. Depending on whether the local diner or the cosmetics store is busy, different understaffing and overstaffing penalties are first defined. These penalty values are defined based on the intuition that we penalize more for **understaffing** during busy hours than for regular hours (or non-busy hours) for all types of jobs, and we penalize more for **overstaffing** during regular hours than for busy hours for all types of jobs. This is always true for both the local diner and the cosmetics store examples.

In addition to different penalty values during regular and busy hours, the penalty values may also vary depending on the type of work performed.

For a local diner, it is assumed that there are 3 types of jobs (ℓ): cashier, server, and cook. During regular hours, cashiers and cooks have the same understaffing and overstaffing penalties as the demand for these two types of jobs are typically fixed (i.e. local diners will always need cooks and cashiers to maintain their operations). Servers, however, have a lower penalty score in understaffing scenarios and a higher penalty score in overstaffing situations. This is because fewer servers (understaffing) is acceptable as the demand is generally lower and having each server cover more tables would be manageable. On the other hand, too many servers (overstaffing) would mean too much idle time for every average server and thus incur extra cost for the diner owner.

During busy hours, a slightly higher penalty is given to understaffing of cashier because in a small local diner, there may be only one cashier and understaffing means that there will be no cashier at all to help customers check out. Although cooks are still essential for diner's regular hour and busy hour operations, one cook could manage the work of two cooks in the case of understaffing at a slower speed. For servers, larger penalty is given to understaffing situations than overstaffing situations. This is because having more servers is always good as a response to high demands and can help speed up the diner service process. However, having too few servers during peak demand periods could put too much workload on each server, resulting in slower service and poor customer experience. Table 2 summarizes the penalty values for the synthetic local diner examples.

Local Diner Penalty Values	**Understaffing**			**Overstaffing**		
	Cashier	**Server**	**Cook**	**Cashier**	**Server**	**Cook**
Regular Hours	2	1	2	2	3	2
Busy Hours	5	3	4	1	2	1

Table 2: Penalty Assignment For Local Diner Examples

For the real cosmetics store, no descriptions of the type of work performed in zone 1 and zone 3 are given. Therefore, the penalty values are the same for both zone 1 and zone 3. Table 3 summarizes the penalty values for the real cosmetics store example.

Cosmetic Store Penalty Values	**Understaffing**		**Overstaffing**	
	Zone 1	**Zone 3**	**Zone 1**	**Zone 3**
Regular Hours	1	1	2	2
Busy Hours	2	2	1	1

Table 3: Penalty Assignment For Cosmetics Store

6.2 Synthetic Local Diner Example 1

As an initial step, a local diner example is developed to evaluate the proposed IP model. The example is based on a factitious local diner with 3 types of jobs (ℓ) to run its daily operations: servers, cashiers, and cooks. There is a total of **16 employees** (i) working at the diner, each with a given set of day and time slot availability. **Each employee only possesses a specific skill to cover one type of job** (e.g., if an employee is a server, he or she cannot be a cashier or a cook). The local diner opens **everyday from 10AM to midnight** for customers. During **busy hours from 2PM to midnight**, the diner runs at **full-capacity (2 servers, 1 cashier, and 1 cook)**. Otherwise, the diner runs at **half-capacity (1 server, 1 cashier, and 1 cook)**.

The roster created must follow 8 rostering rules given by the store owner (rule 1) and the labour union (rules 2 to 8): 1) 100% demand coverage must be ensured to maximize profit, 2) employees cannot work more than 5 days consecutively, 3) employees need at least 8 hours between shifts to ensure good rest, 4) employees cannot work more than 1 shift per day, 5) employees cannot work more than 6 shifts per week, 6) for each day an employee works, the employee must work a minimum of 4 hours and no more than 10 hours, 7) employees cannot work less than 20 hours or more than 40 hours per week, and 8) the length of each shift must be between 4 hours and 8 hours.

Three sets of data are created with varying granularity of time slots. A time slot can span a 15-minute time interval, a 30-minute time interval, or a 60-minute time interval. A shift may be composed of one or more time slots and must satisfy the rostering rules provided. Figure 2 shows a roster generated using the 60-minutes interval dataset.

	Day1	Day2	Day3	Day4	Day5	Day6	Day7
Employee0	10 - 17 as cook	0.0	10 - 17 as cook	0.0	10 - 17 as cook	0.0	10 - 17 as cook
Employee1	17 - tomorrow 0 as cook	17 - tomorrow 0 as cook	0.0	0.0	17 - tomorrow 0 as cook	17 - tomorrow 0 as cook	0.0
Employee2	0.0	10 - 17 as cook	0.0	10 - 17 as cook	0.0	10 - 17 as cook	0.0
Employee3	0.0	0.0	17 - tomorrow 0 as cook	17 - tomorrow 0 as cook	0.0	0.0	17 - 24 as cook
Employee4	0.0	0.0	14 - 18 as cashier	0.0	10 - 17 as cashier	14 - 18 as cashier	10 - 18 as cashier
Employee5	17 - tomorrow 0 as cashier	14 - 19 as cashier	10 - 14 as cashier	16 - tomorrow 0 as cashier	17 - tomorrow 0 as cashier	0.0	0.0
Employee6	10 - 17 as cashier	10 - 14 as cashier	0.0	10 - 16 as cashier	0.0	10 - 14 as cashier	0.0
Employee7	0.0	19 - tomorrow 0 as cashier	18 - tomorrow 0 as cashier	0.0	0.0	18 - tomorrow 0 as cashier	18 - 24 as cashier
Employee8	10 - 14 as server	10 - 14 as server	10 - 14 as server	10 - 14 as server	0.0	0.0	10 - 14 as server
Employee9	14 - 19 as server	0.0	14 - 18 as server	14 - 19 as server	0.0	15 - 19 as server	14 - 19 as server
Employee10	19 - tomorrow 0 as server	0.0	0.0	0.0	19 - tomorrow 0 as server	19 - tomorrow 0 as server	19 - 24 as server
Employee11	14 - 19 as server	14 - 19 as server	0.0	0.0	10 - 16 as server	10 - 15 as server	0.0
Employee12	0.0	0.0	14 - 19 as server	0.0	14 - 19 as server	14 - 19 as server	14 - 19 as server
Employee13	0.0	14 - 19 as server	18 - tomorrow 0 as server	14 - 20 as server	16 - 20 as server	0.0	0.0
Employee14	19 - tomorrow 0 as server	19 - tomorrow 0 as server	0.0	19 - tomorrow 0 as server	0.0	19 - tomorrow 0 as server	0.0
Employee15	0.0	19 - tomorrow 0 as server	19 - tomorrow 0 as server	20 - tomorrow 0 as server	20 - tomorrow 0 as server	0.0	19 - 24 as server

Figure 2: Local Diner Example 1 60-Minutes Interval Roster

6.3 Synthetic Local Diner Example 2

This example adds more complexity to Local Diner Example 1. Basic setting remains the same except that there is a total of **30 employees** working at the diner, and **each employee can possess multiple skills to cover different job types** (e.g. an employee can work as a server today and as a cashier tomorrow). The opening time of the local diner now extends to **24/7** for customers. The diner's busy hours have changed accordingly. During **busy hours (lunch time from 11AM to 2PM and dinner time from 6PM to 9PM)**, the diner runs at **full-capacity (3 servers, 1 cashier, and 2 cooks)**. Otherwise, the diner runs at **half-capacity (2 servers, 1 cashier, and 1 cook)**. The roster created must follow the same eight rostering rules given by the store owner and the labour union, as defined in Section 6.2. Similarly, three sets of data are created to generate and evaluate rosters with different granularity of time slots (15-minute, 30-minute, or 60-minute time intervals). Figure 3 shows a roster generated using the 60-minute time interval dataset.

	Day1	Day2	Day3	Day4	Day5	Day6	Day7
Employee0	0.0	0.0	0.0	0 - 7 as cook	0 - 6 as cook	0 - 7 as cook	0 - 5 as cook
Employee1	0 - 5 as server	11 - 15 as cook	0 - 4 as server	0 - 6 as server	0.0	11 - 15 as cook	5 - 10 as cook
Employee2	0 - 4 as cook	0 - 7 as server	0.0	5 - 9 as server	0 - 4 as server	0 - 4 as server	0 - 7 as server
Employee3	5 - 12 as server	10 - 14 as cashier	13 - 20 as server	0.0	7 - 14 as cashier	8 - 14 as server	2 - 8 as cashier
Employee4	11 - 16 as cashier	14 - 20 as cashier	4 - 8 as server	4 - 11 as cashier	4 - 8 as server	0.0	0.0
Employee5	0.0	0.0	3 - 10 as cashier	11 - 18 as cashier	0.0	10 - 14 as cashier	8 - 14 as cashier
Employee6	11 - 15 as server	4 - 8 as server	4 - 10 as server	11 - 17 as server	0.0	11 - 18 as server	7 - 13 as server
Employee7	12 - 19 as server	12 - 19 as server	16 - 20 as server	0.0	5 - 10 as server	5 - 9 as server	15 - 22 as server
Employee8	16 - 22 as server	18 - 22 as server	18 - 22 as server	6 - 11 as server	11 - 17 as server	0.0	4 - 8 as server
Employee9	0.0	13 - 19 as server	8 - 13 as server	9 - 14 as server	0.0	20 - tomorrow 0 as server	0.0
Employee10	11 - 16 as server	11 - 15 as server	20 - tomorrow 0 as server	17 - tomorrow 0 as server	20 - tomorrow 0 as server	0.0	19 - 24 as server
Employee11	0.0	19 - tomorrow 0 as server	20 - tomorrow 0 as server	19 - tomorrow 0 as server	18 - tomorrow 0 as server	0.0	13 - 19 as server
Employee12	20 - tomorrow 0 as cook	18 - tomorrow 0 as cook	18 - 22 as cook	18 - 22 as cook	18 - tomorrow 0 as cook	0.0	0.0
Employee13	18 - 22 as cook	18 - 22 as cook	19 - tomorrow 0 as cook	0.0	0.0	7 - 11 as cook	18 - 22 as cook
Employee14	4 - 8 as cook	0.0	9 - 14 as cook	20 - tomorrow 0 as cook	0.0	18 - tomorrow 0 as cook	10 - 14 as cook
Employee15	20 - tomorrow 3 as cashier	20 - tomorrow 3 as cashier	11 - 15 as server	0.0	11 - 15 as cook	11 - 15 as cook	0.0
Employee16	11 - 15 as cook	0.0	11 - 15 as cook	11 - 15 as cook	0.0	1 - 5 as cashier	11 - 15 as cook
Employee17	19 - tomorrow 0 as server	0.0	10 - 14 as cashier	0 - 4 as cashier	12 - 17 as cook	14 - 20 as server	0.0
Employee18	16 - 20 as cashier	0.0	0 - 5 as cook	0.0	18 - tomorrow 1 as cashier	19 - tomorrow 2 as cashier	19 - 24 as cook
Employee19	14 - 20 as cook	0.0	14 - 19 as cook	0.0	0 - 7 as cashier	0.0	14 - 19 as cook
Employee20	0 - 4 as cashier	14 - 18 as cook	5 - 9 as cook	14 - 20 as cook	0.0	15 - 22 as cook	0.0
Employee21	4 - 11 as cashier	3 - 10 as cashier	18 - tomorrow 0 as cashier	18 - tomorrow 0 as cashier	0.0	5 - 10 as cashier	0.0
Employee22	18 - tomorrow 0 as server	0.0	0.0	14 - 19 as server	14 - 18 as cashier	14 - 19 as cashier	14 - 18 as cashier
Employee23	0.0	19 - tomorrow 0 as server	14 - 18 as cashier	0.0	14 - 20 as server	4 - 8 as server	18 - 24 as cashier
Employee24	0.0	8 - 13 as server	0.0	11 - 15 as server	10 - 15 as server	0 - 5 as server	8 - 15 as server
Employee25	0 - 5 as server	0 - 4 as server	0 - 4 as server	0 - 5 as server	8 - 14 as server	0.0	0 - 4 as server
Employee26	0.0	7 - 12 as server	10 - 16 as server	0.0	0 - 5 as server	9 - 15 as server	11 - 15 as server
Employee27	0.0	0.0	0.0	7 - 14 as cook	17 - 22 as cook	18 - tomorrow 0 as server	18 - 24 as server
Employee28	8 - 14 as cook	0 - 7 as cook	0.0	18 - 22 as server	17 - 22 as server	0.0	0.0
Employee29	5 - 11 as server	7 - 14 as cook	0.0	0.0	6 - 12 as cook	18 - 22 as server	0.0

Figure 3: Local Diner Example 2 60-Minutes Interval Roster

6.4 Real Cosmetics Store

After testing the proposed IP model on synthetic data, the client provided real data sets for a 7-day planning horizon from a cosmetics store for further evaluation. This example contains 2 types of jobs (ℓ): zone 1 and zone 3. In addition, there are a total of **66 employees** (i) working in the cosmetics store with their day and time availability. In the given data, **all employees are able to perform work in zone 3, but only some employees are also able to cover job in zone 1**.

The cosmetics store opens **everyday from 9AM to 9PM (12 hours in total)**, with demands specified for every 15-minute time interval. There is no explicit definition of regular and busy hours as before in the synthetic local diner examples. However, it is observed that the demands are the lowest during the first few hours after store opening and during the last few hours leading up to store closing. Therefore, these hours are defined as **regular hours (9AM-11AM and 7PM-9PM)**. During the other hours, the demands are higher, hence, these hours are defined as **busy hours (11AM-7PM)**.

In addition, it is also discovered that for zone 1, the number of employees available (supply) is larger than the demand for all hours everyday, while for zone 3, the number of employees available (supply) is smaller than the demand for most hours everyday. This means that it would not be possible to achieve 100% coverage for the cosmetics store example. For this reason, rostering rule 1 in Section 6.2 is relaxed to 70% minimum demand coverage in experiments (in Section 7). Furthermore, rostering rule 7 is relaxed by lowering the minimum number of working hours per week from 20 hours to 0 hour to accommodate the shortage of

employees discovered in zone 3. All other rostering rules remain the same as the local diner examples. Lastly, three sets of data are created to generate and evaluate rosters with different granularity of time slots (15-minute, 30-minute, or 60-minute time intervals). Figure 4 shows a roster generated using the 60-minute time interval dataset. Only the first 15 and last 15 employees are shown as an illustration.

	Day1	Day2	Day3	Day4	Day5	Day6	Day7
Employee0	13 - 18 as zone3	11 - 19 as zone3	9 - 14 as zone3	0.0	14 - 20 as zone3	15 - 19 as zone3	10 - 16 as zone3
Employee1	0.0	14 - 18 as zone3	11 - 19 as zone3	11 - 18 as zone1	14 - 19 as zone1	14 - 19 as zone1	0.0
Employee2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Employee3	0.0	9 - 13 as zone3	9 - 13 as zone3	9 - 14 as zone3	9 - 15 as zone3	12 - 18 as zone3	0.0
Employee4	10 - 18 as zone1	14 - 19 as zone1	9 - 13 as zone1	13 - 19 as zone1	0.0	11 - 19 as zone1	0.0
Employee5	0.0	0.0	0.0	16 - 20 as zone3	14 - 19 as zone3	0.0	10 - 18 as zone3
Employee6	0.0	0.0	0.0	0.0	11 - 19 as zone3	10 - 18 as zone3	10 - 18 as zone3
Employee7	14 - 19 as zone3	0.0	0.0	0.0	0.0	10 - 18 as zone3	11 - 17 as zone3
Employee8	15 - 19 as zone1	15 - 19 as zone1	9 - 13 as zone3	14 - 19 as zone3	9 - 13 as zone1	0.0	10 - 15 as zone1
Employee9	0.0	0.0	11 - 19 as zone3	0.0	0.0	0.0	0.0
Employee10	9 - 15 as zone1	9 - 14 as zone3	9 - 13 as zone3	9 - 16 as zone3	9 - 14 as zone1	0.0	14 - 18 as zone1
Employee11	10 - 18 as zone3	9 - 14 as zone3	10 - 18 as zone3	0.0	12 - 18 as zone3	14 - 20 as zone3	12 - 16 as zone3
Employee12	0.0	0.0	0.0	0.0	0.0	10 - 18 as zone3	11 - 19 as zone3
Employee13	9 - 15 as zone3	9 - 16 as zone3	9 - 15 as zone3	9 - 17 as zone3	9 - 14 as zone3	0.0	0.0
Employee14	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...
Employee51	11 - 19 as zone3	15 - 20 as zone3	0.0	11 - 18 as zone3	10 - 18 as zone3	0.0	9 - 16 as zone3
Employee52	0.0	9 - 16 as zone3	13 - 19 as zone3	9 - 13 as zone3	14 - 18 as zone3	11 - 19 as zone3	0.0
Employee53	0.0	9 - 14 as zone3	0.0	13 - 19 as zone3	9 - 14 as zone3	10 - 18 as zone3	0.0
Employee54	13 - 18 as zone3	14 - 19 as zone3	10 - 18 as zone3	9 - 13 as zone3	14 - 19 as zone3	0.0	11 - 17 as zone3
Employee55	15 - 19 as zone3	11 - 19 as zone3	13 - 19 as zone3	11 - 19 as zone3	9 - 14 as zone3	0.0	10 - 18 as zone3
Employee56	9 - 14 as zone3	0.0	11 - 19 as zone3	9 - 14 as zone3	9 - 14 as zone3	9 - 17 as zone3	12 - 16 as zone3
Employee57	0.0	0.0	0.0	0.0	0.0	10 - 18 as zone3	11 - 18 as zone3
Employee58	9 - 16 as zone3	14 - 18 as zone3	10 - 18 as zone3	0.0	13 - 18 as zone3	9 - 17 as zone3	0.0
Employee59	10 - 18 as zone3	13 - 19 as zone3	0.0	14 - 19 as zone3	13 - 19 as zone3	13 - 20 as zone3	13 - 19 as zone3
Employee60	16 - 20 as zone3	16 - 20 as zone3	16 - 20 as zone3	0.0	16 - 20 as zone3	11 - 19 as zone3	13 - 19 as zone3
Employee61	0.0	0.0	12 - 19 as zone3	9 - 15 as zone3	14 - 19 as zone3	11 - 19 as zone3	12 - 17 as zone3
Employee62	13 - 19 as zone3	12 - 19 as zone3	0.0	12 - 19 as zone3	9 - 16 as zone3	0.0	0.0
Employee63	12 - 19 as zone3	0.0	15 - 19 as zone3	13 - 19 as zone3	13 - 18 as zone3	8 - 15 as zone3	13 - 17 as zone3
Employee64	0.0	0.0	0.0	0.0	0.0	11 - 19 as zone3	9 - 17 as zone3
Employee65	15 - 19 as zone3	9 - 14 as zone3	0.0	0.0	11 - 19 as zone3	0.0	11 - 17 as zone3

Figure 4: Cosmetics Store Example 60-Minutes Interval Roster

7 Computational Experiments

7.1 Performance Measures

The two key metrics for algorithm performance evaluations are roster quality and computational time. Roster quality is measured by the penalty score of the produced roster, which is calculated from overstaffing penalty and understaffing penalty. Computational time is defined as the running time required for the algorithm to produce a roster.

7.1.1 Roster Quality

As shown in Table 4, the IP model (an MP approach) gives much lower penalty scores while GRASP (an MH approach) gives much higher penalty scores. This observation matches the characteristics discussed above in the Literature Review section: MH may produce feasible, but suboptimal solutions while MP always gives the best possible roster in terms of roster quality. This is because GRASP only tries to do a neighbourhood search and converges at a local minimal penalty solution point. On the other hand, MP examines the entire solution space and converges to the global optimal solution point(s).

GRASP	15 Minutes Time Intervals	30 Minutes Time Intervals	60 Minutes Time Intervals
Local Diner Example 1 (16 Employees)	Penalty = 202.5	Penalty = 212.5	Penalty = 281
Local Diner Example 2 (30 Employees)	Penalty = 230.25	Penalty = 154	Penalty = 227
Cosmetic Store (66 Employees)	Penalty = 611.5	Penalty = 605	Penalty = 605

IP Model	15 Minutes Time Intervals	30 Minutes Time Intervals	60 Minutes Time Intervals
Local Diner Example 1 (16 Employees)	Penalty = 0	Penalty = 0	Penalty = 0
Local Diner Example 2 (30 Employees)	Penalty = 0	Penalty = 0	Penalty = 0
Cosmetic Store (66 Employees)	Penalty = 19	Penalty = 11	Penalty = 6

Table 4: Solution Quality Comparison Between GRASP and IP Model

7.1.2 Computational Time

Similarly, the computational time results are shown below in Table 5. In smaller problem instances, similar computational times are observed from both the IP model and GRASP. For larger problem instances, the IP model uses much more time than GRASP. This is reasonable since IP model examines a much larger search space to find the global optimal solution point(s) while GRASP only searches around its initial starting neighbourhood. In summary, even with the most advanced optimization solver, the IP approach still is not able to solve large rostering problems in a comparable period of time when compared to GRASP.

GRASP	15 Minutes Time Intervals	30 Minutes Time Intervals	60 Minutes Time Intervals
Local Diner Example 1 (16 Employees)	5.343 s	2.75 s	1.31 s
Local Diner Example 2 (30 Employees)	62.97 s	18.09 s	9.23 s
Cosmetic Store (66 Employees)	85.40s	18.28s	5.41s

IP Model	15 Minutes Time Intervals	30 Minutes Time Intervals	60 Minutes Time Intervals
Local Diner Example 1 (16 Employees)	4.11 s	1.26 s	0.45 s
Local Diner Example 2 (30 Employees)	6624 s	395 s	72 s
Cosmetic Store (66 Employees)	21538.72 s	695.17 s	216 s

Table 5: Computational Time Comparison Between GRASP and IP Model

7.2 Problem Size Comparison

Given the previous analysis, it is obvious that neither the IP model nor GRASP can outperform the other in terms of both the roster quality and the computational time. The MP approach guarantees the best roster quality from its optimal solution while MH benefits from its fast neighbourhood search algorithm for local convergence. Therefore, it raises the interest of examining the IP model's computational time on different problem sizes and exploring the possibility of substituting GRASP with the IP model in certain contexts (e.g., small problem sizes).

As discussed previously, Local Diner Example 1 has 16 employees, three different types of tasks, and opens for 14 hours; Local Diner Example 2 has 30 employees, three different types of tasks, and opens 24/7; Cosmetics Store Example has 66 employees, two types of tasks, and opens for 12 hours. Table 6 shows the associated computational time and number of variables for various problem sizes (different time slot lengths). As the time slot length decreases, the problem size gets larger as there will be more time slots to consider when assigning shifts to employees. For example, in an hour, there will be two 30-minute time interval time slots and four 15-minute time interval time slots.

	15 Minutes Time Intervals	30 Minutes Time Intervals	60 Minutes Time Intervals
Local Diner Example 1 (16 Employees)	4.11 s Variable types: 0 continuous, 100912 integer (96880 binary)	1.26 s Variable types: 0 continuous, 50512 integer (48496 binary)	0.45 s Variable types: 0 continuous, 25312 integer (24304 binary)
Local Diner Example 2 (30 Employees)	6624 s (~110 minutes) Variable types: 0 continuous, 185682 integer (181650 binary)	395 s (~6.5 minutes) Variable types: 0 continuous, 92946 integer (90930 binary)	72 s (~1.2 minutes) Variable types: 0 continuous, 46578 integer (45570 binary)
Cosmetic Store (66 Employees)	21538.72 s (~131 minutes) Variable types: 0 continuous, 269262 integer (266574 binary)	695.17 s (~18 minutes) Variable types: 0 continuous, 134862 integer (133518 binary)	216 s (~13 minutes) Variable types: 0 continuous, 67662 integer (66990 binary)

Table 6: Problem Size Comparisons

A positive correlation between problem sizes and computational time needed to solve the rostering problems is observed. Specifically, the number of employees seems to have a relatively linear relationship with the computational time, while the length of a rostering time slot seems to have a negative exponential relationship. This is because an increase in the number of employees simply adds more labour to be considered, but decreasing the time slot length increases the number of decision variables quadratically, resulting in an exponential increase.

From both Table 4 and Table 5, a trade-off between roster quality and computational time is observed. Faster computational speed means that searching is done in a smaller search space, resulting in lower roster quality. However, since MP's searching process is an iterative process, it is possible to combine both approaches to hedge parts of their deficits and capture some advantages from both approaches. For instance, when the computational times of the two approaches are similar (i.e., when the problem size is small), an automatic algorithm selection can be added to the MH (GRASP) to force the MH to use an optimization approach (IP model).

8 Implementation Plan for Client

The code for the optimization methodology will be shared via encrypted University of Toronto OneDrive.

In the code for the IP model (**Gurobi_IP_Model.ipynb**), the parameters are set up to store the information including the number of employees, the length of the rostering period, the length of each interval to be assigned, the number of tasks, the demand for each task, the employee time availability, the employee skill sets, the penalties for overstaffing and understaffing, and the upper and lower limits for the constraints. The parameter names and their data types are listed in Table 7 below.

Since not all the customers of our client have the same requirements, the boolean variables add flexibility and adjustability to turn on and off the constraints of the IP model. After setting up the parameters, the client can easily choose which constraints are applicable to the problem by changing the corresponding boolean variables. The mapping of each constraint to the control variable is shown in Table 8.

Since the decision variable is whether each employee is assigned on each day, in each time slot for each task, the roster can be easily produced and visualized using the decision variables after the optimization problem is solved. The final roster (**.xlsx file**) showing the shifts and tasks assigned to each employee across the scheduling period is available in the **updated_schedule_save.xlsx** file.

Besides the final rostering schedule, additional output files will also be generated: 1) b_{ijkl} files for each day in the rostering period, 2) e_{ijkl} files for each day in the rostering period, 3) x_{ijkl} files for each day in the rostering period, 4) overstaffing s_{jkl} file, 5) understaffing s_{jkl} file, and 6) W_{ij} files. The detailed explanations of the output files are listed in Table 9.

However, the time to solve the optimization problem largely depends on the problem size and the difficulty of the problem instances. Since the time taken to generate a rostering schedule is also an essential factor to be considered in real-life, the client can always relax some constraints and solve the simplified optimization problem, and then use the corresponding solution as a warm start to the current GRASP algorithm. Another possible way to provide a warm start to GRASP is to change to a larger time slot length (for example, if the problem uses time slots of 15-min time intervals originally, it can be changed to 30-min time intervals). This can decrease the number of decision variables to be solved for the optimization problem, hence, leading to a shortened computing time. Similarly, the roster obtained can act as the initial guess for GRASP in order to achieve a better solution quality (i.e. lower overstaffing and understaffing penalties).

Information	**Name of the Parameter**	**Type of the Parameter**	**Illustrative Example**
Number of employees	num_employees	integer	10 means 10 employees
Length of the rostering period	J	integer	7 means a 7-day rostering period
Length of each interval (represented by how many intervals in one hour)	interval_factor	integer	2 means 30-min interval
Number of tasks	L	integer	3 means 3 tasks
Demand for each task	demand	array	[[0,0,1], [2,0,1]] means 1 employee is needed for task 3 in the first two hours, and 2 employee is needed for task 1 in the second hour
Employee time availability	time_aval	array	[[0,0,0,1,1,1], [1,1,0,0,0,0]] means employee 1 is available for the 4th to 6th time intervals, and employee 2 is available for the first two time intervals (number of columns should match 24*interval factor)
Employee skill sets	employee_skills	array	[[0,0,1],[1,0,0]] means that employee 1 is capable of task 3, and employee 2 is capable for task 1
Penalty for overstaffing	overstaffing	array	[[1,2,1],[1,1,1]] means at time interval 1, penalties for overstaffing (assign 1 employee more than demand) at task 1, 2 and 3 are 1,2, and 1 respectively, and at time interval 2 the penalties are 1 for all tasks
Penalty for understaffing	understaffing	array	[[1,2,1],[1,1,1]] means at time interval 1, penalties for understaffing (assign 1 employee less than demand) at task 1, 2 and 3 are 1,2, and 1 respectively, and at time interval 2 the penalties are 1 for all tasks
Minimum coverage requirement for demand	coverage_level	float	1 means 100% demand coverage is required (i.e. no understaffing)
Maximum consecutive days	consec_day	integer	4 means an employee can't work for more than 4 consecutive days
Minimum number of hours can be assigned to an employee in a day	g_min	integer	4 means if an employee is assigned for a day, the minimum number of hours assigned is 4 hours
Maximum number of hours can be assigned to an employee in a day	g_max	integer	8 means if an employee is assigned for a day, the maximum number of hours assigned is 8 hours
Minimum number of hours can be assigned to an employee in a week	h_min	integer	20 means an employee must work at least 20 hours per week
Maximum number of hours can be assigned to an employee in a week	h_max	integer	40 means an employee can't work for more than 40 hours per week
Maximum number of shifts can be assigned to an employee in a day	m_day	integer	1 means that an employee can't be assigned more than 1 shift per day
Maximum number of shifts can be assigned to an employee in a week	m_week	integer	6 means that an employee can't be assigned more than 1 shift per week

Table 7: Summary of Parameter Descriptions

Constraint	**Name of the Boolean Variable**
Minimum Coverage	min_coverage_rule
Consecutive Days Rule	consec_day_rule
Time Between Shifts Rule	time_btw_shift_rule
Daily Hours Rule (Min/Max)	daily_hours_rule
Weekly Hours Rule (Min/Max)	weekly_hours_rule
Shifts Per Day Rule	shift_per_day_rule
Shifts Per Week Rule	shift_per_week_rule

Table 8: Summary of Constraint Control Variables

Output File	**Explanation**
Day_{}_b_ijkl.xlsx	Each employee tab gives the starting time slot for each shift for each task for the employee on day {}, if it is 1, it means it is a starting time slot for a shift for the employee, otherwise it is not
Day_{}_e_ijkl.xlsx	Each employee tab gives the ending time slot (exclusive) for each shift for each task for the employee on day {}, if it is 1, it means it is an ending time slot for a shift for the employee, otherwise it is not
Day_{}_x_ijkl.xlsx	Each employee tab gives the time slot assigned for each task for the employee on day {}, if it is 1, it means the time slot is assigned for the employee, otherwise it is not
overstaffing_sjkl.xlsx	Each day tab gives the number of overstaffing for each time slot for each task on the day
understaffing_sjkl.xlsx	Each day tab gives the number of understaffing for each time slot for each task on the day
W_ij.xlsx	The rows (i) represent employees, and columns (j) represent day, if it is 1, it means that employee i is assigned a shift on day j, otherwise no shift is assigned

Table 9: Description of Output Files

9 Conclusion

In conclusion, an integer programming (IP) model is formulated to create seven-day rostering schedules. The model aims to minimize the demand coverage penalty (from overstaffing and understaffing) while enforcing a set of selected fundamental scheduling constraints, such as minimum time between shifts and maximum daily hours.

Two synthetic local diner datasets and one real store dataset are used to generate rostering schedules and evaluate both IP and baseline GRASP models. Compared to the baseline model, the IP model outperforms GRASP in terms of roster quality, guaranteeing optimal rosters created. However, although similar performance is achieved on smaller problem instances, GRASP outperforms the IP model in terms of computational time. This makes sense as GRASP is a metaheuristic algorithm which searches for a local optimal value in a small neighbourhood, whereas the IP model has a much larger search space and finds a global optimal value.

Therefore, the IP model does not outperform GRASP completely. The capstone team recommends leveraging the IP model for smaller problem instances and GRASP for larger and more complex problem instances. Hence, decision makers have to make a trade-off between solution quality and computational speed depending on problem sizes.

This work is a variation of the nurse scheduling problem applied to the rostering domain, which has stricter requirements than the scheduling problem. As previous research and studies mostly focused on scheduling problems, it could be considered as the first generalized optimization approach to tackle rostering problems. Through this project, the optimal rosters produced would minimize understaffing and overstaffing while satisfying rostering constraints, thereby saving money for users. It could also decrease the effort required later on to adjust the produced rosters manually. This would also be a good starting framework that can be extended in the future to offer flexibility by accommodating varying user needs or inputs.

10 Future Work

While the proposed mathematical optimization model is able to guarantee the optimal solution quality, it could lead to extremely long computing time for larger or more difficult problem instances. Therefore, the team suggests to investigate possible hybrid models in the future, that is to say, to solve the optimization problem on relaxed settings and design a heuristic algorithm to improve the obtained rostering schedules.

Furthermore, with employee satisfaction becoming more and more important in the workspace, it is essential for organizations to take employee preferences into account when generating rostering schedules. For instance, although an employee can be available on both Monday and Tuesday, he or she may prefer a shift on Tuesday as opposed to Monday. If such a factor can be actively incorporated and coded in the rostering algorithm, employee satisfaction can be effectively maximized and decision makers' trust in the algorithm can be increased.

Finally, the latest research on inverse optimization could be leveraged to solve the rostering problem from the end-user's perspectives. As the client has previously mentioned, it is quite common that some store managers will still manually adjust the rostering schedule after it is generated by the GRASP algorithm based on the demand provided. It indicates that the objectives or constraints of the end users (for example, the store managers) are not effectively represented by the current problem settings. Inverse optimization is a process that reverses the traditional optimization by taking the final decisions (adjusted rostering schedules) as inputs, and determines the corresponding objectives and constraints that render optimal decisions [17].

References

[1] *Dayforce*. URL: https://www.ceridian.com/ca/products/dayforce (visited on 03/28/2022).

[2] *Weekly Client Meeting with Ceridian*.

[3] *How a rotating roster could benefit your workplace*. URL: https://ento.com/rotating-roster-benefits/ (visited on 03/29/2022).

[4] A.T Ernst et al. "Staff scheduling and rostering: A review of applications, methods and models". In: *European Journal of Operational Research* 153.1 (2004). Timetabling and Rostering, pp. 3–27. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(03)00095-X. URL: https://www.sciencedirect.com/science/article/pii/S037722170300095X.

[5] B.M.W. Cheng, J.H.M. Lee, and J.C.K. Wu. "A nurse rostering system using constraint programming and redundant modeling". In: *IEEE Transactions on Information Technology in Biomedicine* 1.1 (1997), pp. 44–54. DOI: 10.1109/4233.594027.

[6] C. Anantaram et al. "Crew rostering system-An expert system for scheduling crew for Indian Airlines". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. 1993, pp. 63–70. DOI: 10.1109/CAIA.1993.366660.

[7] E.M. Morgado and J.P. Martins. "An AI-based approach to crew scheduling". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. 1993, pp. 71–77. DOI: 10.1109/CAIA.1993.366659.

[8] Ei Shwe Sin. "Reinforcement learning with EGD based hyper heuristic system for exam timetabling problem". In: *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*. IEEE. 2011, pp. 462–466.

[9] Joe Henry Obit et al. "Designing a multi-agent approach system for distributed course timetabling". In: *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*. 2011, pp. 103–108. DOI: 10.1109/HIS.2011.6122088.

[10] Andreas T Ernst et al. "Staff scheduling and rostering: A review of applications, methods and models". In: *European journal of operational research* 153.1 (2004), pp. 3–27.

[11] C Anantaram et al. "Crew rostering system-an expert System for scheduling crew for indian airlines". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. IEEE. 1993, pp. 63–70.

[12] Ernesto M Morgado and Joao P Martins. "An AI-based approach to crew scheduling". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. IEEE. 1993, pp. 71–77.

[13] Alessandro Agnetis et al. "Multiagent scheduling". In: *Berlin Heidelberg: Springer Berlin Heidelberg. doi* 10.1007 (2014), pp. 978–3.

[14] Joe Henry Obit et al. "Designing a multi-agent approach system for distributed course timetabling". In: *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*. IEEE. 2011, pp. 103–108.

[15] Manion Anderson et al. “Optimization Helps Scheduling Nursing Staff at the Long-Term Care Homes of the City of Toronto”. In: *arXiv preprint arXiv:2102.09461* (2021).

[16] *What is integer programming?* URL: https://www.ibm.com/docs/en/icos/12.8.0.0?topic=problem-what-is-integer-programming (visited on 01/09/2022).

[17] Timothy CY Chan, Rafid Mahmood, and Ian Yihang Zhu. “Inverse optimization: Theory and applications”. In: *arXiv preprint arXiv:2109.03920* (2021).

Attribution Table

Project Title:	CD1	Supervisor:	Merve Bodur
Document Name:	Final Design Specifications (FDS)	Date:	April 1, 2022

This form must be filled out and signed for each team-written document. The completed form must be attached to the document, though not included as part of the document (not included in the table of contents, page numbers, or word count limits). It should accurately reflect each team member's contribution to the document and in what way they contributed.

If there are irreconcilable differences that are preventing all team members from signing the attribution table then each team member must write a letter (max 800 words) explaining their position on the difference and suggest a solution.

Making fraudulent claims in an attribution table displays intent to deceive and is a serious academic offence.

Section	Student Names			
	Jiahua Chen	Jiaru Li	Sijia Li	Mingkun Wang
1 Executive Summary	RD			
2 Relevant Terminologies		RD		
3 Introduction	RD	MR	MR	
4 Solution Selection	RS	RS	RS	RD, RS

5 Methodologies	MR	RD	RD	MR
6 Application to Local Diner and Cosmetics Store	RD	MR	RD	MR
7 Computational Experiments	MR	MR	MR	RD
8 Implementation Plan for Client		RD	MR	
9 Conclusion	MR	RD		
10 Future Work		RD		MR
All	FP, CM, ET	FP, CM, ET	FP, CM, ET	FP, CM, ET

Fill in abbreviations for roles for each of the required content elements. You do not have to fill in every cell. The “all” row refers to the complete report and should indicate who was responsible for the final compilation and final read through of the completed document.

RS – research

RD – wrote first draft

MR – major revision

ET – edited for grammar and spelling

FP – final read through of complete document for flow and consistency

CM – responsible for compiling the elements into the complete document

OR – other

If you put OR (other) in a cell please put it in as OR1, OR2, etc. Explain briefly below the role referred to:

OR1: _____

OR2: _____

By signing below, you verify that you have:

- Read the attribution table and agree that it accurately reflects your contribution to the associated document.
- Written the sections of the document attributed to you and that they are entirely original.
- Accurately cited and referenced any ideas or expressions of ideas taken from other sources according to an accepted standard.
- Read the University of Toronto Code of Behaviour on Academic Matters and understand the definition of academic offense includes (but is not limited to) all forms of plagiarism. Additionally, you understand that if you provide another student with any part of your own or your team's work, for whatever reason, and the student having received the work uses it for the purposes of committing an academic offence, then you are considered an equal party in the offence and will be subject to academic sanctions.

Print Name:	Jiahua Chen	Signature:	Jiahua Chen
Print Name:	Jiaru Li	Signature:	Jiaru Li
Print Name:	Sijia Li	Signature:	Sijia Li
Print Name:	Mingkun Wang	Signature:	Mingkun Wang