Using Dominance in Solving Complex, Combinatorial Optimization Problems: Applications from Healthcare Provider Scheduling and Vehicle Routing

by

Young-Chae Hong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2017

Young-Chae Hong

hongyc@umich.edu

ORCID iD: 0000-0001-6330-6101

I dedicate this dissertation to the God of miracles
for all love, support and encouragement throughout my life.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Amy Cohn for her invaluable support and patience. This dissertation could not be finished without her encouragement. I would also like to thank Prof. Marina Epelman for her amazing advice for critical proof. I want to thank Prof. Pascal Van Hentenryck and Prof. Amitabh Sinha for serving on my committee and for their support and review of this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

ABM	arc-based model
ACGME	Accreditation Council for Graduate Medical Education
AHP	analytic hierarchy process
AY	academic year
BOPI	bi-objective problem with integer values
BSPs	bad sleep patterns
COA	cuckoo optimization algorithm
CP	constraint programming
CVRP	capacitated VRP
DP	dynamic programming
DVRP	distance-constrained VRP
EA	evolutionary algorithm
ED	emergency department
ER	emergency room
ESPPRC	elementary shortest path problem with resource constraints
GA	genetic algorithm
GME	graduate medical education
GP	goal programming
HVRP	heterogeneous fleet VRP
IP	integer programming
LP	linear programming

MDVRP	multi-depot VRP
MIP	mixed-integer programming
MOIP	multi-objective integer programming
MOP	multi-objective optimization problem
MOPI	multi-objective problem with integer values
MP	mathematical programming
NRMP	National Resident Matching Program
NSE	night shift equity
NSP	nurse scheduling problem
PBM	path-based model
PCCs	post-continuity clinic shifts
PEDS	University of Michigan Pediatric Emergency Department Scheduling
PSP	physician scheduling problem
RSP	resident schduling problem
RSPPRC	cycle relaxation of ESPPRC
TCHVRP	time-constrained heterogeneous vehicle routing problem
TOIP	tri-objecive integer programming
TSE	total shift equity
TSP	traveling salesman problem
UFSs	uncovered flex shifts
VRP	vehicle routing problem
VRPTW	VRP with time windows
WESPPRC	weak dominance relaxation of ESPPRC

ABSTRACT

The focus of this dissertation is to develop mathematical methods for the multi-criteria optimization problem and the vehicle routing problem. We approach these problems through the concept of Pareto dominance. Our goal is to develop general algorithms that utilize Pareto dominance and solve the problems in reasonable time.

We first consider the problem of assigning medical residents to shifts within a pediatric emergency department. This problem is challenging to solve for a number of reasons. First, like many other healthcare personnel scheduling problems, it has a non-homogeneous work force, with each resident having different characteristics, requirements, and capabilities. Second, residency scheduling problems must not only ensure adequate resources for patient care but must also meet educational training needs, adding further complexity and constraints. Finally, since many factors should be taken into account when selecting the "best" schedule, there is no one clear, well-defined single objective function under which to optimize. Thus, it is difficult, if not impossible, to preassign weights that allow these factors and the preferences of the scheduler (typically, a Chief Resident) to be captured in a mathematical objective function.

We propose an integer programming formulation and an iterative, interactive approach in which we use this integer program for ill-defined multiple objective criteria which are often in conflict with each other. After we identify quantifiable metrics through the interactive approach, we develop an integer programming-based approach embedded within a recursive algorithm to provide the Chief with a set of Pareto-dominant schedules from which to select. We then present our collaborative work with the University of Michigan C.S. Mott Children's Hospital in building monthly schedules, focusing on both the tractability of our methods and a case study to study how a Chief

Resident would evaluate the Pareto set.

When building schedules, an alternative is to use a column generation approach in which each variable represents complete sequences of tasks for a single agent to perform. In Chapter 4, we show how the concept of Pareto dominance can be used to generate columns efficiently. We evaluate dynamic programming-based column generation approaches for the vehicle routing problem since the models and algorithms proposed for the vehicle routing problems can be used effectively not only for the solution of transportation problems concerning the delivery or collection of goods but also for the solution of scheduling problems arising in healthcare as well.

In particular, we consider a new variant of the Time-Constrained Heterogeneous Vehicle Routing Problem (TCHVRP). In this problem, the cost and travel time of any given arc vary by vehicle type within a heterogeneous fleet. We make no assumptions about Pareto dominance across vehicles; nor do we assume that cost and time are correlated. Our research is motivated by situations in which existing fleets are evolving to incorporate hybrid vehicles in addition to their existing vehicle types; for many vehicles, the cost per mile depends heavily on the type of driving (such as highway versus city). We formulate TCHVRP as a path-based model, which we solve using column generation. We introduce several different methods to solve the pricing problem. We conclude by conducting empirical analyses to assess the performance of the proposed approach.

CHAPTER 1

Introduction

In this dissertation, we study mathematical models and algorithms for solving combinatorial optimization problems: a multi-criteria healthcare scheduling problem and a vehicle routing problem. Specifically, we develop Pareto-based approaches to solving them. First, for the multi-criteria healthcare scheduling problem, we apply an interactive method to see the impact of the decision maker's feedback on the quality of the schedule and then consider a Pareto-based approach that conveys much more information to the decision maker than the interactive method. Next, we introduce a dynamic programming-based algorithm that utilizes dominance in order to solve the pricing problem of the vehicle routing problem via column generation.

The remainder of this dissertation is organized as follows: In Chapter 1, we review the literature on healthcare personnel scheduling, multi-objective optimization problems, and the vehicle routing problem. Then, we explain an iterative, interactive approach we've used for the University of Michigan Pediatric Emergency Department Scheduling (PEDS) problem and present computational results in Chapter 2. In Chapter 3, we suggest a Pareto-dominant approach and demonstrate the tractability and usability of this approach. In Chapter 4, we show dynamic programming-based approaches to solving the shortest path problem with resource constraints in the vehicle routing problem (VRP) and introduce the concept of Pareto dominance to improve computational time. In Chapter 5, we present some conclusions and future research suggestions.

1.1 Healthcare Personnel Scheduling

Through the last decades, many studies have been attempted to solve healthcare personnel scheduling problems. Most of these studies have been done to schedule nurses in a hospital department [1, 2]. In addition, there exists literature focusing on physicians [3, 4] and on residents [5, 6].

1.1.1 Literature Review

Scheduling, the "allocation of resources to tasks over given time periods", is a decision-making process that occurs throughout almost all manufacturing and services industries [7]. Examples include manufacturing processes [8]; transportation systems such as airlines [9], railways [10], and public transportation [11]; staff scheduling in call centers [12]; emergency services such as police [13], ambulance [14], and fire departments [15]; and even toll booths [16].

Scheduling problems are solved by using a variety of solution approaches, from heuristic algorithms to exact methods, based on problem-specific characteristics (e.g. cyclic vs. non-cyclic, deterministic vs. stochastic, single machine vs. parallel machine, and so on). The following review papers address many of these methods and characteristics [7, 17, 18].

For *personnel scheduling* or *human resource allocation* problems, Edie's paper [16] on traffic delays at toll booths was the first formal approach, and the first integer programming (IP) formulation was introduced by Dantzig [19]. Since then, these problems have been studied widely since labor cost is a major direct cost in many environments [18, 20, 21]. These problems provide many unique challenges, given the need to satisfy personal preferences and variability across workers' skill sets.

Within healthcare, the personnel scheduling literature can be primarily divided into two categories. The first, and most abundant, is in *nurse scheduling* [1, 2]. In *physician scheduling*, most of the literature focuses either on shift scheduling, primarily for emergency department physicians [3, 22, 23, 24], or on scheduling operating room time for surgeons [25].

The key difference between nurses and physicians is their working conditions. Nurses often work under the collective agreement so that they are not directly employed by the hospitals. On the other hand, physicians are more typically employed by negotiating work agreements with their hospital individually. In addition, nurses have more regular shifts with few changes since the demand for nurses is relatively constant over a planning period. However, the need for physicians for each department in the hospital can vary over a day due to the fluctuation in patients [4]. Thus, physicians can start at different times depending on the numbers of incoming patients and their work conditions under labor contracts with their hospital. Also, physicians cannot switch their shifts individually because of their different specialties and labor contracts, whereas days-off requested by a nurse can be replaced by another nurse because of mutual agreements. The requirements for physician's on-call services adds more complexity to the physician scheduling problem (PSP) than the nurse scheduling problem (NSP) [3].

Despite these practical differences between nurse and physician scheduling problems, many papers successfully utilized solution methods for the NSP to solve the PSP since their mathematical formulation are not that different [24]. Both allocate medical employees to work shifts over a planning period subject to hard constraints that cannot be violated and soft constraints associated with employee satisfaction. In general, the hard constraints represents legal regulations, whereas the soft constraints represent individual preferences.

Cheang et al. [1] and Burk et al. [2] presented a bibliographic survey of key models and approaches for NSP and outlined the current state of the art of the field until 2004. These briefly cover mathematical programming (MP) methods and metaheuristics. Levner et al. [26] focused on the cyclic or periodic nature of nurse scheduling. Recently, Brucker et al. [17] classified personnel scheduling problems including a model for NSP and discussed their complexity. Vanhoucke and Maenhout [27] proposed a benchmark problem generator and indicators that assess the degree of complexity of nurse scheduling problems and the robustness of proposed approaches. In general, it has been shown that optimization-based schedules are superior to those generated manually by the head nurse [28].

As mentioned above, the demand for physicians is not uniform over a shift. This makes it harder to match the supply of physicians with their demand by a patient. Thus, PSP has more complicated constraints than NSP by individual labor contracts with their hospital, whereas NSP has relatively few general rules over different hospitals [4]. By this complex nature of physicians, PSP has received very less attention than NSP.

Beaulieu et al. [3] were one of the first papers to develop a shift-based model for scheduling emergency room physicians by incorporating days-off requests and physician preferences into the mixed-integer programming (MIP) formulation. Carter and Lappierre [22] analyzed several scheduling procedures for emergency room physicians currently in use at six different hospitals and provided a collection of rules and recommendations to improve them. Rousseau et al. [29] presented a combination of three approaches that included constraint programming (CP), local search, and genetic algorithms to solve the PSP. Brunner et al. [4] introduced a heuristic decomposition strategy and an implicit formulation for the flexible shift scheduling problem of physicians in hospitals. Carrasco [30] studied a simple random and greedy strategies to schedule the physicians in the pediatrics department of a hospital in Spain over a one-year planning horizon.

1.1.2 Resident Schduling Problem (RSP)

The path to becoming a physician involves undergraduate education, medical school, and graduate medical education (GME). Residency is a part of GME after undergraduate education and medical school to become a physician. Residents receive additional specialty training under the supervision of more experienced attending physicians as they provide patient care, becoming progressively more independent.

As they provide patient care while being trained, residents are both providers and learners so that resident schduling problems must satisfy both patient care needs and educational requirements. For an optimal learning environment, it has to provide a variety of working activities according to residents' seniority level and guarantee the minimum required number of training experiences. Thus, residents are alternately assigned to different activities for new experiences, and this makes resident schduling problem (RSP) more complex than the full-time physicians.

In addition, RSPs are often challenging to solve for many reasons, including heterogeneity of the workforce, personal preferences, fatigue issues, concerns over continuity of care, etc. When scheduling residents, we are faced with all of these challenges and more, including significant diversity in the residents' skill sets and levels of training as well as the need to not only provide adequate patient care but to also ensure that the residents meet their educational requirements.

Also, we note that the quality of a schedule can have a significant impact on the well-being of residents and the quality of patient care services. Poor-quality schedules can lead to fatigue, lack of sleep, professional burnout, and even depression. Recently, the emotional well-being of residents has received much attention since many residents are at high risk for serious mental health issues [31, 32, 33, 34, 35]. According to the American Foundation for Suicide Prevention, about 300 to 400 physicians commit suicide each year [36]. In addition, emotional distress and fatigue in residents can lead to medical errors and negatively impact patient care [37, 38, 39]. Under the Libby Zion Law [40], there were many reforms by the Accreditation Council for Graduate Medical Education (ACGME), making new regulations and common program requirements for all medical resident training institutions in the United States to balance between *"limits on duty hours and the need for educational experiences*" [5, 41].

There are several types of residency scheduling. For example, the *block schedule* specifies periods of time (often month-long) where residents are assigned to specific services over the course of the academic year. The *call schedule* defines periods of time over which residents are responsible for taking calls outside of normal working hours in order to meet patient needs. The *shift schedule* (for appropriate services) assigns residents to specific tasks at specific times within a given block. For example, an emergency department is typically staffed for 24 hours a day, 7 days a week, and the shift schedule determines which residents work when. **In this paper, we focus on residency shift scheduling for a pediatric emergency department.**

The literature on residency scheduling dates back at least as far as 1994 with the paper by Ozkarahan [5]. It proposed a goal programming (GP) model focusing on both the requirements of the residency program and the desires of residents. Sherali et al. [42] developed a MIP model to solve the night allocation problem of residents for on-call schedules while considering departmental staffing and skill requirements as well as individual preferences. Franz and Miller [43] addressed the problem of scheduling the daytime training rotations of the second and third year family practice medicine residents. This problem has a set of educational requirements for a wide variety of clinic experiences that must be satisfied to ensure that the residents gain sufficient experience. Day et al. [44] formulated an IP model for scheduling the weekly work hours of surgery residents considering the rules of the ACGME, a private non-profit organization that regulates resident duty hours. Topaloglu [45] proposed a multi-objective programming model for scheduling residents with different seniority levels. Cohn et al. [46] combined heuristic and MIP approaches to make a one-year residents schedule for the on-call shifts at three different hospitals staffed by the psychiatry residency program at Boston University School of Medicine. Topaloglu and Ozkarahan [47] used an MIP and a column generation approach via CP for scheduling residents shifts over four-week planning periods under the ACGME requirements. Güler et al. [48] proposed a GP model for scheduling the night and weekend shifts of the residents in an anesthesia and reanimation department, and Güler et al. [49] proposed a model for the assignments of residents to out-patient clinics in a physical medicine and rehabilitation department. Recently, Bard et al. [50] suggested a network model for monthly scheduling that assigned residents to clinic duty during their training in internal medicine. Guo et al. [6] presented a generic version of the RSP that produces a one-year schedule and showed a proof of its NP-completeness. They also developed a greedy heuristic and analyzed its performance using data provided by the internal medicine residency program at the University of Illinois College of Medicine at Urbana-Champaign. Bard et al. [51] proposed several integer programming-based heuristics for constructing annual block schedules for family medicine residents with continuity clinic considerations. Bard et al. [52] focused on block schedules for internal medicine residents.

Within an emergency department, there are unique characteristics that can influence schedule quality [3, 22, 23, 24]. Beaulieu et al. [3] proposed a GP model for scheduling emergency room (ER) physicians over a six-month period considering a large number of various rules. Carter

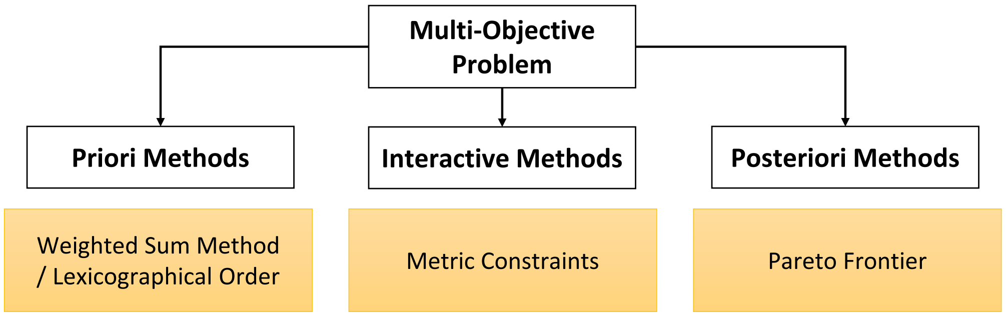


Figure 1.1: Classification of MOP approaches

and Lapierre [22] addressed the problem of ER physicians in six different hospitals by managing ergonomic constraints for the circadian rhythm. Topaloglu [23] used GP to address the monthly shift scheduling problem for emergency medicine residents. Gendreau et al. [24] examined several methods for physician scheduling in the ER.

1.2 Multi-objective Optimization Problem (MOP)

In multi-objective optimization problem (MOP), there are several conflicting objective functions, and there is no single optimal solution that simultaneously optimizes all the objective functions. Thus, the decision makers cannot find an optimal solution relative to all objectives. Instead, they look for the most preferred solution. Thus, **the concept of optimality is replaced by Pareto optimality, the set of Pareto-dominant solutions.**

1.2.1 Classification of MOP Approaches

According to the decision time when the decision maker provides their preference information, MOP methods can be classified into three categories: *priori*, *interactive*, and *posteriori* methods [53].

In *priori* methods, the decision maker is asked to express their preferences clearly before the solution process through relative importance or weights to the objective functions. For example, the weighted sum and lexicographic approaches are examples of *priori* methods. Given weights from the decision maker, weighted sum methods solve for a positively weighted convex sum of the multiple objectives [54]. On the other hand, if there exists strict relative priority between metrics, we could use lexicographical optimization which optimizes each metric in hierarchical order. After making one single-objective scalar function from the multiple objectives, *priori* methods can utilize traditional optimization techniques based on single-objective function seamlessly. However, *priori* methods are only valid if the trade-off weight or relative priority from the decision maker matches with true preference. Normally, it is very difficult for the decision maker to quantify preferences or weights accurately beforehand.

In *interactive* methods, the decision maker progressively gives preference information toward the most preferred solution. The decision maker alternatively involves two phases: solution process and evaluation process. The decision process converges to the most preferred solution by evaluating solutions iteratively until the decision maker is satisfied with the solution. This method is useful when goals or weights are not clear yet. However, the decision maker never sees the whole set of possible choices or an approximation of it. Hence, the most preferred solution after a few iterations is a myopic choice from what they have evaluated so far [55].

In *posteriori* methods, the set of potential solutions are generated, and later the decision maker selects one among them based on preference. The decision process is divided into two independent phases: first, it generates all the possible alternatives; subsequently, the decision maker is involved in selecting the most preferred one among them when all possible choices are on the table. The decision maker is involved only in the second phase. Thus, this method is useful when the interaction with the decision maker is rarely available. Also, since none of the potential solutions have been left undiscovered, it reinforces a decision maker's confidence in the final judgment [55]. However, many real-world size MOPs are computationally expensive to generate all potential choices. Also, the size of all the possible alternatives grows exponentially. Thus, it is necessary to provide a small representative subset to approximate the whole range space of choices. The weighted sum methods and the ϵ-constraint methods [56] are commonly used for*posteriori* methods.

1.2.2 Posteriori Approaches

To generate the set of potential solutions, we need the concept of Pareto optimality. Intuitively, any feasible solutions that are dominated by another solution by giving worse or equal values in all objectives are not attractive. Therefore, dominated solutions will not be selected by decision makers. We say a solution is Pareto optimal (or Pareto dominant) if and only if there is no alternative solution that can improve one of the objectives without worsening any other one. The goal of *posteriori* methods is to generate the set of Pareto-dominant solutions to ensure that no solution is excluded which would be the most preferred one.

Posteriori methods fall into two classes: evolutionary algorithm (EA) based methods and MP based methods. While each run of EA produces a set of potential solutions simultaneously, MP produces one potential solution per optimization search. Although EAs seem to be powerful search mechanisms for the large size of real-world problems, they only generate an approximation or a subset of Pareto optimal solutions. On the other hand, MPs can find all possible potential solutions. In this paper, we focus on MP based methods.

1.2.2.1 Evolutionary Algorithms (EA)

EAs are stochastic optimization algorithms in which the Pareto frontier of an MOP is approximated by evolving a population of solutions. The algorithm starts by generating a set of candidate solutions and subsequently evolves it though two biological principles: *selection* and *variation*. The selection represents the competition among living beings. A scalar fitness value defines the quality of each solution and the chance to reproduce. Solutions of high quality (better fitness value) are more likely to survive and to reproduce their genetic information. A stochastic selection process simulates the natural selection process. On the other hand, the variation imitates the natural capability of creating new living beings through recombination and mutation. Although the underlying principles are simple, EAs have proven as a general, robust and powerful search algorithm for MOPs [57, 58]. A genetic algorithm (GA) [59, 60] and cuckoo optimization algorithm (COA) [61] are examples for EA based methods.

1.2.2.2 Mathematical Programming (MP)

MP-based methods are exact methods to find the complete Pareto frontier of an MOP. In MP-based methods, the original MOP is converted to a single-objective problem through a parameterized objective function or constraints. The parameters are algorithmically set by the multi-objective optimization algorithm. Basically, several single-objective problems with different parameter settings are solved iteratively, and each single run is independent problem of the underlying MOP. The most widely used scalarization methods in MOPs are the*weighted sum method* and the *ϵ-constraint method*.

Given an MOP with n objectives $\{f_1, \cdots, f_n\}$ over a feasible set $\mathcal{S}$ and weights $w \in \mathbb{R}^n$, the weighted sum method is formulated as follows:

$$\min \sum_i w_i f_i(x)$$
$$\text{subject to: } x \in \mathcal{S}$$

where $f_i(x)$ is the ith objective function, w_i is the weight for $f_i(x)$, and $\mathcal{S}$ is the feasible solution space.

The weighted sum method converts multiple objectives into a single scalar objective function by forming a linear combination of them. On the condition that all weights are positive, it can be proved that the minimizer of the single-objective function is a Pareto-optimal solution for the original MOP. Given weights w, assume that a feasible solution x minimizing the single-objective function $f_w = \sum_i w_i f_i(x)$ is not Pareto optimal. Then, there exists a solution y dominating x (i.e., there exists some $i \in \{1, \cdots, n\}$ such that $f_i(y) < f_i(x)$ and $f_j(y) \leq f_j(x)$ for all $j \neq i$). Therefore, it follows that $f_w(y) < f_w(x)$. This contradicts the assumption that $f_w(x)$ is minimized.

On the other hand, in the ordinary ϵ-constraint method, only one of the objective functions is optimized by adding the other objective functions as constraints to the original solution space $\mathcal{S}$.

For any $\epsilon \in \mathbb{R}^{n-1}$ and some $i \in \{1, \ldots, n\}$, the ϵ-constraint method is formulated as follows:

$$\begin{aligned} & \min f_i(x) \\ \text{subject to: } & x \in \mathcal{S} \\ & f_j(x) \leq \epsilon_j, \qquad && \forall j \neq i \end{aligned}$$

where ϵ_j are the upper bounds of satisfaction level on the constrained objectives.

By selecting the main objective function and adjusting satisfaction levels properly on the right-hand side of the constrained objectives, it can find any Pareto-optimal solution for the original MOP. However, there is no guarantee that the ordinary ϵ-constraint method always generates Pareto-optimal solutions. Theoretically, the obtained optimal solution is guaranteed to be Pareto-optimal solution only if all $(n-1)$ constrained objective functions are binding. We demonstrate the weak Pareto solution in Figure 1.2. Mavrotas [55] suggested the augmented ϵ-constraint method by introducing positive surplus variables, also known as slack variables to guarantee that the obtained solutions are Pareto-optimal. In the augmented ϵ-constraint method, the main objective function is augmented by the sum of the surplus values as follows:

$$\begin{aligned} & \min f_i(x) - \delta \sum_j s_j \\ \text{subject to: } & x \in \mathcal{S} \\ & f_j(x) + s_j = \epsilon_j, \qquad && \forall j \neq i \\ & s_j \geq 0 \end{aligned}$$

In the literature, several variations of the ϵ-constraint method have been developed to improve its performance [55, 62].

The ϵ-constraint method has several advantages over the weighted sum method. First, it only produces a Pareto *extreme* solution in the original linear programming (LP) feasible region. On the

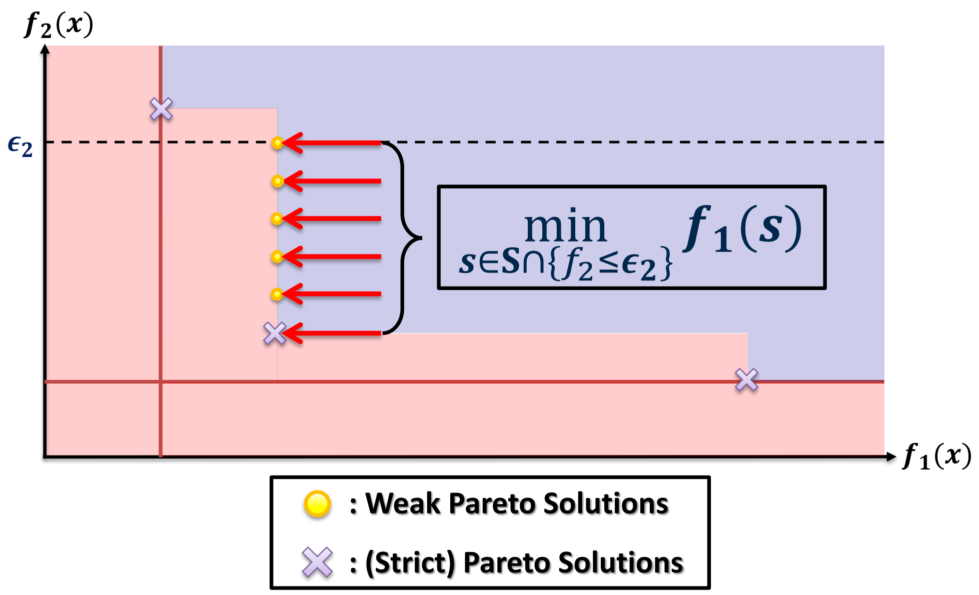

$f_2(x)$
ϵ_2
$\min_{s \in S \cap \{f_2 \le \epsilon_2\}} f_1(s)$
$f_1(x)$
: Weak Pareto Solutions
: (Strict) Pareto Solutions

Figure 1.2: Weak Pareto solutions on the Pareto frontier

contrary, ϵ-constraint method can generate non-extreme Pareto solutions with the proper selection of ϵ. Second, the weighted sum method could not generate all Pareto-optimal solutions for the non-convex feasible region such as IP or MIP. These *unsupported* Pareto solutions are illustrated in Figure 1.3. Third, there are many combinations of weights that result in the same Pareto extreme solution. Finally, the meaning of weights or scaling of the objective functions in the weighted sum method is ambiguous. However, in the ϵ-constraint method, the value of ϵ has more clear meaning since it represents upper bounds on constrained objectives. Thus, decision makers can appropriately select ϵ to move toward their preferred solution.

1.2.3 Healthcare Scheduling with Multi-Criteria Decision

Unlike many industrial optimization problems in which the dominant objective is to minimize cost or maximize profit, many healthcare problems involve multiple, potentially conflicting, criteria. In general, nurses work under collective union agreements, and this yields the staffing cost pressure on hospitals. Therefore, the goal of NSP is to minimize salary cost and to maximize their individual satisfaction simultaneously. On the other hands, PSP focuses more on personal preferences [24]. In

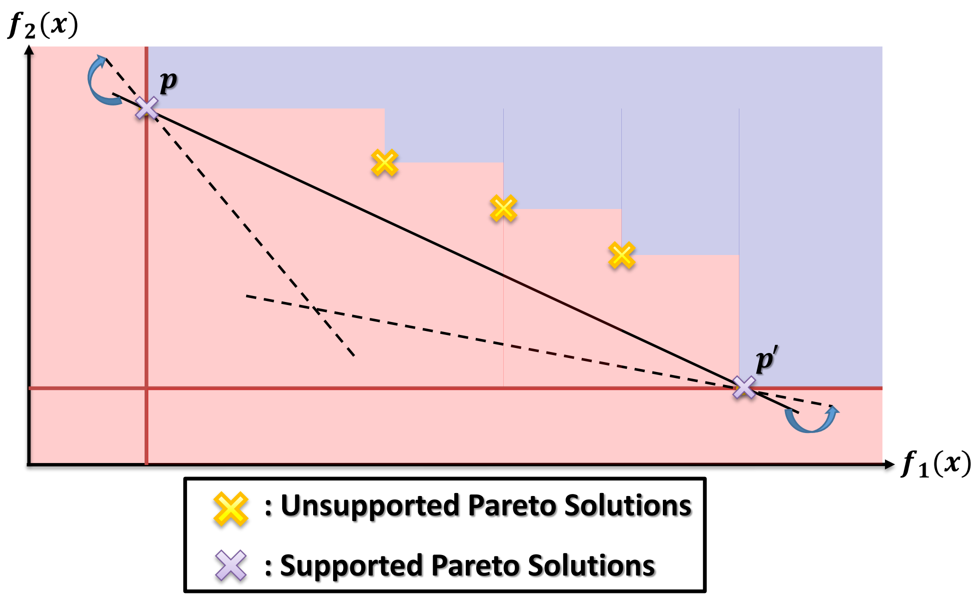


Figure 1.3: Supported and unsupported Pareto solutions on the non-convex Pareto frontier

RSP, it needs to provide a good balance between education and patient care activities [23]. There are patient needs, educational requirements, and personal preferences that all need to be taken into consideration when evaluating the quality of a schedule.

Within the healthcare scheduling literature, several papers have specifically recognized this need to address multiple objective criteria. The most commonly discussed methods are GP (i.e., seeking to meet targets), weighted sum approaches (converting multiple objective criteria into one scalar objective function), or some combination of the two [63]. Many nurse [64, 65, 66], physician [3] and residency scheduling papers [5, 23, 45, 47, 48, 49, 50] formulate the problem with both hard and soft constraints and handle soft constraints as multiple objectives with GP.

1.3 Vehicle Routing Problem (VRP)

The vehicle routing problem (VRP) is to find the optimal (i.e., minimum cost) design of routes for a fleet of vehicles from a depot to serve a given set of customers. Many variations of the VRP have been considered since Dantzig and Ramser [67] first proposed the basic VRP. The VRP has been intensively studied since many combinatorial optimization problems and their practical applications can be solved effectively by the models and the algorithms for the VRPs.

1.3.1 Literature Review

Exact approaches to heterogeneous fleet VRP (HVRP), and to VRPs in general, are typically based on IP formulations. These are usually either vehicle flow formulations or set partitioning formulations.

Vehicle flow formulations use binary decision variables to indicate whether a given vehicle (or vehicle type) travels between two customers in the optimal solution. Formulations of this type for the VRP can be first found in Garvin et al. [68], and Gavish and Graves [69, 70]. The heterogeneous VRP was formulated in Gheysens et al. [71] and Golden et al. [72] by using three-index binary variables x_{ij}^k as vehicle flow variables that take value 1 if a vehicle of type k travels directly from customer i to customer j, and 0 otherwise. Such formulations must include subtour elimination constraints, however, which often lead to heavily fractional solutions to the LP relaxation and, in turn, weak lower bounds on the optimal objective value.

VRP is also often modeled, therefore, with a set partitioning formulation, as originally proposed by Balinski and Quandt [73]. The set partitioning formulation for heterogeneous VRPs associates a binary variable x_r^t with each feasible route r and vehicle type t. However, given that the number of candidate routes is exponentially large, it is often impractical to explicitly enumerate all routes in the model. Instead, *delayed column generation* algorithms are used, using a pricing problem such as those posed by Rao and Zionts [74], Foster and Ryan [75], and Agarwal, Mathur, and Salkin [76].

Both of these types of exact approaches, however, can suffer from significant computational issues. In fact, all the VRP variations are NP-hard as they are a generalization of the traveling salesman problem (TSP), which is theoretically proven as NP-hard. Even pure VRP problem instances with more than 100 nodes can typically not be solved to optimality in a reasonable amount of time. Variations such as HVRP, capacitated VRP (CVRP) and distance-constrained

VRP (DVRP) are even more complex to solve [77, 78, 79, 80]. Therefore, many heuristics and hybrid algorithms have also been developed in order to find high-quality solutions in acceptable run times [81, 82].

Dating back as far as Dantzig and Ramser [67], many heuristics have been published. The savings heuristic by Clarke and Wright [83] iteratively merges partial routes to form a set of feasible routes, and the sweep algorithm by Gillett and Miller [84] sequentially generates non-overlapping feasible routes by rotating a half-line rooted at the depot. In addition to classical VRP heuristics, several meta-heuristic approaches have been developed as well. The best-known examples of meta-heuristics include tabu search [85], simulated annealing [86], and genetic algorithms [87]. Specific meta-heuristics for VRP are provided in Taillard [88], Gendreau [89], Nagata [90], and Prins [91]. Many of the most successful VRP heuristics, however, rely on the use of distance as a surrogate for cost, assuming a constant cost per mile for each vehicle and a given distance matrix. Because we are specifically interested in the case where distance and cost may not be correlated, we cannot build directly on these heuristics for time-constrained heterogeneous vehicle routing problem (TCHVRP). We instead present an alternative approach based on a path-based model, as we discuss later in the paper.

There is a rich and extensive literature on VRP in general. We refer the reader to Laporte [92], Toth and Vigo [93, 94] and Golden, Raghavan, and Wasil [95] for comprehensive surveys. Additional studies and algorithms focused specifically on HVRPs can be found in Baldacci [96].

CHAPTER 2

Interactive Shift Scheduling Problem

2.1 Motivation

Residency schedules are frequently built by Chief Residents (as was the case at Mott), either manually or with the limited help of some basic spreadsheet tools. This is not only a tedious and a time-consuming process, but it is often difficult to find even a feasible schedule, fully satisfying all rules and requirements, that is of high-quality with respect to patient care, educational requirements, and personal preferences.

A natural approach to solving complex combinatorial problems such as those encountered in residency scheduling is through IP; this is often a successful way to find *feasible solutions* to the residency shift scheduling problem [97].

On the other hand, it remains a significant challenge to assist the Chief Residents in evaluating and selecting from the potentially large set of feasible solutions because of the lack of a clearly-defined single objective function. Unlike many industrial scheduling applications, the goal of a residency scheduling problem is not to minimize cost. Rather, there are many different criteria, associated with quality of patient care, resident personal preferences, and educational needs, under which a schedule is evaluated.

One approach commonly used in addressing multi-criteria objective problems is to put weights on each of the metrics in order to establish a single objective function. In our experience, the weighted sum approach is not appropriate for building resident shift schedules, for three reasons (see similar observations in [98]). First, it can be very challenging for the Chief Residents to identify weights that accurately represent their preferences. Second, even if the weights could be inferred by reviewing prior chosen schedules, preferences often change from schedule to schedule. For example, matching appropriate levels of seniority to shift types may be a high priority at the beginning of the academic year, as new residents join the program, whereas priority towards the end of the year might shift to facilitating residents' fellowship interview schedules. Third, the weighting may be non-linear. For example, the Chief Residents might be willing to give up one vacation request in order to improve patient hand-offs on one shift, but not be willing to give up 5 requests to reduce 5 hand-offs. Thus, solutions generated by using a weighted-sum objective often will not reflect the true preferences of the Chiefs.

Recognizing this limitation of assigning weights to trade off between different scheduling criteria, our goal is to develop a mechanism by which Chief Residents can develop high-quality schedules in a timely manner, enabling them to ensure that all rules and requirements are met while focusing on the quality of the schedule with respect to patient care, resident educational needs, and personal requests.

To develop such an iterative, interactive mechanism, we leverage three key ideas, in a collaborative manner that merges both IP and the Chief Residents' expertise. **First**, we note that integer programming feasibility problems can often be solved very quickly, as is the case in this application. **Second**, we have observed that it is often much easier for the Chief Residents to qualitatively describe to us what they like and dislike about a specific schedule than it is for them to quantify, in abstraction, their preferences. **Third**, we have also learned that the Chiefs are not actually seeking an "optimal" schedule – in fact, they recognize that there are trade-offs that must be made between different criteria and that multiple schedules may all be acceptable. Instead of seeking optimality, their focus is on finding a schedule that is of high quality with respect to many different metrics.

Given these three points, we have developed a simple but effective process for developing monthly schedules in collaboration with the residency program's Chief Residents. We begin by importing the month's data (resident cohort, educational requirements, external requirements, vacation requests, etc.) into an integer programming-based decision support tool that generates a schedule satisfying all hard constraints. We next review with the Chiefs the value of this solution relative to each of their objective criteria. The Chiefs then identify undesirable characteristics (e.g. "Too many bad sleep patterns have been assigned"), and we add constraints to the feasibility problem bounding the associated metrics and re-run the algorithm in real-time, either leading to an improved solution or returning a certificate of infeasibility. The process repeats, with constraints on the objective criteria being successively tightened and loosened as a function of the Chiefs' preferences, until the Chiefs are satisfied with the solution.

We present our collaborative work with the University of Michigan C.S. Mott Children's Hospital in building monthly schedules, focusing on both our IP formulation and the iterative, interactive approach in which we use this integer program.

2.2 University of Michigan Pediatric Emergency Department Scheduling (PEDS)

2.2.1 Background

Residency is the phase of graduate medical education after completing medical school while continuing training to become a physician. Residents are trained in progressively more specialized areas under the supervision of more experienced attending physicians, becoming increasingly more independent as they advance in seniority.

Medical students enter residency through the National Resident Matching Program (NRMP). The "match" happens annually in the spring, with medical students assigned ("matched") to residency program throughout the United States. Across the U.S., there are more than 30,000 residency positions that are filled by the national matching program annually. Pediatric residents make up almost 10% of this population.

Residency can take anywhere from three to upwards of seven years depending on the specialty chosen. For example, pediatric residency or internal medicine training takes three years. General or orthopedic surgery can take five years to complete. After completion of a residency program, one to three years of additional training in a subspecialty may be chosen by some physicians for further training. For example, in pediatrics, a pediatrician may train to become a neonatologist. In internal medicine, a physician may subspecialize in cardiology.

The University of Michigan Pediatric Emergency Department provides services 24 hours a day, 7 days a week to care for children with medical problems that cannot wait or are too severe to be seen by their primary care providers. The emergency department (ED) offers unscheduled care ranging from common minor pediatric problems to major medical and traumatic emergencies. Approximately 20,000 children a year are seen at the University of Michigan Pediatric Emergency Department. It is designated a Level I Pediatric Trauma Center which means it is certified by the American College of Surgeons Committee on Trauma to provide the care to the most severely injured children.

In addition to attending physicians and other clinical personnel, the ED is staffed by medical residents of varying levels of seniority and from a variety of training programs (e.g. Pediatrics, Internal Medicine, Psychiatry, etc.), with roughly 30 *interns* (i.e. first year residents) joining the program each year while more senior residents remain for additional years of training.

In order for a pediatric resident to complete their graduate medical education, they must complete rotations that encompass the competencies set forth by the ACGME. To be successful, the pediatric resident must have inpatient and outpatient training. The training also includes subspecialty time in the emergency department. Over the course of 3-years at the University of Michigan, a pediatric resident will spend 12-weeks in the ED.

2.2.2 PEDS Problem Statement

Our research is based on the assigning of residents to shifts within the Pediatric Emergency Department of the University of Michigan C.S. Mott Children's Hospital. The ED is staffed with residents from many programs including Pediatrics, Emergency Medicine, and Family Medicine. Thus, it has a non-homogeneous work force, with each resident having different skills, requirements, and preferences. In a given year, there can be 4–6 residents from Family Medicine or Emergency Medicine who rotate through the pediatric emergency department for additional training.

There are seven overlapping 9-hour shifts that are scheduled every day to staff the Pediatric Emergency Department: Shift 1 (7 AM – 4 PM), Shift 2 (9 AM – 6 PM), Shift 3 (12 PM – 9 PM), Shift 4 (4 PM – 1 AM), Shift 5 (5 PM – 2 AM), Shift 6 (8 PM – 5 AM) and Shift 7 (11 PM – 8 AM). Shifts 1 and 2 are are considered to be *morning shifts*, Shift 3 is considered to be a “flex” shift, Shifts 4 and 5 are *day shifts*, and Shifts 6 and 7 are *overnight shifts*. The flex shift should ideally be staffed by a resident to provide additional support to the attending physicians during the peak hours of the day, but is not required to be staffed. All other shifts must be staffed by exactly one resident.

Residents are assigned to work in the ED for month-long rotations. The senior residents start and end on the first and last days of the month, while the interns transition from one rotation to another on the 27th of each month to ensure a smooth transition. From a scheduling standpoint, the implication of this is that certain shifts at the end of the month (known as “optional shifts”) can be left unfilled to be staffed with incoming interns at the scheduling of the following month.

Note, however, that not all shifts can be staffed by interns – specifically, senior residents must stafff the ED at the busiest times of day. In addition, there is a balance between the number of residents who are rotating through the ED from other departments. In a given year there can be 4-6 residents from family medicine or emergency medicine who rotate through the pediatric emergency department for additional training. These residents also have limitations on when they can be staffed, typically as a function of other regularly-scheduled training activities.

Finally, in addition to their monthly rotations on different services (including the emergency department), residents also maintain a panel of patients whom they care for throughout the year in the form of *continuity clinics*. Ideally, residents’ ED shifts should not conflict with attending their weekly clinics.

The primary University of Michigan Pediatric Emergency Department Scheduling (PEDS) rules scheduling rules are as follows:

- **Coverage:** All shifts must be staffed by exactly one resident except the flex shifts (which can be left unstaffed if necessary) and the optional shifts at the end of the month (which can wait to be staffed by interns coming onto the following month).

- **Invalid Assignments:** There are many reasons why a specific resident cannot be assigned to a specific shift. These include seniority (i.e. interns cannot staff certain shifts), program conflicts (e.g. certain training programs have educational commitments on certain days of the week), and continuity clinics. In addition, we treat vacation requests as hard constraints in this context. Within the PEDS problem, residents are allowed to make personal requests. These are typically only for critical issues (interviews, certification boards, weddings and other major family events, etc.) and as such the Chiefs put highest priority on satisfying these requests. In addition, there is typically enough flexibility in the assignment of shifts that requests can be granted. We discuss this further in Section 2.4.

- **Pre-Assignment:** In some cases, there are specific shifts that *must* be assigned to specific residents. These are often to ensure the completion of educational responsibilities for residents who have been on leave.

- **Pediatric Coverage:** For certain pairs of overlapping shifts, it is necessary to ensure that at least one of the two shifts is staffed by a resident from the Pediatrics program.

- **Duty Hours Restrictions:** Regulatory guidelines require that, after completion of a shift, residents must have at least ten hours off-duty before beginning a subsequent shift.

- **Limits On Consecutive Shifts:** There are limits on the number of consecutive days in a row that residents can be assigned to shifts, and tighter limits on the number of consecutive days in a row that residents can be assigned to overnight shifts.

2.3 PEDS Feasiblity Formulation

The following model defines the feasible set of resident shift schedules, relative to the rules outlined in Section 2.2.2.

Notation

R	set of residents, $r \in \{1, 2, \dots, \lvert R\rvert\}$
D	set of days, $d \in \{1, 2, \cdots, \lvert D\rvert\}$; In our data instances, D is either 30 or 31
S	set of daily shifts, $s \in \{1, 2, \dots, \lvert S\rvert\}$; In our data instances, there are 7 shifts
T	set of training programs, $t \in \{$EMS, PED, EM, FM, $\dots\}$
t_r	training program of resident r, $t_r \in T \; \forall r \in R$
C_r	set of days on which resident r has continuity clinic, $C_r \subseteq D \; \forall r \in R$
W_r	set of days on which resident r can work, $W_r \subseteq D \; \forall r \in R$
F	set of flex shifts, $F \subset S$; In our data instances, shift 3 (i.e., 12 PM – 9 PM) is the only element of F
N	set of night shifts, $N \subset S$; In our data instances, shifts 6 and 7 (i.e., 8 PM – 5 AM and 11 PM – 8 AM) are in N
K	set of shift pairs where $k \in K$ is a pairs of shifts such that at least one must be covered by a resident r of type *PED*, $k \subset S$; In our data instances, $\{$shifts 1 and 2$\}$, $\{$shifts 4 and 5$\}$, $\{$shifts 6 and 7$\}$ are in K
H_r	set of day-shift pairs that cannot be assigned to resident r, $H_r \subset D \times S \; \forall r \in R$
A_r	set of day-shift pairs that must be assigned to resident r, $A_r \subset D \times S \; \forall r \in R$
$J_{(d,s)}$	set of day-shift pairs that would cause a duty-hours violation if assigned in addition to day-shift (d, s), $J_{ds} \subset D \times S \; \forall (d, s) \in D \times S$. Note that (d, s) is included in the set $J_{(d,s)}$
$\overline{D}$	maximum allowable number of days in a row that a resident can work a shift
$\overline{N}$	maximum allowable number of days in a row that a resident can work a night shift

Decision Variables

x_{rds} binary variable, equals 1 if resident r is assigned to shift s on day d; otherwise 0

$\forall r \in R, \forall s \in S, \forall d \in D$

Formulation

$$\min \quad 0 \tag{2.1}$$

$$\text{subject to: } \sum_{r \in R} x_{rds} = 1 \qquad \forall s \in S \backslash F, \forall d \in D \tag{2.2}$$

$$\sum_{r \in R} x_{rds} \leq 1 \qquad \forall s \in F, \forall d \in D \tag{2.3}$$

$$\sum_{(d,s) \in H_r} x_{rds} = 0 \qquad \forall r \in R \tag{2.4}$$

$$\sum_{(d,s) \in A_r} x_{rds} = 1 \qquad \forall r \in R \tag{2.5}$$

$$\sum_{r \in \{r : t_r = \text{PED}\}} \sum_{s \in k} x_{rds} \geq 1 \qquad \forall d \in D, \forall k \in K \tag{2.6}$$

$$\sum_{(d,s) \in J_{(d,s)}} x_{rds} \leq 1 \qquad \forall r \in R, \forall d \in D, \forall s \in S \tag{2.7}$$

$$\sum_{i=d}^{d+\overline{D}} \sum_{s \in S} x_{ris} \leq \overline{D} \qquad \forall r \in R, \forall d \in D \tag{2.8}$$

$$\sum_{i=d}^{d+\overline{N}} \sum_{s \in N} x_{ris} \leq \overline{N} \qquad \forall r \in R, \forall d \in D \tag{2.9}$$

$$x_{rds} \in \{0, 1\} \qquad \forall r \in R, \forall d \in D, \forall s \in S \tag{2.10}$$

Constraints (3.2) ensure that exactly one resident is assigned to every non-flex shift. Constraints (3.3) ensure that flex shifts are covered by at most one resident. Constraints (2.4) ensure that residents are not assigned to prohibited shifts. Constraints (2.5) ensure that residents cover pre-assigned shifts. Constraints (2.6) ensure that at least one of two shifts in a required pair are assigned to a Pediatrics resident. Constraints (3.7) ensure that residents are not assigned to shifts that begin within ten hours of a previously worked shift. Constraints (3.8) ensure that no resident works more than a maximum allowable number of days in a row. Constraints (3.9) ensure that no resident works more than a maximum allowable number of nights in a row.

2.4 Interactive Approach to PEDS

When evaluating the quality of a residency scheduling, there are several metrics of importance to be evaluated, from the varying perspetives of the residency program and its educational needs, the providing of patient care, and the residents' personal preferences and well-being. Furthermore, these metrics are in some cases more qualitative than quantitative. Thus, schedules must be evaluated holistically; it is difficult to assign an abstract objective function to find the "best" solution without user engagement.

As such, the emphasis of our work has been not only on developing optimization-based tools that enable us to quickly find feasible solutions that satisfy all of the firm requirements, but also on developing an inteactive collaborative process in which the operations researchers and the Chief Residents can work together in an efficient and effective manner to jointly develop high-quality solutions.

We begin by noting the importance of deep collaboraiton even in defining the feasibility requirements and identifying quantifiable metrics. In our experience, it is difficult for the Chiefs (who only serve in their role for a year, have no scheduling training, and are learning solely based on institutional knowledge passed on from year to year) to clearly and completely articulate all of the requirements. Instead, we learned them by trial and error – building schedules based on the rules we know, then having them review them and identify violated requirements for us to incorporate into the program.

Likewise, we worked together, in evaluating individually schedules, to provide quantifiable metrics to help guide our assessment. In particular, the following are of particular importance to the Chiefs:

- **Total Shift Equity:** Because not all residents are on service for the same amount of time (e.g. some have a full month and some have a half-month assignment), not all residents should perform equal numbers of shifts. The Chiefs do, however, pay careful attention to how many shifts each resident is working, seeking to be equitable.

- **Night Shift Equity:** Similarly, the Chiefs pay careful attention to how many night shifts are assigned to each resident, to ensure fairness.

- **Bad Sleep Patterns:** A *bad sleep pattern* is a sequence of shifts that are legal from a duty-hours perspective, but undesirable relative to circadian rhythm. For example, if a resident works a morning shift on Monday, an overnight shift on Tuesday, and then another morning shift on Thursday, it is difficult for them to match their sleep schedule to their work. (for more details, see Table 2 in [99]). Whereverable possible, the Chiefs prefer to avoid assigning bad sleep patterns.

- **Post-continuity Clinic Shifts:** Although it is legal to work an ED shift immediately after a continuity clinic, the Chiefs again prefer to avoid such assignments because it requires the resident to either leave the clinic early or arrive at the ED late. It is also a long assignment for the resident to work both back-to-back.

- **Intern-undesirable Shifts:** Certain shifts are fully prohibited from interns being assigned to them, as noted in Section 2.2.2. Other shifts are allowed, but the Chiefs prefer to avoid them where possible.

- **Covered Optional Shifts:** As noted earlier, interns begin their ED rotations on the 27th of the preceding month. Thus, when building a monthly schedule, it is permissible to leave shifts on the 27th and later unfilled, to be filled when building next month's schedule. This can be difficult, however, if the following month is tightly constrained. Thus the Chiefs may not want to leave too many optional shifts uncovered if they anticipate difficulties in the building the following month's schedule.

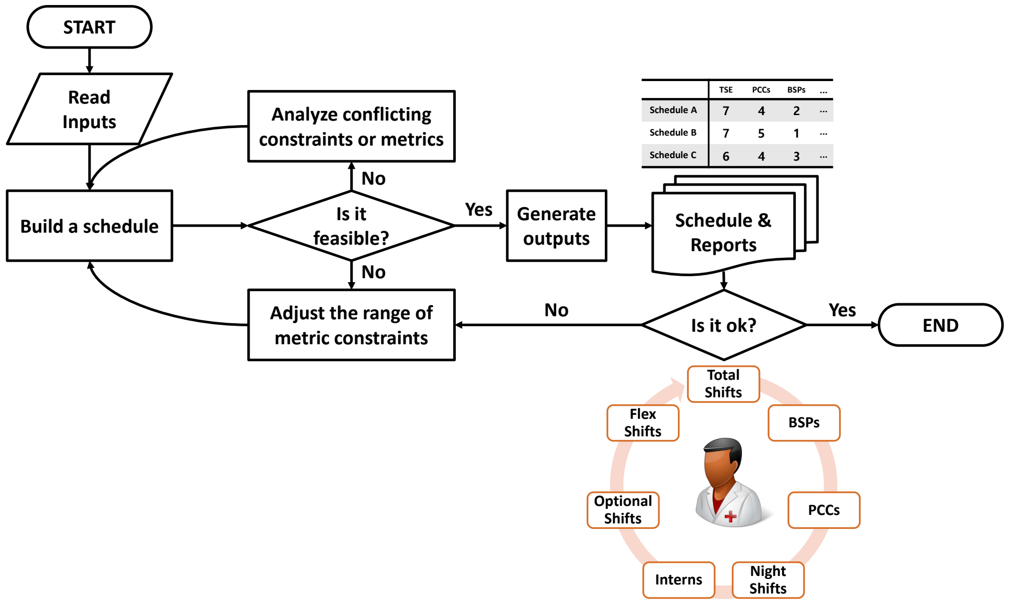
- **Uncovered Flex Shifts:** Similarly, it is not required that all flex shifts be covered, but these correspond to busy times during daily cycle of the ED's patient volume and therefore when possible it is desirable to staff them.

For the past several years, the following process has been effective in building the monthly ED residenct shift schedules: First, the Chiefs provide the team with a set of input files that contain the upcoming month's residents, their continuity clinics schedules, their time-off requests, etc..

Next, we first try to find a feasible schedules that satisfies all time-off requests and continuity clinics. If it is infeasible, we analyze the problem instance to identify causes of infeasibility and then work with the Chiefs to identify which clinics to cancel and/or which requests to deny (these are the highest priority and most sensitive, and need to be assessed individually).

Once we have a feasible schedule (i.e. have a valid set of continuity clinics and time-off requests that can be honored), we focus on the metrics. Specifically, rather than imposing an explicit objective function, we use upper bounds to limit the individual metrics, with the Chiefs deciding which bounds to tighten or loosen at each iteration. For example, if a given schedule has a large number of post-continuity clinics, we might relax all other bounds and minimize this metric to get a sense of what is possible, then slowly lift this bound to gain better solutions with respect to other metrics. It is by no means a scientific process, and yet it has continuously enabled the Chiefs to quickly converge on solutions with which they are satisfied. We illustrate this process in Figure 2.1.

It is worth noting that this process not only provides a high-quality schedule for the Pediatric Emergency Department, but also has enormously educational value. By working together and learning how each of us brings our own perspective to the problem, we find each month that we converge more quickly, e.g. the operations researchers in building the first initial draft of the schedule are progressively better able to predict the Chiefs' preferences. This is also providing junior students (typically, undergraduates now do the bulk of the work in actually running the model) with valuable hands-on experience with a real-world problem, including the monthly interactions with the Chiefs. Finally, the Chiefs themselves have consistently demonstrated not only a willing-



	TSE	PCCs	BSPs	...
Schedule A	7	4	2	...
Schedule B	7	5	1	...
Schedule C	6	4	3	...

Figure 2.1: Iterative, interactive approach in the PEDS problem

ness but an eagerness to learn as much as they can about the model and how it works, and this has changed the way they view other problems that they encounter in their work.

2.5 Computational Results

In this section, we present computational experiments to demonstrate the tractability and effectiveness of this approach. Briefly, because the feasibility problems can be solved so quickly (on the order of minutes or even seconds) and because we have developed an ever-increasing understanding of the Chiefs' preferences which enables us to produce high-quality initial schedules, this collaborative process typically takes on the order of at most one to two hours per monthly schedule, frequently with only a few minor changes requested by the Chiefs relative to the initial schedule. This is significantly less than the twenty-plus hours per month that was required of the Chief Residents to build schedules manually prior to the development of this integer programming-based approach. Far more importantly, the resulting schedules are of significantly higher quality than those generated manually. Note that this section is extracted from our published paper [99].

2.5.1 Data Sets

Our computational results are based on real-world data from the C.S. Mott Children's Hospital Pediatric Emergency Department in the 2012–2013 academic year. The C.S. Mott Children's Hospital pediatric ED is a level 1 pediatric trauma center located in Ann Arbor, Michigan. It served nearly $19,000$ patients in the 2010–2011 academic year (AY), and more than $23,000$ patients in AY 2012–2013. Resident schedules from AY 2010–2011 and 2012–2013 were chosen for review. The AY 2010–2011 was the most recent year that the schedule was constructed completely by hand, and AY 2012–2013 was the first complete year scheduled using our interactive method. The intervening AY 2011–2012 was a year of transition and was omitted.

To assess schedule quality, we evaluated 4 metrics of schedule quality that were compared between the 2010–2011 and 2012–2013 academic year: total shift equity (TSE), night shift equity (NSE), occurrence of post-continuity clinic shifts (PCCs) immediately following outside clinic responsibilities, and occurrence of challenging bad sleep patterns (BSPs).

The TSE and the NSE refer to variance in shift number among residents in any given month. The PCCs were chosen as a negative quality metric because of the difficulty of preceding an ED shift with clinic duty or other outside requirement. The BSPs were defined as consecutive shift assignments that yield a difficult sleep schedule for residents and were determined by informally surveying senior residents on challenging shift transitions.

Data collected included the monthly resident complement, total shifts per resident, night shifts per resident, PCCs, and BSPs. These variables were calculated for each month in the study years, then averaged within each year. Total shift distribution variance per month and night shift distribution variance per month were also calculated and averaged within each year. Student's t-tests were used to compare the data between the study years. Pediatrics Chief Residents were informally surveyed throughout the project on the amount of time necessary to create a schedule, both manually and utilizing our interactive method.

We solved the PEDS problem by impelementing our iteractive aprroach using C++ with CPLEX

Year	2012						2013					
Month	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.	Jan.	Feb.	Mar.	Apr.	May	Jun.
Time(sec)	0.28	0.39	0.11	1.73	0.55	0.16	0.39	0.23	0.44	0.3	0.47	1.42

Table 2.2: CPU time per monthly schedule for 2012–2013 academic year

API version 12.1 on a computer with an Intel Xeon 3.20 GHz processor and 8 GB of memory. Statistical analysis was completed using Minitab 16.

2.5.2 Results

A critical key to the success of this approach is that the individual feasibility problems be fast enough to solve that they can be reviewed and re-run in an interactive, hands-on manner. Table 2.2 shows that each instance was solved in a seconds.

Building a typical monthly schedule takes up to 3 hours for the Chiefs to generate the input data files (mainly collecting the residents' requests and continuity clinic schedules). The operations research team then typically spends a few hours generating a high-quality initial draft schedule, and then the complete team typically meets for an hour or two to review, refine, and finalize the schedule. In contrast, prior Chiefs report that each month's schedule required 12 to 16 hours to build manually, plus 10 to 12 additional hours of later corrections, all of this time on behalf of the Chiefs, and with lesser quality in the final schedule. In sum, the total time to build a monthly schedule was between 22 and 28 hours by hand and 4 to 6 hours using our interactive method. Using the interactive method resulted in a 79% to 82% time reduction per month.

The number of residents working in the ED per month had a mean (μ) of 15.7 and a standard deviation (σ) of 2.06 in 2010–2011, and had $\mu = 11.2$ and $\sigma = 1.18$ in 2012–2013. As some residents took a 2-week vacation block during their ED rotation, for each statistic, the number of residents was normalized per month, such that a resident who was working in the ED for 15 days in a 30-day month was counted as 0.5 residents.

Each resident worked a mean of 15.8 ($\sigma = 1.68$) total shifts per month in 2010–2011 and 16.4 ($\sigma = 1.62$) total shifts per month in 2012–2013. The mean number of monthly night shifts per resident was 4.7 ($\sigma = 0.98$) in 2010–2011 and 5.0 ($\sigma = 1.50$) in 2012–2013. Total number of BSPs decreases from 83 in 2010–2011 to 14 in 2012–2013 and the number of PCCs decreases from 72 to 32. The number of BSPs and PCCs are normalized *per resident* unit for a statistical test.

	2010-2011 mean (σ)	**2012-2013 mean (σ)**
Residents / month	15.7 (2.06)	11.2 (1.18)
Total Shifts / resident	15.8 (1.68)	16.4 (1.62)
Night Shifts / resident	4.7 (0.98)	5.0 (1.50)
Total BSP / year	83	14
Total PCC / year	72	32

Academic Years 2010–2011 vs. 2012–2013: Number of residents per month, total shifts per resident, night shifts per resident, BSPs per year, and PCCs per year.

Table 2.3: Number of residents per month and average occurrence for the metrics

The BSPs decreased by 85.7% from 2010–2011 to 2012–2013 (mean change, 0.54 BSPs / resident per month; $P < 0.001$; 95% CI 0.31 to 0.77). The number of PCCs decreased by 66.7% (mean change, 0.36 PCCs / resident per month; $P < 0.002$; 95% CI 0.16 to 0.56). While there was no significant difference in total shift disparity between years (percentage change, 25%; $P = 0.491$; 95% CI -0.02 to 0.04), there was statistically significant reduction in night shift disparity, which decreased by 55.6% ($P < 0.001$; 95% CI 0.02 to 0.05).

	2010-2011	**2012-2013**	**Difference (Resident per Month)**		
	Resident per Month, mean (σ)		**Mean Difference**	**95% confidence interval (CI)**	**P Value**
BSP	0.6312 (0.2821)	0.0897 (0.3109)	0.5415	(0.3110, 0.7720)	< 0.001
PCC	0.5388 (0.26)	0.1757 (0.22)	0.3631	(0.1655, 0.5608)	0.002
Total Shift Variability	0.08 (0.02)	0.06 (0.03)	0.0089	(-0.0189, 0.0367)	0.491
Night Shift Variability	0.09 (0.03)	0.04 (0.02)	0.0421	(0.02652, 0.05763)	< 0.001

Table 2.4: Normalized quality of metrics by monthly residents

Finally, we note that by coordinating the building of the schedule around both a mathematical model and an associated interactive process, we have improved the continuity from year to year as the Chief residents transition. We also see a greater percetion of fairness and acceptance of the schedule by the residents overall.

2.6 Conclusions

In this paper, we present an iterative and interactive IP-based approach that incorporates complicated rules and qualitative preference of residents with high flexibility. This approach provided faster and higher quality of schedules to the chief residents than the previous manual scheduling process. This also contributes to improving patient safety, and resident satisfaction, and morale for educational training.

CHAPTER 3

Pareto Shift Scheduling Problem

3.1 Motivation

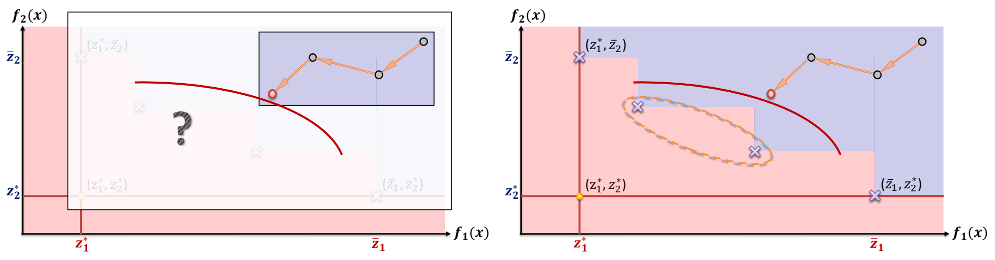
Residency scheduling problems are often challenging to solve because of a large number of conflicting rules and requirements needed to ensure both adequate patient care and resident educational opportunities. Given the combinatorial complexity of residency scheduling problems, optimization-based decision support tools seem appropriate for aiding the Chief Residents in their scheduling task, but additional complexity is introduced by the fact that there is no one clear objective function. For example, from a patient's perspective, we might want to encourage schedules where only senior residents staff overnight shifts, when there is less supervision from attending physicians. On the other hand, from a training perspective, we might want to encourage schedules where less experienced residents gain from the types of training experiences most likely to occur during the overnight shifts. Thus, many different metrics (associated with patient care, resident education, and resident satisfaction) are of value to the Chiefs in selecting a final schedule.

One common approach for multi-criteria objective problems is the weighted sum methods to establish a single objective function [63]. However, in our experience, this has been unsuccessful for residency scheduling for two key reasons. First, it can be very difficult for the Chief Residents to accurately quantify their preferences. Second, these preferences change on a regular basis – for example, focusing on training may be more important in early months of the academic year because the residents are new to the program, while later in the year it may be more important to focus on residents' day-off requests because of fellowship interviews. Thus, even if methods were developed to intuit the Chief Residents' preferential weights from past planning periods, these weights may not be appropriate for future planning periods.

Another approach (as seen in [5, 23, 45, 48, 49]) is to utilize the analytic hierarchy process (AHP) [100], in which users are surveyed to provide relative preferences between pairs of metrics. However, we again find this inappropriate for the pediatric resident scheduling problem for the following reasons: First, because the preferences change over time, this surveying process would have to be repeated monthly. Second, the pairwise comparison questionnaire surveys in AHP are too simplistic to capture complex conditional dependencies. As a simple example, as suggested by [101, 102], I might prefer red wine to white wine in general, but prefer white wine if eating fish. Finally, because AHP relies on an underlying weighted sum approach, it may fail to find all Pareto dominant solutions within an MIP formulation [55, 103].

On the other hand, in interactive methods, the Chief Residents are actively involved in the solution process. They investigate the presented schedule and express their preference information. The preference is subsequently used to generate a new feasible schedule, and we provide it to the Chief residents. Thus, the Chief Residents progressively refine their preferences until they are satisfied with the final solution and do not wish to continue further. However, as shown in Figure 3.1, it tends to select a myopic solution without knowing better possible alternatives since they are satisficers, not optimizers; they are not likely to spend extra time to find a better schedule if they feel current one is good enough.

Therefore, as an alternative, we propose not to generate a single "optimal" schedule (in fact, optimality is not defined) but rather to provide the Chiefs with the complete set of Pareto dominant schedules, from which they can choose. A solution is Pareto dominant if there is no other feasible solution that is as good, or better, on all metrics. If one solution Pareto-dominates another, then the latter does not need to be considered by the Chiefs. Apparently, instead of evaluating the potentially large set of feasible solutions, we can assist the Chief Resident to focus on only Pareto frontier, the set of all Pareto schedules. In addition, since Pareto frontier gives the whole picture of



Suppose that the red line is satisfaction line. Then, the Chief Residents could stop searching further if the search reaches the satisfaction line since they are happy. However, there could exist Pareto-dominant schedules that dominate the choice since they never see the whole picture of possible alternatives.

Figure 3.1: Interactive approach vs. Pareto-dominant approach

alternatives to the decision maker, the decision maker can have high confidence in the final decision by comparing various Pareto schedules before the final choice. To this end, we have developed an integer programming-based approach embedded within a recursive algorithm to generate all Pareto-dominant solutions.

Some papers [104, 105] suggest a Pareto-based heuristic method based on simulated annealing, which is a stochastic search algorithm first introduced by Kirkpatrick et al. [86]. One of the earliest papers to use Pareto-based optimization for healthcare personnel schedule is Jaszkiewicz [104]. It introduced a Pareto-based simulated annealing approach to solving the NSP by using a weighted-sum method with adaptively changing weights. Later, another paper [105] extended the Pareto-based simulated annealing approach to solving the multi-objective NSP with two options to address user preferences in different ways. They show that the Pareto-based approach provides high-quality solutions and flexible decision systems that are capable of handling changing environments which are not possible to precisely capture when using constant weights. However, their methods are based on an approximated Pareto frontier instead of the whole frontier. We introduce a way to provide the Chiefs with the complete set of Pareto dominant solutions.

3.2 PEDS Simplified Feasiblity Formulation

In this paper, we consider the problem of building monthly schedules for medical residents who provide shift coverage in a pediatric emergency department. In particular, given the multi-criteria nature of the problem, we focus on generating Pareto-dominant schedules from which the scheduler (typically, a Chief Resident) can choose. For the sake of exposition, we focus on the following rules, which are amongst the most common and most important in residency shift scheduling. For a more complex and nuanced model, see the mathematical formulation in Chapter 2.

1. Required Shift Coverage: Each of the six required (i.e., non-flex) shifts must be covered by exactly one resident for each day in the planning horizon.

2. Flex Shift Coverage: The flex shift can be assigned to *at most* one resident for each day in the planning horizon.

3. Total Shifts: There is a lower bound and an upper bound on the total number of shifts that can be assigned to each resident.

4. Night Shifts: There is a lower bound and an upper bound on the total number of night shifts that can be assigned to each resident.

5. Working Days in a row: There is an upper bound on the allowable number of consecutive days (any shift) that can be worked in a row.

6. Working Nights in a row: There is an upper bound on the allowable number of consecutive days on which an overnight shift (8 PM – 5 AM, 11 PM – 8 AM) can be worked.

7. Intern Limitations: First-year residents (*interns*) cannot be assigned to the 7 AM – 4 PM nor the 11 PM – 8 AM shifts.

8. Program Limitation: Residents in certain programs cannot work specific days of the week due to other program requirements.

9. Ten Hours Rest: There must be at least ten hours between the end of one shift and the start of the next shift for each resident.

10. Continuity Clinic: If a resident works a *continuity clinic* (a standing year-long assignment to a weekly community clinic), then they cannot work certain shifts before, during, and after their continuity clinic (5 PM – 2AM, 8 PM – 5 AM, 11 PM – 8 AM before the clinic day and 7 AM – 4 PM, 9 AM – 6 PM, 12 PM – 9 PM, 4 PM – 1 AM, 5 PM – 2 AM on the clinic day)

In this paper, to assess the quality of a resident schedule for the pediatric ED, we consider the following four metrics. Note that all of these metrics are defined over the set of non-negative integers value.

- Number of *Bad Sleep Patterns* (BSPs) are sequences of consecutive shift assignments that disrupt residents' circadian rhythm (see Table 2 in [99]). For example, it is legal to work a Monday morning shift, a Tuesday overnight shift, and a Thursday morning shift, but doing so is highly disruptive to a resident's sleep patterns and thus undesirable.

- Number of *Post-Continuity Clinic* shifts (PCCs) are shifts in the ED assigned to a resident shortly after completion of a continuity clinic. Although it is legal from the perspective of duty-hour requirements, it yields a very long work stretch. In addition, performing both tasks typically requires leaving clinic early or arriving late to the ED.

- Number of *Denied Vacation Requests* (DVRs) are periods of time for which the resident requested time off but was nonetheless assigned to work at least one shift during this time period.

- Number of *Uncovered Flex Shifts* (UFSs) refer to noon shifts ("flex shifts") that go unassigned. While this is permissible, it is not desirable.

The following model defines the feasible set of resident shift schedules, relative to the rules outlined above.

3.2.1 Notation

R	set of residents, $r \in \{1, 2, \cdots, \mid R \mid\}$
T	set of training programs, $t \in \{$EMS, PED, EM, FM, $\cdots\}$
t_r	training program of resident r, $t_r \in T \; \forall r \in R$
L	set of seniority levels, $l \in \{$interns, seniors$\}$
l_r	seniority level of resident r, $l_r \in L \; \forall r \in R$
D	set of days, $d \in \{1, 2, \cdots, \mid D \mid\}$; In our data instances, D is either 30 or 31
C_r	set of days on which resident r has continuity clinic, $C_r \subseteq D \; \forall r \in R$
S	set of shifts, $s \in \{1, 2, \cdots, \mid S \mid\}$; In our data instances, there are 7 shifts
F	set of flex shifts, $F \subset S$; In our data instances, shift 12 PM – 9 PM is the only element of F
N	set of night shifts, $N \subset S$; In our data instances, shifts 8 PM – 5 AM and 11 PM – 8 AM are in N
I	set of intern-prohibited shifts, $I \subset S$; In our data instances, shifts 7 AM – 4 PM and 11 PM – 8 AM are in I
P	set of shifts that are defined as the post-continuity clinic shifts, $P \subset S$; In our data instances shifts 8 PM – 5 AM and 11 PM – 8 AM are in P
O_d	set of day-shift pairs that conflict with continuity clinic on day d, $O_d \subset D \times S$ $\forall d \in D$
H_t	set of day-shift pairs that cannot be worked by residents in training program t, $H_t \subset D \times S \; \forall t \in T$
$J_{(d,s)}$	set of day-shift pairs that would cause a duty-hours violation if assigned in addition to day-shift (d, s), $J_{ds} \subset D \times S \; \forall (d, s) \in D \times S$. Note that (d, s) is included in the set $J_{(d,s)}$
U	set of bad (undesirable) sleep patterns where $u \in U$ is a combination of shift offsets on multiple days. Note that $\mid u \mid$ represents the number of shift offsets in u.

$U_{(d,u)}$	set of date-shift pairs associated with bad sleep pattern u on day d, $U_{(d,u)} \subset D \times S$ $\forall (d,u) \in D \times U$
V	set of vacation requests where $V_r \in V$ is a set of day-shift pairs $(d,s) \in D \times S$ that resident r requests as part of a day off, $V_r \subset D \times S \; \forall r \in R$
$\overline{D}$	maximum allowable number of days in a row that a resident can work a shift
$\overline{N}$	maximum allowable number of days in a row that a resident can work a night shift
$\underline{S}_r$, $\overline{S}_r$	lower and upper bounds on the number of total shifts for resident r, $\forall r \in R$
$\underline{N}_r$, $\overline{N}_r$	lower and upper bounds on the number of night shifts for resident r, $\forall r \in R$

3.2.2 Variables

x_{rds}	binary variable, equals 1 if resident r is assigned to shift s on day d; otherwise 0
y_{rdu}	binary variable, equals 1 if resident r is assigned to bad sleep pattern u on day d; otherwise 0
z_{rd}	binary variable, equals 1 if resident $r \in R$ is assigned to work a post-continuity clinic shift on day $d \in C_r$; otherwise 0
q_v	binary variable, equals 1 if vacation request $v \in V$ is denied; otherwise 0

3.2.3 Formulation

$$\min \quad 0 \tag{3.1}$$

$$\text{subject to: } \sum_{r \in R} x_{rds} = 1 \qquad \forall s \in S \backslash F, \forall d \in D \tag{3.2}$$

$$\sum_{r \in R} x_{rds} \leq 1 \qquad \forall s \in F, \forall d \in D \tag{3.3}$$

$$\sum_{d \in D} \sum_{s \in I} x_{rds} = 0 \qquad \forall r \in \{r : l_r = \text{interns}\} \tag{3.4}$$

$$\sum_{r \in \{r : t_r = t\}} \sum_{(d,s) \in H_t} x_{rds} = 0 \qquad \forall t \in T \tag{3.5}$$

$$\sum_{d \in C_r} \sum_{(d,s) \in O_d} x_{rds} = 0 \qquad \forall r \in R \tag{3.6}$$

$$\sum_{(d,s) \in J_{(d,s)}} x_{rds} \leq 1 \qquad \forall r \in R, \forall d \in D, \forall s \in S \tag{3.7}$$

$$\sum_{i=d}^{d+\overline{D}} \sum_{s \in S} x_{ris} \leq \overline{D} \qquad \forall r \in R, \forall d \in D \tag{3.8}$$

$$\sum_{i=d}^{d+\overline{N}} \sum_{s \in N} x_{ris} \leq \overline{N} \qquad \forall r \in R, \forall d \in D \tag{3.9}$$

$$\underline{S}_r \leq \sum_{d \in D} \sum_{s \in S} x_{rds} \leq \overline{S}_r \qquad \forall r \in R \tag{3.10}$$

$$\underline{N}_r \leq \sum_{d \in D} \sum_{s \in N} x_{rds} \leq \overline{N}_r \qquad \forall r \in R \tag{3.11}$$

$$y_{rds} \leq x_{rij} \qquad \forall r \in R, \forall d \in D, \forall u \in U, \forall (i,j) \in U_{(d,u)} \tag{3.12}$$

$$y_{rds} + \mid u \mid \geq \sum_{(d,s) \in U_{(d,u)}} x_{rds} + 1 \qquad \forall r \in R, \forall d \in D, \forall u \in U \tag{3.13}$$

$$z_{rd} \geq x_{rds} \qquad \forall r \in R, \forall d \in C_r, \forall s \in P \tag{3.14}$$

$$z_{rd} \leq \sum_{s \in P} x_{rds} \qquad \forall r \in R, \forall d \in C_r \tag{3.15}$$

$$q_v \geq x_{rds} \qquad \forall v = (r, V_r) \in V, \forall (d, s) \in V_r \tag{3.16}$$

$$q_v \leq \sum_{(d,s) \in V_r} x_{rds} \qquad \forall v = (r, V_r) \in V \tag{3.17}$$

$$x_{rds} \in \{0, 1\} \qquad \forall r \in R, \forall d \in D, \forall s \in S \tag{3.18}$$

$$y_{rdu} \in \{0, 1\} \qquad \forall r \in R, \forall d \in D, \forall u \in U \tag{3.19}$$

$$z_{rd} \in \{0, 1\} \qquad \forall r \in R, \forall d \in C_r \tag{3.20}$$

$$q_v \in \{0, 1\} \qquad \forall v \in V \tag{3.21}$$

Decision variables (3.18) indicate whether to assign resident r to shift s on day d or not. Auxiliary variables (A.19) indicate whether to assign resident r to bad sleep pattern u on day d, (A.20) indicate whether to assign resident r to a post-continuity clinic shift on day d, (3.21) indicates whether the scheduler denies vacation request v in V.

Constraints (3.2) ensure that exactly one resident is assigned to every non-flex shift. Constraints (3.3) ensure that flex shifts are covered by at most one resident. Constraints (3.4) ensure that interns are not assigned shifts that must be covered by seniors. Constraints (3.5) ensure that residents are not assigned to shifts that conflicts with their program requirements. Constraints (3.6) ensure that residents are not assigned to shifts that conflicts with their continuity clinics. Constraints (3.7) ensure that residents are not assigned to shifts that begin within ten hours of a previously worked shift. Constraints (3.8) ensure that no resident works more than a maximum allowable number of days in a row. Constraints (3.9) ensure that no resident works more than a maximum allowable number of nights in a row. Constraints (3.10) and (3.11) ensure that each resident is assigned to a number of shifts and night shifts between their respective lower and upper bounds.

Constraints (A.1) and (A.2) link decision variables x_{rds} and auxiliary variables y_{rds} to count the number of bad sleep patterns. Constraints (A.3) and (A.4) link decision variables x_{rds} and auxiliary variables z_{rd} to count the number of post-continuity clinic shifts. Constraints (3.16) and (3.17) link decision variables x_{rds} and auxiliary variables q_v to count the number of denied vacation requests.

3.3 Finding the Pareto Frontier of PEDS metrics

In Section 3.2, we presented an IP formulation for finding feasible solutions to the PEDS problem. We denote by $\mathcal{S}$ the set of all feasible schedules. These solutions can be evaluated under several different metrics (i.e., number of BSPs, PCCs, DVRs, UFSs). In this section, we discuss a method that can be used to find all *Pareto dominant points* associated with this feasible region. In general, this method is applicable to any multi-objective problem with integer-valued metrics.

Notation and Sets

n	total number of metrics
$\mathcal{S}$	solution (or decision) space, i.e., the set of feasible schedules
s	schedule, i.e., an element in the solution space, $s \in \mathcal{S}$
$f_i(s)$	value of the ith metric for schedule s for all $s \in \mathcal{S}$ and $i \in \{1, \cdots, n\}$
$f(s)$	metric point, i.e., the vector of metric values for schedule s; $f(s) = (f_1(s), \cdots, f_n(s))$
$\mathcal{M}$	metrics (or objective) space, i.e., the set of metric points associated with all feasible schedules; $\mathcal{M} = \{f(s) \mid s \in \mathcal{S}\}$

Without loss of generality, we assume that we want minimal values for each of the metrics. Note that there may be multiple schedules that all map to the same metric point. We will therefore sometimes use the notation m for an element in $\mathcal{M}$, because m does not necessarily correspond to a unique schedule. In details, every feasible schedule evaluated by n metrics $\{f_1, \cdots, f_n\}$ should map to some metric point m in $\mathcal{M}$ although multiple different feasible schedules could map to the same one.

Key definitions for a MOP are dominance and Pareto frontier. We introduce the following key definitions to understand the concept of Pareto optimality.

Definition 3.3.1. *Dominance ($\prec$): Given schedules s and $s' \in \mathcal{S}$, we say that s dominates s'*

($s \prec s'$) if and only if $f_i(s) \leq f_i(s')$ for all $i \in \{1, \cdots, n\}$ and $\exists i \in \{1, \cdots, n\}$ such that $f_i(s) < f_i(s')$. Similarly, given metric points m and $m' \in \mathcal{M}$, we say that m dominates m' ($m \prec m'$) if and only if all schedules associated with m dominate all schedules associated with m'.

Definition 3.3.2. *Strictly Dominance($\lll$): Let schedules s and $s' \in \mathcal{S}$. We say that s strictly dominates s' ($s \lll s'$) if and only if $f_i(s) < f_i(s')$ for all $i \in \{1, \cdots, n\}$. Similarly, given two metric point m and $m' \in \mathcal{M}$, we say that m strictly dominates m' ($m \lll m'$) if and only if a schedule associated with m strictly dominates a schedule associated with m'.*

Definition 3.3.3. *Pareto point ($\hat{m}$): The metric point $m \in \mathcal{M}$ is a Pareto point if and only if there is no other metric point m' in $\mathcal{M}$ such that $m' \prec m$.*

Definition 3.3.4. *Pareto frontier ($\mathcal{P}$): set of all Pareto points, $\mathcal{P} = \{m \in \mathcal{M} \mid \nexists m' \in \mathcal{M} : m' \prec m\}$.*

Definition 3.3.5. *ideal value(f_i^*): the best value of ith metric value over Pareto frontier, $f_i^* = \min\{f_i(s) : f(s) \in \mathcal{P}\}$.*

Definition 3.3.6. *nadir value($\bar{f}_i$): the worst value of ith metric value over Pareto frontier, $\bar{f}_i = \max\{f_i(s) : f(s) \in \mathcal{P}\}$.*

Definition 3.3.7. *ideal point(p^*): a vector of ideal values, $f^* = (f_1^*, \cdots, f_n^*)$.*

Definition 3.3.8. *nadir point($\bar{p}$): a vector of nadir values, $\bar{f} = (\bar{f}_1, \cdots, \bar{f}_n)$.*

Given a feasible region $\mathcal{S}$ and a vector of metrics $\{f_1, \cdots, f_n\}$, we denote the MOP of finding the associated Pareto frontier $\mathcal{P}$ by $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$. Our approach focuses specifically on the case where the metrics only yield integer values. We define the MOP with n integer metrics $\{f_1, \cdots, f_n\}$ over feasible set $\mathcal{S}$ as follows:

$$\min \quad f(s) = (f_1(s), f_2(s), \cdots, f_n(s)) \tag{3.22}$$

$$\text{subject to: } s \in \mathcal{S} \tag{3.23}$$

$$f_i(s) \in \mathbb{Z} \quad \forall i \in \{1, \cdots, n\} \tag{3.24}$$

In this paper, we present an exact algorithm to find all points on the Pareto frontier $\mathcal{P}$ of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$. For conventional notation, we denote the set $\{s : f(s) \leq u\}$ in a subscript by $\{f(s) \leq u\}$ or $\{f \leq u\}$ if there is no confusion. We also define the weighted sum single-objective problem with additional upper bound constraints on all metrics as follows:

$$F_{\mathcal{S}}(\{(f_1, w_1, u_1), \cdots, (f_n, w_n, u_n)\}) = \min \quad w_1 f_1(s) + \cdots + w_n f_n(s) \tag{3.25}$$

$$\text{subject to: } s \in \mathcal{S} \tag{3.26}$$

$$f_1(s) \leq u_1 \tag{3.27}$$

$$\vdots \tag{3.28}$$

$$f_n(s) \leq u_n \tag{3.29}$$

where w_i represents a weight and u_i represents an upper bound on metric i, respectively.

Finally, we define the following lexicographic optimization problem $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ for a feasible region $\mathcal{S}$ and a vector of metrics $\{f_1, \cdots, f_n\}$. In the lexicographic optimization problem $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$, we assume that the metrics are ranked in the order of importance so that f_1 is more important than all other metrics, and f_n is the least important metric. Then, we solve a sequence of single-objective problems for each metric in order. First, it starts by solving $\mathbb{P}_{\mathcal{S}}(\{f_1\})$ to minimize the first metric f_1 over the feasible set $\mathcal{S}$. Once we get the optimal value $\hat{z}_1$ of f_1 from $\mathbb{P}_{\mathcal{S}}(\{f_1\})$, we solve $\mathbb{P}_{\mathcal{S} \cap \{f_1(s) \leq \hat{z}_1\}}(\{f_2\})$ to minimize the second metric f_2 over the feasible set $\mathcal{S}$ to which the new upper bound (i.e., $f_1(s) \leq \hat{z}_1$) on f_1 is added. Again, once we get the optimal value $\hat{z}_2$ of f_2 from $\mathbb{P}_{\mathcal{S} \cap \{f_1(s) \leq \hat{z}_1\}}(\{f_2\})$, we solve $\mathbb{P}_{\mathcal{S} \cap \{f_1(s) \leq \hat{z}_1, f_2(s) \leq \hat{z}_2\}}(\{f_3\})$ to minimize the third metric f_3 over the feasible set $\mathcal{S} \cap \{s : f_1(s) \leq \hat{z}_1\}$ with new upper bound (i.e., $f_2(s) \leq \hat{z}_2$) on f_2 and so on. At the ith iteration, we solve $\mathbb{P}_{\mathcal{S}\cap\{f_1 \leq \hat{z}_1, \cdots, f_{i-1} \leq \hat{z}_{i-1}\}}(\{f_i\})$ by minimizing f_i over the feasible set $\mathcal{S}$ with a upper bounds $f_j(s) \leq \hat{z}_j$ for all $j < i$ where $\hat{z}_j$ is the optimal value of previous problems above successively. We describe the details of $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ in Algorithm 1.

Algorithm 1: $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$, lexicographic optimization problem for $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$

```
   INPUT   : $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$
1  STEP 1 : optimize $f_1$
2      Solve $\mathbb{P}_{\mathcal{S}}(\{f_1\})$
3      Set $\hat{z}_1 = F_{\mathcal{S}}(\{(f_1, 1, \infty), (f_2, 0, \infty), \cdots, (f_n, 0, \infty)\})$
4  STEP 2 : optimize $f_2$
5      Solve $\mathbb{P}_{\mathcal{S}\cap\{f_1 \leq \hat{z}_1\}}(\{f_2\})$
6      Set $\hat{z}_2 = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), (f_2, 1, \infty), (f_3, 0, \infty), \cdots, (f_n, 0, \infty)\})$
7      ⋮
8  STEP n : optimize $f_n$
9      Solve $\mathbb{P}_{\mathcal{S}\cap\{f_1 \leq \hat{z}_1, \cdots, f_{n-1} \leq \hat{z}_{n-1}\}}(\{f_n\})$
10     Set $\hat{z}_n = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), (f_2, 0, \hat{z}_2), \cdots, (f_{n-1}, 0, \hat{z}_{n-1}), (f_n, 1, \infty)\})$
   OUTPUT  $(\hat{z}_1, \hat{z}_2, \cdots, \hat{z}_n)$
   :
```

By Definition 3.3.1 and 3.3.3, we prove the following theorem:

Theorem 3.3.1. *The metric point $(\hat{z}_1, \cdots, \hat{z}_n)$ gained by solving $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ is a Pareto point for $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$.*

Proof. Suppose that there exists some metric point $\tilde{m} = (\tilde{z}_1, \cdots, \tilde{z}_n)$ that dominates the metric point $\hat{m} = (\hat{z}_1, \cdots, \hat{z}_n)$ gained by solving $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$. Then, by Definition 3.3.1, there is the first index $i \in \{1, \ldots, n\}$ such that $\tilde{z}_i < \hat{z}_i$ and $\tilde{z}_j = \hat{z}_j$ for all $j < i$. On the other hand, we get $\hat{z}_i$ by solving $F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), \cdots, (f_{i-1}, 0, \hat{z}_{i-1}), (f_i, 1, \infty), (f_{i+1}, 0, \infty), \cdots, (f_n, 0, \infty)\})$ in Algorithm 1. It implies that no metric point has $f_i < \hat{z}_i$ when $f_j = \hat{z}_j$ for all $j < i$. It contradicts the assumption that there exists the first index i of the metric point $\tilde{m}$. Thus, no metric point dominates $\hat{m}$. By Definition 3.3.3, it implies that $\hat{m}$ is a Pareto point. □

3.3.1 Literature Review: Exact Algorithm

We define *multi-objective problem with integer values (MOPI)* to be any optimization problem with multiple objective functions, where these objective functions always yield integer values. We consider the problem of finding all points on the Pareto frontier for a given MOPI. We begin by reviewing the literature on a subset of MOPI, the multi-objective integer programming (MOIP), where the optimization problem is specifically an integer program. Klein and Hannan [106] suggest an approach for the n dimensional MOIP based on a sequence of progressively more constrained single-objective optimization problems, eliminating the known dominated solutions; improvements of this appear in [103, 107]. Approaches for improving an upper bound on the worst-case computational time include the *full p-split method* [108, 109, 110] and the *full $(p-1)$-split method* [107, 111, 112]. Recently, Boland proposed new two variants of the *full $(p-1)$-split method*, called the *L-shaped search method* [113], and the *quadrant shrinking method*, for the tri-objecive integer programming (TOIP) problem [114].

In this paper, we use the recursive method [115, 116], which is a generalization of the well-known ϵ-constraint method [56] and is known as one of the fastest existing methods [114].

The ϵ-constraint method was first introduced by Haimes et al. [56]. The basic idea of this approach is to optimize a single objective while all other objectives are treated as constraints with upper bounds. The optimal solution of the constrained single objective problem is then used to determine new upper bounds on the objectives in the next iteration, and this is repeated until there is no new solution found. Berube [117] solves bi-objective combinatorial optimization problems by using the ϵ-constraint method. However, the original ϵ-constraint method could find a weakly Pareto point (i.e., we say a metric point m is a weakly Pareto point when there is no other metric point $m' \lll m$, but there exists a metric point $m'' \prec m$). Mavrotas [55] suggests augmented ϵ-constraint method by introducing slack or surplus variables, and Kirlik and Sayın [112] use two-stage ϵ-constraint formulation to guarantee to find Pareto points only by modifying the original ϵ-constraint method. Recently, Özlen and Azizoğlu [115] suggest a recursive method, which is an extension of the ϵ-constraint method for two-dimensional metrics, for n-dimensional objectives problem and later improves its computational performance by using a relaxation technique [116].

3.3.2 Recursive method for MOPI

The recursive algorithm proposed by Özlen and Azizoğlu [115] is to find the Pareto frontier of the MOIP problem by recursively identifying Pareto frontier of subproblems with fewer objectives. The key idea of recursive method is the inductive relationship between the nadir value of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ and the Pareto frontier of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_{n-1}\})$. To simplify notations in this section, we denote the vector of metric values as z and the value of the ith metric value as z_i.

3.3.2.1 Bi-objecive Problem with Integer Values (BOPI)

As preparation for the treatment of a high dimensional multi-objective problem, this section is devoted to a two dimensional multi-objective problem, $\mathbb{P}_{\mathcal{S}}(\{f_1, f_2\})$. First, we assume that the decision maker has two objectives in mind. We introduce a well-known ϵ-constraint method for solving bi-objective problem with integer values (BOPI). This simple case can give an example to illustrate the basic idea of the recursive method graphically. We will define the BOPI as follows:

$$\min(f_1(s), f_2(s)) \tag{3.30}$$

$$\text{subject to: } s \in \mathcal{S} \tag{3.31}$$

$$f_i(s) \in \mathbb{Z} \quad \forall i \in \{1, 2\} \tag{3.32}$$

where $\mathcal{S}$ refers to set of all feasible schedules and parameter $\{f_1, f_2\}$ represents set of metrics we will consider as objective functions.

(a) Range of Metric Space for BOPI To find the region in objective space that contains the Pareto frontier, we define the following two points that define lower and upper bounds on metric values of the Pareto frontier. The ideal value of f_1 is $F_{\mathcal{S}}(\{(f_1, 1, \infty), (f_2, 0, \infty)\})$ and denote it by z_1^*. Similarly, the ideal value of f_2 is $z_2^* = F_{\mathcal{S}}(\{(f_1, 0, \infty), (f_2, 1, \infty)\})$. Then, $z^* = (z_1^*, z_2^*)$ is the ideal point that defines lower bounds on metric values of all Pareto points. As an upper bound of each metric value, nadir values of f_1 and f_2 are $\bar{z}_1 = F_{\mathcal{S}}(\{(f_1, 1, \infty), (f_2, 0, z_2^*)\})$ and $\bar{z}_2 = F_{\mathcal{S}}(\{(f_1, 0, z_1^*), (f_2, 1, \infty)\})$ respectively. Then, $\bar{z} = (\bar{z}_1, \bar{z}_2)$ is the nadir point that defines upper bounds on metric values of all Pareto points. By definition of an ideal and a nadir value, it follows that $z_1^* \leq \hat{z}_1 \leq \bar{z}_1$ and $z_2^* \leq \hat{z}_2 \leq \bar{z}_2$ for all Pareto points $(\hat{z}_1, \hat{z}_2)$ in $\mathcal{P}$.

Corollary 3.3.1.1. *The corner points $(z_1^*, \bar{z}_2)$ and $(\bar{z}_1, z_2^*)$ are in $\mathcal{P}$.*

Proof. By Definition 3.3.5 and 3.3.6, we can know that $z_1^* = F_{\mathcal{S}}(\{(f_1, 1, \infty), (f_2, 0, \infty)\})$ and $\bar{z}_2 = F_{\mathcal{S}}(\{(f_1, 0, z_1^*), (f_2, 1, \infty)\})$. Obviously, by Algorithm 1, we can get $(z_1^*, \bar{z}_2)$ by solving $\mathbb{H}_{\mathcal{S}}(\{f_1, f_2\})$. Similarily, we can get $(\bar{z}_1, z_2^*)$ from $\mathbb{H}_{\mathcal{S}}(\{f_2, f_1\})$ by switching a role of f_1 and f_2. Therefore, by Thoerem 3.3.1, two corner points $(z_1^*, \bar{z}_2)$ and $(\bar{z}_1, z_2^*)$ are in $\mathcal{P}$. □

Theorem 3.3.2. *For all pareto points $(\hat{z}_1, \hat{z}_2)$ in $\mathcal{P}$, it follows that $z_1^* \leq \hat{z}_1 \leq \bar{z}_1$ and $z_2^* \leq \hat{z}_2 \leq \bar{z}_2$.*

Proof. Suppose that there is a Pareto point $(\hat{z}_1, \hat{z}_2)$ such that $\hat{z}_1 < z_1^*$ or $\hat{z}_1 > \bar{z}_1$. Then, it contradicts Definition 3.3.5 or 3.3.6. Therefore, if a point $(\hat{z}_1, \hat{z}_2)$ in $\mathcal{P}$, it should follow $z_1^* \leq \hat{z}_1 \leq \bar{z}_1$. Similarily, it holds for $\hat{z}_2$. □

(b) Exact ϵ-constraint method for BOPI Berube [117] suggests the exact ϵ-constraint method to finding the Pareto frontier of the bi-objective optimization problem with integer objective values through a sequence of constrained problems based on a progressive tightening an upper bound u_2 on f_2. Briefly, throughout the algorithm, u_2 is improved (i.e., decreased when it minimizes the objective) monotonically and find new Pareto point that has a worse value of f_1 as a tradeoff of tighter upper bound on f_2. Since there are several variations of improvement [55, 112, 117] from the conventional ϵ-constraint method [56], we restate an original version of them by using $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ in Algorithm 1 in order to keep the consistency of notations with our sequential method for general n-objective problem in Section 3.3.3.

```
Algorithm 2: $\epsilon$-constraint method for BOPI, $\mathbb{P}_{\mathcal{S}}(\{f_1, f_2\})$
   INPUT  : $\mathbb{P}_{\mathcal{S}}(\{f_1, f_2\})$
   INIT: $k = 1, \mathcal{P} = \{\emptyset\}, u_2 = \infty$
 1 WHILE $\mathcal{S} \cap \{s : f_2(s) \leq u_2\} \neq \emptyset$ DO
 2     Let $\mathcal{U}_2 = \{\emptyset\}$
 3     BEGIN Lexicographic Optimization
 4         Solve $\mathbb{H}_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{f_1, f_2\})$
 5         Set $(\hat{z}_1^k, \hat{z}_2^k)$ be the solution
 6         Update $\mathcal{P} = \mathcal{P} \cup (\hat{z}_1^k, \hat{z}_2^k)$ and $k = k + 1$
 7         Set $\mathcal{U}_2 = \mathcal{U}_2 \cup \{\hat{z}_2^k\}$
 8     END
 9     Set $\bar{z}_2 = \max\{\mathcal{U}_2\}$ and $u_2 = \bar{z}_2 - 1$
10 END
   OUTPUT  $\mathcal{P}$
   :
```

In Algorithm 2, we first optimize for f_1 metric with an upper bound constraint on f_2 (i.e., $f_2 \leq u_2$). Denote the optimal value of f_1 by $\hat{z}_1$. Second, we optimize for f_2 relative to the optimal metric value of f_1 and denote it by $\hat{z}_2$. Note that these two steps are equal to the lexicographic optimization problem $\mathbb{H}_{\mathcal{S}}(\{f_1, f_2\})$ for $\mathbb{P}_{\mathcal{S}}(\{f_1, f_2\})$. We have proved above that the point $(\hat{z}_1, \hat{z}_2)$ gained by solving $\mathbb{H}_{\mathcal{S}}(\{f_1, f_2\})$ is a Pareto point in $\mathcal{P}$. Finally, we tighten an upper bound u_2 of f_2 by subtracting 1 from $\hat{z}_2$ and repeat this process to find the other Pareto point until u_2 is less than z_2^*. Before showing that Algorithm 2 finds all Pareto points without missing anything, we first state two following theorems.

Theorem 3.3.3. *Given the metric point $(\hat{z}_1, \hat{z}_2)$ gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{f_1, f_2\})$ where $\hat{z}_2 \leq u_2$, there is no other Pareto point that has $f_2 > \hat{z}_2$ for $\mathbb{P}_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{f_1, f_2\})$.*

Proof. By Algorithm 1, we get $\hat{z}_1$ and $\hat{z}_2$ respectively by solving $\hat{z}_1 = F_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{(f_1, 1, \infty), (f_2, 0, \infty)\})$ and $\hat{z}_2 = F_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{(f_1, 0, \hat{z}_1), (f_2, 1, \infty)\})$. Note that, for $\mathbb{P}_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{f_1, f_2\})$, the ideal value of f_1 is $z_1^* = F_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{(f_1, 1, \infty), (f_2, 0, \infty)\})$ and the nadir value of f_2 is $\bar{z}_2 = F_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{(f_1, 0, z_1^*), (f_2, 1, \infty)\})$. Since $z_1^* = \hat{z}_1$, the nadir value of f_2 for $\mathbb{P}_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{f_1, f_2\})$ is $\hat{z}_2$. By Definition 3.3.6, there is no Pareto point that has $f_2 > \hat{z}_2$ for $\mathbb{P}_{\mathcal{S}\cap\{f_2 \leq u_2\}}(\{f_1, f_2\})$. □

Theorem 3.3.4. *For each $k \in \{1, \cdots, K\}$ where K is the total number of iterations in Algorithm 2, $(\hat{z}_1^k, \hat{z}_2^k)$ is a unique Pareto point having $f_2 = \hat{z}_2^k$.*

Proof. For some $k \in \{1, \cdots, K\}$, suppose that there is another distinct Pareto point $(\tilde{z}_1, \tilde{z}_2)$ that has $f_2 = \hat{z}_2^k$ (i.e., $\tilde{z}_2 = \hat{z}_2^k$). To be distinct Pareto point, $\tilde{z}_1$ should be less than $\hat{z}_1^k$. Therefore, by Definition 3.3.1, $(\tilde{z}_1, \tilde{z}_2)$ dominates $(\hat{z}_1^k, \hat{z}_2^k)$. However, note that we get $(\hat{z}_1^k, \hat{z}_2^k)$ by solving $\mathbb{H}_{\mathcal{S} \cap \{f_2 \leq u_2\}}(\{f_1, f_2\})$ in Algorithm 2, which is a Pareto point by Theorem 3.3.1. Therefore, it contradicts the assumption that $(\hat{z}_1^k, \hat{z}_2^k)$ is a Pareto point. □

By the above Theorem 3.3.3 and 3.3.4, there is no unfounded Pareto points whose f_2 is greater than or equals to $\bar{z}_2$, where $\bar{z}_2$ is the nadir value of f_2 for $\mathbb{P}_{\mathcal{S} \cap \{f_2(s) \leq u_2\}}(\{f_1, f_2\})$. Thus, we can strictly restrict search space by decreasing upper bound of f_2 up to $\bar{z}_2 - 1$ without missing any Pareto point. Also, throughout a sequence of monotonically decreasing u_2, we always get a Pareto point by solving the lexicographic optimization $\mathbb{H}_{\mathcal{S} \cap \{f_2 \leq u_2\}}(\{f_1, f_2\})$. This implies that Algorithm 2 finds all Pareto points for $\mathbb{P}_{\mathcal{S}}(\{f_1, f_2\})$.

Theorem 3.3.5. *Algorithm 2 finds all Pareto points in $\mathcal{P}$ for $\mathbb{P}_{\mathcal{S}}(\{f_1, f_2\})$.*

Proof. First, by Corollary 3.3.1.1, we have proved that the corner point $(z_1^*, \bar{z}_2)$ is in $\mathcal{P}$, which is the first Pareto point founded by Algorithm 2. By Theorem 3.3.3, we know that $\hat{z}_2^k$ is the nadir value of f_2 for $\mathbb{P}_{\mathcal{S} \cap \{f_2 \leq u_2\}}(\{f_1, f_2\})$ where $u_2 = \hat{z}_2^{k-1} - 1$ and $\hat{z}_2^0 = \infty$ for each iteration $k \in \{1, \cdots, K\}$ in Algorithm 2. Thus, there is no other Pareto point such that f_2 is between $\hat{z}_2^k + 1$ and u_2 inclusively. Also, by Theorem 3.3.4, we know that a point $(\hat{z}_1^k, \hat{z}_2^k)$ is a unique Pareto point at $f_2 = \hat{z}_2^k$ for each iteration $k \in \{1, \cdots, K\}$ in Algorithm 2. Therefore, there is no unfounded Pareto points such that f_2 is greater than or equal to $\hat{z}_2^k$. Thus, we don't miss any Pareto point by strictly decreasing a upper bound on f_2 by substracting 1 from $\hat{z}_2^k$. By the termination condition in Algorithm 2, we know that we search whole possible range of f_2 shown in Theorem 3.3.2, where Pareto points exists. Therefore, Algorithm 2 finds all Pareto points in $\mathcal{P}$. □

3.3.2.2 Multi-objective Problem with Integer Values (MOPI)

In previous section, we explained how to find the Pareto frontier $\mathcal{P}$ for BOPI. Conceptually, the method for BOPI is pretty straightforward. By generalizing this method, we can find the Pareto frontier for a n dimensional multi-objective problem, $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$. Unfortunately, the notation gets a bit heavy in the general case that has more than two objectives. For the sake of simplicity, given two vectors $u = (u_1, \ldots, u_p)$ and $r = (r_1, \ldots, r_p)$ for all $p \in \{1, \ldots, n\}$, we use the operator overloading $u \prec r$ if and only if $u_i \leq r_i$ for all $i \in \{1, \ldots, p\}$ and $\exists i \in \{1, \ldots, p\}$ such that $u_i < r_i$ as an extension of Definition 3.3.1 to a vector of arbitrary length. Similarily, $u \preceq r$ represents $u_i \leq r_i$ for all $i \in \{1, \ldots, p\}$. We also denote the set $\{s : f(s) \preceq u\}$ in a subscript by $\{f(s) \preceq u\}$ or $\{f \preceq u\}$ if there is no confusion.

(a) Range of Metric Space for MOPI Unlike BOPI, the calculation of the range of metrics in problems involving more than two objectives is not a trivial task [55, 118, 119, 120]. First, we can get easily the ideal point $z^* = (z_1^*, \ldots, z_n^*)$ of the Pareto frontier, which defines lower bounds on each metric value, by solving $z_i^* = F_{\mathcal{S}}(\{(f_1, w_1, \infty), \ldots, (f_i, w_i, \infty), \ldots, (f_n, w_n, \infty)\})$ where $w_i = 1$ for all $i \in \{1, \ldots, n\}$ and $w_j = 0$ for all $j \neq i$.

While the ideal point is easily attainable by optimizing the individual objective, the nadir point is not. To define a region of objective space that contains the Pareto frontier, we suggest an algorithm to calculate nadir values of each metric. Without loss of generality, we just show you a way to find the nadir value of the last metric f_n, $\bar{z}_n = \max\{f_n(s) : s \in \mathcal{E}\}$. The nadir value of other metrics can be calculated in a similar way. By definition of the ideal and nadir value, it follows that $z_i^* \leq \hat{z}_i \leq \bar{z}_i$ for all $i \in \{1, .., n\}$ for each Pareto point $(\hat{z}_1, \ldots, \hat{z}_n)$ in $\mathcal{P}$.

Theorem 3.3.6. *Given the $n-1$ dimensional Pareto frontier $\mathcal{P}_{\mathcal{S}}(\{f_1, \cdots, f_{n-1}\})$ of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_{n-1}\})$, the nadir value of f_n for $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ is the following value:*

$$\bar{z}_n = \max_{k=1,\ldots,K} \hat{z}_n^k$$

where $\hat{z}_n^k = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1^k), \cdots, (f_{n-1}, 0, \hat{z}_{n-1}^k), (f_n, 1, \infty)\})$ *for each point* $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k) \in \mathcal{P}_{\mathcal{S}}(\{f_1, \cdots, f_{n-1}\})$*, and* $K = |\mathcal{P}_{\mathcal{S}}(\{f_1, \cdots, f_{n-1}\})|$.

Proof. Suppose that there exists a Pareto point $\tilde{z} = (\tilde{z}_1, \ldots, \tilde{z}_n) \in \mathcal{P}$ such that $\tilde{z}_n > \bar{z}_n$. Since we assume that $\bar{z}_n$ is the maximum value of f_n among all Pareto points $(\hat{z}_1^k, \ldots, \hat{z}_n^k)$ in Therorem 3.3.6, it holds that $\tilde{z}_n > \hat{z}_n^k$ for all $k \in \{1, \ldots, K\}$. Therefore, the projection $(\tilde{z}_1, \ldots, \tilde{z}_{n-1})$ of the Pareto point $\tilde{z}$ to the first $n-1$ dimensional metric subspace should not be dominated by any Pareto points $(\hat{z}_1^k, \ldots, \hat{z}_{n-1}^k)$ in $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$. Otherwise, there exists some $k \in \{1, \ldots, K\}$ such that the Pareto point $(\hat{z}_1^k, \ldots, \hat{z}_n^k)$ dominates $\tilde{z}$, and then it contradicts the assumption of $\tilde{z} \in \mathcal{P}$. Note that, for any metric point in the $n-1$ dimensional metric subspace, it should be either a Pareto point or dominated by some Pareto point in $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$. Since $(\tilde{z}_1, \ldots, \tilde{z}_{n-1})$ is not dominated by $(\hat{z}_1^k, \ldots, \hat{z}_{n-1}^k)$ for all $k \in \{1, \ldots, K\}$, it should be a Pareto point in $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$. In Theorem 3.3.6, we get the nadir value $\bar{z}_n$ of f_n by taking the largest value of $\hat{z}_n^k$ from all Pareto points $(\hat{z}_1^k, \ldots, \hat{z}_{n-1}^k)$ in $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$, $k \in \{1, \ldots, K\}$. Therefore, it should be $\bar{z}_n \geq \tilde{z}_n$. This contradicts $\tilde{z}_n > \bar{z}_n$. Therefore, there is no Pareto point $(\hat{z}_1, \ldots, \hat{z}_n) \in \mathcal{P}$ such that $\hat{z}_n > \bar{z}_n$. In other word, if some point $(z_1, \ldots, z_n)$ has $z_n > \bar{z}_n$, then $(z_1, \ldots, z_n) \notin \mathcal{P}$. □

Let $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$ be the Pareto frontier of the $n-1$ objective problem, $\mathbb{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$, and K be the number of points in the Pareto frontier $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$. For each point $m = (\hat{z}_1^k, \ldots, \hat{z}_{n-1}^k)$ in $\mathcal{P}_{\mathcal{S}}(\{f_1, \ldots, f_{n-1}\})$ for all $k \in \{1, \ldots, K\}$, we solve for f_n relative to m and denote it by $\hat{z}_n^k = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1^k), \ldots, (f_{n-1}, 0, \hat{z}_{n-1}^k), (f_n, 1, \infty)\})$. Then, the nadir value $\bar{z}_n$ of f_n is computed by $\max_{k=1,\ldots,K} \hat{z}_n^k$.

(b) Recursive method for MOPI Özlen and Azizoğlu [115] proposes a recursive algorithm to find the Pareto frontier of the MOIP by recursively identifying the Pareto frontier for sub-problems with fewer objective functions. The basic idea for solving a multi-objective minimization problem with n integer-valued objectives is the following. We improve an upper bound u_n on f_n monotonically throughout a recursive algorithm, starting with $u_n = +\infty$. In each iteration, we first find the

Pareto frontier of the problem with $n-1$ metrics with an upper bound u_n on f_n, i.e., we solve the problem $\mathbb{P}_{\mathcal{S}\cap\{f_n\le u_n\}}(\{f_1,\cdots,f_{n-1}\})$. Suppose that the Pareto frontier of this problem consists of K points denoted by $(\hat{z}_1^k,\cdots,\hat{z}_{n-1}^k)$ for $k=1,\ldots,K$. Then, for each $k=1,\ldots,K$, we minimize the last metric $f_n(s)$ subject to the constraints

$$s \in \mathcal{S}$$

$$f_i(s) \le \hat{z}_i^k,\ \forall i = 1,\ldots,n-1$$

and denote the optimal value by $\hat{z}_n^k$. Let $\bar{z}_n = \max_{k=1,\ldots,K} \hat{z}_n^k$ be the *nadir value* of f_n over the current set. Finally, we tighten an upper bound on f_n by subtracting 1 from the nadir value $\bar{z}_n$ and repeat this process to find all Pareto points until u_n makes an infeasible set. Note that the last step utilizes the fact that the metric functions are integer-valued. We summarize this recursive method in Algorithm 3, incorporating the fact that MOPI has integer-only objective values.

```
Algorithm 3: Recursive method for MOPI, $\mathbb{P}_{\mathcal{S}}(\{f_1,\cdots,f_n\})$
  INPUT  : $\mathbb{P}_{\mathcal{S}}(\{f_1,\cdots,f_n\})$
  INIT: Initialize the upper bound $u_n$ of $f_n$ to $\infty$
  REPEAT
      1. First find the Pareto frontier of the problem with $n-1$ metrics with an upper bound
         $u_n$ on $f_n$, i.e., we solve the problem $\mathbb{P}_{\mathcal{S}\cap\{f_n\le u_n\}}(\{f_1,\cdots,f_{n-1}\})$.
      2. Suppose the Pareto frontier of this problem consists of $K$ points denoted by
         $(\hat{z}_1^k,\cdots,\hat{z}_{n-1}^k)$ for $k=1,\ldots,K$. Then, for each $k \in 1,\ldots,K$, minimize the last
         metric $f_n(s)$ subject to the constraints $s \in S$ and $f_i(s) \le \hat{z}_i^k$, $\forall i=1,\ldots,n-1$, and
         denote the optimal value by $\hat{z}_n^k$.
      3. Add points $(\hat{z}_1^k,\cdots,\hat{z}_n^k)$, $k=1,\ldots,K$, to $\mathcal{P}$
      4. Let $\bar{z}_n = \max_{k=1,\ldots,K} \hat{z}_n^k$ be the nadir value of $f_n$ over the current set.
      5. Tighten an upper bound on $f_n$ by setting $u_n := \bar{z}_n - 1$. Note that this step utilizes
         the fact that the metric functions are integer-valued.
  UNTIL the problem becomes infeasible for some value of $u_n$
  OUTPUT $\mathcal{P}$
  :
```

We claim the following three theorems in order to prove that Algorithm 3 finds all Pareto points in the Pareto frontier of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$.

Theorem 3.3.7. *Given the maximum value $\bar{z}_n^t$ gained from $\mathcal{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_{n-1}\})$ in Algorithm 3 for each iteration $t \in \{1, \dots, T\}$ where T is total number of interations until Algorithm 3 terminates, there is no other Pareto point for $\mathbb{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_n\})$ such that $f_n > \bar{z}_n^t$ where $u_n^t = \bar{z}_n^{t-1} - 1$ and $\bar{z}_n^0 = \infty$.*

Proof. By Theorem 3.3.6, we know that $\bar{z}_n^t$ is the nadir value of f_n for $\mathbb{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_n\})$. By Definition 3.3.6, there is no Pareto point such that $\bar{z}_n^t < f_n \le u_n^t$ for each iteration $t \in \{1, \dots, T\}$. $\square$

Theorem 3.3.8. *For each iteration $t \in \{1, \dots, T\}$ in Algorithm 3, all Pareto points of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ such that $f_n = \bar{z}_n^t$ are founded by Algorithm 3.*

Proof. Suppose that there is an unfounded distinct Pareto point $(\tilde{z}_1, \cdots, \tilde{z}_n)$ such that the value of f_n is $\bar{z}_n^t$ (i.e., $\tilde{z}_n = \bar{z}_n^t$) for some iteration $t \in \{1, \dots, T\}$ in Algorithm 3. Since $(\tilde{z}_1, \cdots, \tilde{z}_n)$ is a Pareto point of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ and $\bar{z}_n^t$ is the nadir value of f_n in the iteration t, $(\tilde{z}_1, \cdots, \tilde{z}_{n-1})$ should be a Pareto point in the metric space projected into other $n-1$ objectives. Therefore, $(\tilde{z}_1, \cdots, \tilde{z}_{n-1})$ should be in $\mathcal{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_{n-1}\})$. Since we get metric points $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k, \hat{z}_n^k)$ for all $k \in \{1, \cdots, K\}$ from $\mathcal{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_{n-1}\})$ for the iteration t in Algorithm 3 where $K = |\mathcal{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_{n-1}\})|$, the unfounded Pareto point $(\tilde{z}_1, \cdots, \tilde{z}_n)$ should be one of the metric points $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k, \hat{z}_n^k)$ for $k \in \{1, \cdots, K\}$. It contradicts the assumption that the distinct Pareto point $(\tilde{z}_1, \cdots, \tilde{z}_n)$ is unfounded. $\square$

Theorem 3.3.9. *For each iteration $t \in \{1, \dots, T\}$ in Algorithm 3, $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k, \hat{z}_n^k)$ is a Pareto point of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ for all $k \in \{1, \cdots, K\}$ where $K = |\mathcal{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_{n-1}\})|$.*

Proof. Suppose that some metric point $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k, \hat{z}_n^k)$ is not Pareto point of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$. Since we get the metric point $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k, \hat{z}_n^k)$ from $\mathcal{P}_{\mathcal{S}\cap\{f_n \le u_n^t\}}(\{f_1, \cdots, f_{n-1}\})$, we know that $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k)$ is a Pareto point in the projected metric space. Given the Pareto point $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k)$ in the projected metric space, the value of f_n could be less than $\hat{z}_n^k$ because the metric point

$(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k, \hat{z}_n^k)$ is not Pareto point. However, it contradicts the assumption that $\hat{z}_n^k$ is the minimum value of f_n given the projected point $(\hat{z}_1^k, \cdots, \hat{z}_{n-1}^k)$. □

By Theorem 3.3.7 and 3.3.8 above, there is no unfounded Pareto points whose f_n is greater than or equals to $\bar{z}_n$ when we strictly restrict a search space by decreasing an upper bounds of f_n to $\bar{z}_n - 1$. Thus, throughout a sequence of monotonically decreasing an upper found u_n on f_n, we don't lose any Pareto point. This implies that Algorithm 3 finds all Pareto points.

Theorem 3.3.10. *The recursive method in Algorithm 3 finds all Pareto points of* $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$.

Proof. By Theorem 3.3.7 and 3.3.8, we see that we can find all Pareto points at the nadir value $\bar{z}_n^1$ of f_n for $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$ by the first iteration with $u_n^1 = \infty$ in Algorithm 3 and there is no Pareto point above the nadir value. Therefore, we have found all Pareto points that f_n is greater than or equal to $\bar{z}_n^1$ so that we can strictly decrease an upper bound on f_n by substracting 1 from $\bar{z}_n^1$. At each iteration $t \in \{1, \ldots, T\}$ in Algorithm 3, we can find all Pareto points at the nadir value $\bar{z}_n^t$ of f_n for $\mathbb{P}_{\mathcal{S}\cap\{f_n \leq u_n^t\}}(\{f_1, \cdots, f_n\})$ where $u_n^t = \bar{z}_n^{t-1} - 1$ and there is no unfounded Pareto point between $\bar{z}_n^t$ and $\bar{z}_n^{t-1} - 1$ for $\mathbb{P}_{\mathcal{S}\cap\{f_n \leq u_n^t\}}(\{f_1, \cdots, f_n\})$. By induction, we have found all Pareto points that f_n is greater than or equal to $\bar{z}_n^{t-1}$ at the iteration $t \in \{1, \ldots, T\}$. By the termination condition in Algorithm 3, we know that we search whole possible range of f_n where Pareto points exists by Definition 3.3.5 and 3.3.6. Therefore, Algorithm 3 finds all Pareto points in $\mathcal{P}$ by Theorem 3.3.9. □

3.3.3 Sequential method for MOPI

In this section, given the fixed number of objectives, we restate the original recursive algorithm [115] by using the lexicographic optimization problem $\mathbb{H}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$, which is defined in Algorithm 1. Our sequential method in Algorithm 4 shows that we can get each Pareto point $(\hat{z}_1^k, \cdots, \hat{z}_n^k)$ by adjusting a vector of upper bounds $u = (u_2, \cdots, u_n)$ for $\mathbb{H}_{\mathcal{S}\cap\{f_2 \leq u_2, \cdots, f_n(s) \leq u_n\}}$ $(\{f_1, \cdots, f_n\})$ appropriately. The sequential algorithm to find the Pareto frontier of MOPI can be described as follows:

Algorithm 4: Sequential method for $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$

```
INPUT  : $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$
INIT: $k = 1, \mathcal{P} = \{\emptyset\}, u_n = \infty$
WHILE $\mathcal{S} \cap \{s : f_n(s) \le u_n\} \ne \emptyset$ DO
    Let $\mathcal{U}_n = \{\emptyset\}$ and $u_{n-1} = \infty$
    WHILE $\mathcal{S} \cap \{s : f_{n-1}(s) \le u_{n-1},\ f_n(s) \le u_n\} \ne \emptyset$ DO
        Let $\mathcal{U}_{n-1} = \{\emptyset\}$ and $u_{n-2} = \infty$
        $\vdots$
        WHILE $\mathcal{S} \cap \{s : f_3(s) \le u_3, \cdots,\ f_n(s) \le u_n\} \ne \emptyset$ DO
            Let $\mathcal{U}_3 = \{\emptyset\}$ and $u_2 = \infty$
            WHILE $\mathcal{S} \cap \{s : f_2(s) \le u_2, \cdots,\ f_n(s) \le u_n\} \ne \emptyset$ DO
                Let $\mathcal{U}_2 = \{\emptyset\}$
                BEGIN Lexicographic Optimization
                    Solve $\mathbb{H}_{\mathcal{S} \cap \{f_2 \le u_2, \cdots, f_n \le u_n\}}(\{f_1, \cdots, f_n\})$
                    Set $(\hat{z}_1^k, \cdots, \hat{z}_n^k)$ be the solution
                    Update $\mathcal{P} = \mathcal{P} \cup (\hat{z}_1^k, \cdots, \hat{z}_n^k)$ and $k = k + 1$
                    Set $\mathcal{U}_2 = \mathcal{U}_2 \cup \{\hat{z}_2^k\}$
                        $\vdots$
                    Set $\mathcal{U}_n = \mathcal{U}_n \cup \{\hat{z}_n^k\}$
                END
                Set $\bar{z}_2 = \max\{\mathcal{U}_2\}$ and $u_2 = \bar{z}_2 - 1$
            END
            Set $\bar{z}_3 = \max\{\mathcal{U}_3\}$ and $u_3 = \bar{z}_3 - 1$
        END
        $\vdots$
        Set $\bar{z}_{n-1} = \max\{\mathcal{U}_{n-1}\}$ and $u_{n-1} = \bar{z}_{n-1} - 1$
    END
    Set $\bar{z}_n = \max\{\mathcal{U}_n\}$ and $u_n = \bar{z}_n - 1$
END
OUTPUT $\mathcal{P}$
:
```

Briefly, by Theorem 3.3.11, we can combine the first step to find the $n-1$ dimensional Pareto frontier $\mathcal{P}_{\mathcal{S} \cap \{f_n \le u_n\}}(\{f_1, \cdots, f_{n-1}\})$ and the second step to minimize the last index metric relative to each Pareto point of $\mathcal{P}_{\mathcal{S} \cap \{f_n \le u_n\}}(\{f_1, \cdots, f_{n-1}\})$ in Algorithm 3 into a single lexicographic optimization step via $\mathbb{H}_{\mathcal{S} \cap \{f_2 \le u_2, \cdots, f_n \le u_n\}}(\{f_1, \cdots, f_n\})$. We claim the following theorems to prove that Algorithm 4 above finds all Pareto points in the Pareto frontier of $\mathbb{P}_{\mathcal{S}}(\{f_1, \cdots, f_n\})$.

Theorem 3.3.11. *For all upper bounds $(u_2, \cdots, u_n)$ on metrics $\{f_2, \cdots, f_n\}$, if there exist a metric point $(\hat{z}_1, \cdots, \hat{z}_{n-1})$ gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_{n-1}\leq u_{n-1}\}}(\{f_1, \cdots, f_{n-1}\})$ and a metric value $\hat{z}_n = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), \cdots, (f_{n-1}, 0, \hat{z}_{n-1}), (f_n, 1, u_n)\})$, then the metric point $(\hat{z}_1, \cdots, \hat{z}_{n-1}, \hat{z}_n)$ can be gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_n\leq u_n\}}(\{f_1, \cdots, f_n\})$ and it is a Pareto point in $\mathcal{P}$.*

Proof. Given an upper bound $(u_2, \cdots, u_n)$, suppose that there exists a metric point $(\hat{z}_1, \cdots, \hat{z}_{n-1})$ gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_{n-1}\leq u_{n-1}\}}(\{f_1, \cdots, f_{n-1}\})$. Then, by Algorithm 1, we get the value $\hat{z}_n$ of last metric f_n in $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_n\leq u_n\}}(\{f_1, \cdots, f_n\})$ by solving $\hat{z}_n = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), \cdots, (f_{n-1}, 0, \hat{z}_{n-1}), (f_n, 1, u_n)\})$. Therefore, if the metric point $(\hat{z}_1, \cdots, \hat{z}_{n-1})$ can be gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_{n-1}\leq u_{n-1}\}}(\{f_1, \cdots, f_{n-1}\})$, both $(\hat{z}_1, \cdots, \hat{z}_{n-1})$ and $\hat{z}_n$ can be gained together by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_n\leq u_n\}}(\{f_1, \cdots, f_n\})$ by induction. Also, by Theorem 3.3.1, it is a Pareto point in $\mathcal{P}$. □

Theorem 3.3.12. *Given the fixed number of objectives, the sequential method in Algorithm 4 finds all Pareto points in $\mathcal{P}$.*

Proof. First, suppose that we have two objectives ($n = 2$). Trivially, Algorithm 4 is exactly same with Algorithm 2 for bi-objective case. Thus, it can find all Pareto points for $n = 2$. By induction, suppose that Algorithm 4 generates all Pareto points for n objectives. Then, we use Algorithm 4 for Step 1 in Algorithm 3 to find the Pareto frontier of the n objectives problem. As Step 2 in Algorithm 3, we get the minimun value $\hat{z}_{n+1}$ of the metric f_{n+1} by solving $\hat{z}_{n+1} = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), \cdots, (f_n, 0, \hat{z}_n), (f_{n+1}, 1, u_{n+1})\})$ for each point $(\hat{z}_1, \cdots, \hat{z}_n) = \mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_n\leq u_n\}}(\{f_1, \cdots, f_n\})$ given an upper bound $(u_2, \cdots, u_{n+1})$. By Theorem 3.3.11, we know that $(\hat{z}_1, \cdots, \hat{z}_n)$ and $\hat{z}_{n+1}$ can be gained together by solving $\mathbb{H}_{\mathcal{S}\cap\{f_2\leq u_2,\cdots,f_{n+1}\leq u_{n+1}\}}(\{f_1, \cdots, f_{n+1}\})$ and this change of the calculation order does not impact on the individual value of $(\hat{z}_1, \cdots, \hat{z}_n, \hat{z}_{n+1})$. Therefore, Algorithm 4 also works for $n + 1$ objective problem. □

3.3.4 Improvements

The recursive and sequential method are based on solving a sequence of IP subproblems. Therefore, it is critical to solve each IP problem quickly. A key idea to improve them is to make use of information obtained from the already solved subproblems. We improve Algorithm 4 by using the set of already solved subproblems and their Pareto points to reduce the solving of duplicate IPs that produce repeated Paeto points through the relaxation strategy introduced by Özlen et al. [116]. In addition to implementing this relaxation, we also accelerate CPLEX IP solver by providing the set of all known solutions from prior subproblems as feasible starting points, known as a warm start solutions. Basically, there are a cluster of the similar IP problems with slightly different upper bounds on metrics. Since Algorithm 4 improves the upper bounds monotonically throughout the algorithm, there are some IP problems that are feasible to the former optimal solutions.

3.3.4.1 Relaxation

Özlen et al. [116] introduce a way to improve their ealier recurisve algorithm [115] by using the relaxation problem that has been solved before. For the sake of simplicity, we denote $\mathbb{P}_{\mathcal{S}\cap\{f_1(s)\le u_1,\cdots,f_p(s)\le u_p\}}(\{f_1,\cdots,f_p\})$ by $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$ where $u=(u_1,\cdots,u_p)$ for all $p\in\{1,\cdots,n\}$. We restate the definition and lemmas from Özlen et al. [116] using our notations:

Definition 3.3.9. *Relaxation: Given two vectors $u=(u_1,\cdots,u_p)$ and $r=(r_1,\cdots,r_p)$ for a positive number $p\in\{1,\cdots,n\}$, $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\cdots,f_p\})$ is a relaxation of $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$ if $u\prec r$.*

Lemma 3.3.13. *Let $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\cdots,f_p\})$ be a relaxation of $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$ for a positive number $p\in\{1,\cdots,n\}$. If $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\cdots,f_p\})$ is infeasible, then $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$ is also infeasible.*

Lemma 3.3.14. *Let $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\cdots,f_p\})$ be a relaxation of $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$ for a positive number $p\in\{1,\cdots,n\}$. If every Pareto point for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\cdots,f_p\})$ is also feasi-*

ble for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$*, then the set of all Pareto points for* $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\cdots,f_p\})$ *is precisely the set of all Pareto points for* $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\cdots,f_p\})$*.*

By a slight modification of Lemma 3.3.13 and 3.3.14, we prove Theorem 3.3.15 to improve our sequential method for MOPI. Given $u \preceq r$, if a Pareto point gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\ldots,f_n\})$ for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\ldots,f_n\})$ is feasible to $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\ldots,f_n\})$, we can skip solving $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\ldots,f_n\})$ for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\ldots,f_n\})$ in Algorithm 4 since both have the same Pareto point by Theorem 3.3.15.

Theorem 3.3.15. *Let* $\hat{m}$ *be a Pareto point gained by solving* $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\ldots,f_n\})$*. If* $u \preceq r$ *and* $\hat{m}$ *is feasible to* $\mathcal{S}\cap\{s : f(s)\preceq u\}$*, then the Pareto point gained by solving* $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\ldots,f_n\})$ *is the same with* $\hat{m}$*.*

Proof. Let $\hat{m} = (\hat{z}_1,\cdots,\hat{z}_n)$ be a Pareto point gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\ldots,f_n\})$. By Algorithm 1, we get $\hat{z}_1$ by solving $F_{\mathcal{S}}(\{(f_1,1,r_1),(f_2,0,r_2),\cdots,(f_n,0,r_n)\})$ for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1\})$. Similarily, let $\tilde{m} = (\tilde{z}_1,\cdots,\tilde{z}_n)$ be a Pareto point gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\ldots,f_n\})$ where $u\preceq r$. Then, we get $\tilde{z}_1$ by solving $F_{\mathcal{S}}(\{(f_1,1,u_1),(f_2,0,u_2),\cdots,(f_n,0,u_n)\})$ for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1\})$. Note that $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1\})$ is a relaxation of $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1\})$ by Definition 3.3.9. Also, $\hat{z}_1$ is a unique Pareto point for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1\})$ and $\tilde{z}_1$ is one for $\mathbb{P}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1\})$. Since $\hat{z}_1$ is feasible to $\mathcal{S}\cap\{f_1\leq u_1\}$, it holds that $\hat{z}_1=\tilde{z}_1$ by Lemma 3.3.14. Sequentially, we can prove all $\hat{z}_i=\tilde{z}_i$ for all $i\in\{1,\cdots,n\}$. Thus, the Pareto point $\hat{m}$ gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq r\}}(\{f_1,\ldots,f_n\})$ is the same with $\tilde{m}$ gained by solving $\mathbb{H}_{\mathcal{S}\cap\{f(s)\preceq u\}}(\{f_1,\ldots,f_n\})$ if $u\preceq r$ and $hatm$ is feasible to $\mathcal{S}\cap\{f(s)\preceq u\}$. □

3.3.4.2 Warm Start

The warm start is a solution that could possibly help IP solver find an initial solution quickly and provide a good bound on the optimal objective value of the new problem that has to be solved. This warm start could come from previous similar problems that have been solved. Also, we can provide CPLEX solver with multiple warm starts. In this section, we suggest our multiple warm starts strategy to improve computational time of our sequential method in Algorithm 4.

In Algorithm 4, we show that we can find each Pareto point by the lexicographic optimization, $\mathbb{H}_{\mathcal{S}}(\{f_1, \dots, f_n\})$, which consists of a cluster of the similar IP problems except slightly different upper bounds on metrics. Also, since we progressively tighten upper bounds $(u_1, \cdots, u_n)$ on metrics in order, the former optimal solution for $\mathbb{P}_{\mathcal{S}' \cap \{f_1 \le \hat{z}_1, \dots, f_{p-1} \le \hat{z}_{p-1}\}}(\{f_p\})$ is always feasible for the next IP problem $\mathbb{P}_{\mathcal{S}' \cap \{f_1 \le \hat{z}_1, \dots, f_{p-1} \le \hat{z}_{p-1}, f_p \le \hat{z}_p\}}(\{f_{p+1}\})$ for a positive number $p \in \{1, \cdots, n\}$ where $\mathcal{S}' = \mathcal{S} \cap \{s : f_1(s) \le u_1, \cdots, f_n(s) \le u_n\}$ and $\hat{z}_p = F_{\mathcal{S}}(\{(f_1, 0, \hat{z}_1), \cdots, (f_{p-1}, 0, \hat{z}_{p-1})$ $, (f_p, 1, u_p), (f_{p+1}, 0, u_{p+1}), \cdots, (f_n, 0, u_n)\})$. In addition, all previous optimal solutions are feasible to all problems that come after whithin Algorithm 1. Therefore, we can cumulatively use the former optimal solutions as warm starts for the following IP problems in Algorithm 1. In this paper, we use the following multiple warm starts strategy described in Algorithm 5.

Algorithm 5: Enhanced $\mathbb{H}_{\mathcal{S}}(\{f_1, \dots, f_n\})$ by warm starts

```
   INPUT  : $\mathbb{P}_{\mathcal{S} \cap \{f_2 \le u_2, \dots, f_n \le u_n\}}(\{f_1, \dots, f_n\})$
 1 STEP 0 : check feasibility for the set of $\mathcal{S} \cap \{s : f_2(s) \le u_2, \dots, f_n(s) \le u_n\}$
 2     Solve $\mathbb{P}_{\mathcal{S} \cap \{f_2 \le u_2, \dots, f_n \le u_n\}}(\{\emptyset\})$
 3     Set $\tilde{z}^0 = (\tilde{z}^0_1, \dots, \tilde{z}^0_n)$ be the solution of $\mathbb{P}_{\mathcal{S} \cap \{f_2 \le u_2, \dots, f_n \le u_n\}}(\{\emptyset\})$
 4 STEP 1 : optimize $f_1$
 5     Solve $\mathbb{P}_{\mathcal{S} \cap \{f_2 \le u_2, \dots, f_n \le u_n\}}(\{f_1\})$ with a warm start $\tilde{z}^0$
 6     Set $\tilde{z}^1 = (\tilde{z}^1_1, \dots, \tilde{z}^1_n)$ be the solution of $\mathbb{P}_{\mathcal{S} \cap \{f_2 \le u_2, \dots, f_n \le u_n\}}(\{f_1\})$
 7 STEP 2 : optimize $f_2$
 8     Solve $\mathbb{P}_{\mathcal{S} \cap \{f_1 \le \tilde{z}^1_1, f_2 \le u_2, \dots, f_n \le u_n\}}(\{f_2\})$ with warm starts $\tilde{z}^k$ for all $k = 0, 1$
 9     Set $\tilde{z}^2 = (\tilde{z}^2_1, \dots, \tilde{z}^2_n)$ be the solution of $\mathbb{P}_{\mathcal{S} \cap \{f_1 \le \tilde{z}^1_1, f_2 \le u_2, \dots, f_n \le u_n\}}(\{f_2\})$
10 STEP 3 : optimize $f_3$
11     Solve $\mathbb{P}_{\mathcal{S} \cap \{f_1 \le \tilde{z}^1_1, f_2 \le \tilde{z}^2_2, f_3 \le u_3, \dots, f_n \le u_n\}}(\{f_3\})$ with warm starts $\tilde{z}^k$ for all $k = 0, 1, 2$
12     Set $\tilde{z}^3 = (\tilde{z}^3_1, \dots, \tilde{z}^3_n)$ be the solution of $\mathbb{P}_{\mathcal{S} \cap \{f_1 \le \tilde{z}^1_1, f_2 \le \tilde{z}^2_2, f_3 \le u_3, \dots, f_n \le u_n\}}(\{f_3\})$
13     ⋮
14 STEP n : optimize $f_n$
15     Solve $\mathbb{P}_{\mathcal{S} \cap \{f_1 \le \tilde{z}^1_1, \dots, f_{n-1} \le \tilde{z}^{n-1}_{n-1}, f_n \le u_n\}}(\{f_n\})$ with warm starts $\tilde{z}^k$ for all $k = 0, 1, \dots, n-1$
16     Set $\tilde{z}^n = (\tilde{z}^n_1, \dots, \tilde{z}^n_n)$ be the solution of $\mathbb{P}_{\mathcal{S} \cap \{f_1 \le \tilde{z}^1_1, \dots, f_{n-1} \le \tilde{z}^{n-1}_{n-1}, f_n \le u_n\}}(\{f_n\})$
   OUTPUT : $(\tilde{z}^n_1, \dots, \tilde{z}^n_n)$
```

3.4 Computational Experiments

In this section, we evaluate the effectiveness of the recursive method in solving the PEDS problem. First, we assess the tractability with respect to run time; we pay particular attention to the impact of using warm starts. Second, we investigate the size of the Pareto frontiers for standard problem instances. Third, we present a case study demonstrating how the Chief Residents would evaluate these Pareto sets.

3.4.1 Data Sets

Our computational results are based on real-world data from the pediatric ED at the University of Michigan C.S. Mott Children's Hospital. This ED is staffed by residents who rotate on a monthly basis. They collectively cover 7 shifts per day, 7 days per week.

The rules taken into account in this scheduling problem are outlined in Section 2.2.2. Note that, for the sake of exposition, we have simplified some of the details of the actual real-world problem. In our experience, these additional details do not significantly change the structure of the problem, nor do they have an impact on our computational results. Recall that the metrics under consideration are the total numbers of: (1) bad sleep patterns assigned; (2) post-continuity clinics worked; (3) vacation requests denied; and (4) flex shifts uncovered.

We collected data corresponding to 27 months, spanning July 2012 to June 2015 but excluding the months of December, January, and February because these are treated separately by the scheduler due to the winter holidays (December, January) and the shorter-than-usual month (February).

To thoroughly evaluate the tractability and usability of our approach, we explore a wider range of problem instances by creating 9 levels of flexibility for each of the 27 months. This is accomplished by varying 2 basic characteristics of the problem:

- **We vary restrictions on resident equity.** In the first case all residents have to work the same number of shifts (plus or minus one, as needed to account for not being able to evenly divide the total number of shifts). In the second case, each resident can be assigned to the average number of shifts plus or minus up to five additional shifts. In the third case, each resident can be assigned to as much as nine shifts' deviation from the average. Thus, the first case is very tightly constrained but treats the residents most equitably, while the third case has the greatest flexibility but could result in significant inequity across residents. In this paper, we denote these three cases by S1, S2, and S3, resepectively.

- **We vary the set of vacation requests for each resident.** For each resident, we randomly generate single-day vacation requests. In the first case, we assume that each resident has a 10% chance of requesting a given day off (so, on average, they would each make roughly three requests per month); in the second case, the probability of a weekday request is 10% and the probability of a weekend request if 20%; in the third case, all days have a 20% probability of being requested as a day off. In this paper, we denote these three cases by V1, V2, and V3, resepectively.

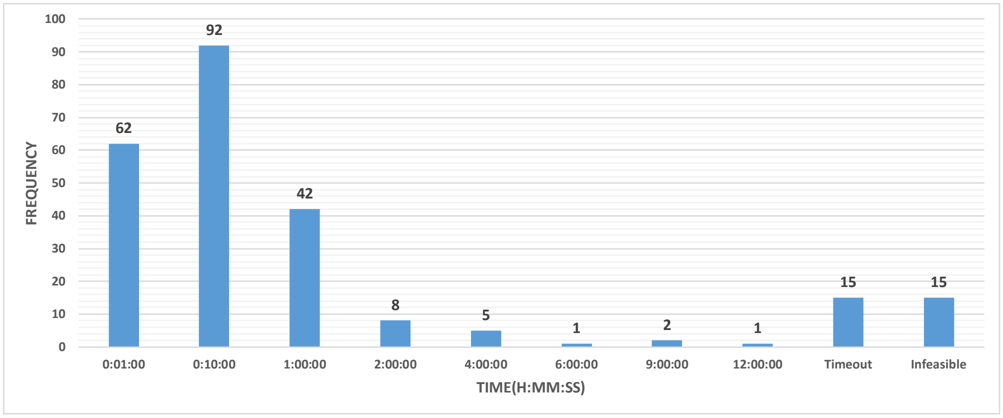
Randomly generating one trial for each of these 9 levels of flexibility (three options for equity times three options for vacation requests) for each of the 27 months yields 243 problem instances. Of these, 15 are easily found to be infeasible – not surprisingly, this is typically the case where the schedules are most tightly restricted (for more details, see Table B.1 in the appendix).

For the remaining instances, we solved them using the IP-based recursive method by implementing our approach in Python and C++ using CPLEX 12.6 C++ API with a 0.1% optimality gap. All computational experiments were performed on an Intel i7 3.40 GHz processor with 8 GB of memory. We terminate if any individual IP runs beyond a 30-minute time limit.

3.4.2 Tractability

To demonstrate the tractability of the enhanced recursive method (including the use of warm starts), we evaluated the run time of the 243 problem instances described above. Of these, 15 instances were infeasible, 15 instances could not be solved to optimality given the thirty minute time limit for the individual IPs, and the remaining 213 instances were solved to completion.

For the 15 instances that had no feasible solution, it was trivial to prove infeasibility, taking no longer than two seconds per instance. For the 15 instances that timed out, it was often the case that many IPs within the instance were solvable, and thus many Pareto points identified for the Chief Residents to consider, but at least one IP required more than thirty minutes to solve and therefore we terminated the algorithm. We note that 14 of the 15 instances that timed out come from May 2014 and June 2014, which were difficult months to schedule due to a lower than usual number of residents assigned to the ED and thus less flexibility in meeting the scheduling requirements. For the remaining 213 *solvable* instances, 62 solved to completion (i.e., generated the complete Pareto frontier) in under one minute, the next 92 in under 10 minutes, and the next 42 in under 1 hour. For a complete summary of run times, see Figure 3.2, with additional details to be found in Table B.1 of the appendix.



Histogram of CPU times to solve all 243 instances of the monthly scheduling problem. Infeasible instances and instances that were not solved with a 30 minutes time limit imposed on individual IPs are depicted as separate bars.

Figure 3.2: Histogram of CPU times

Finally, we evaluate the relative impact of the different levels of flexibility on run time. In Table 3.2, we see the average run time *per Pareto point generated*, as a function of the 9 combinations of flexibility characteristics. Note that we report average time per point in recognition of the fact that different instances yield different-sized Pareto frontiers. We observe that the slowest run times are found for S1/V3, which yields the most tightly-constrained problem instances with heavy vacation requests and thus the associated IPs are the most challenging.

Time Per Point (sec)	V1	V2	V3
S1	22.13	23.69	37.22
S2	14.04	14.81	22.81
S3	12.07	12.49	16.73

Table 3.2: CPU time per Pareto points for 9 levels of flexibility

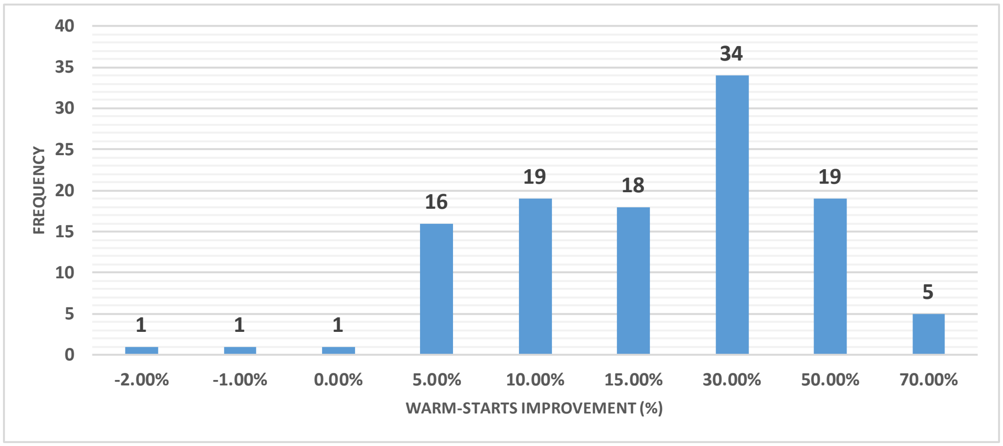
3.4.3 Contributions from Warm Starts Strategy

As noted in Section 3.3.4, our problem easily lends itself to the use of warm starts, because at each iteration we are solving an IP for which the preceding iteration's solution is still feasible. In this section, we discuss the impact on run time of using these warm starts.

Of the 228 *feasible* instances described above, there are fourteen which cannot be solved to completion, given a 30-minute time limit on the individual IPs, independent of whether or not the warm start is applied. There are three instances which can only be solved if warm starts are used. Interestingly, there is also one instance which cannot be solved using warm starts, but can be solved to completion when warm starts are not applied. For more details about these 18 instances, see Appendix Table B.2.

For the 210 instances that can be solved either with or without the use of warm starts, we observe that the overhead of implementing the warm start outweighs any benefits for problems that solve very quickly — for those 96 instances that can be solved in under 3 minutes without a warm start, there is no significant benefit in reading and applying the warm start and in some cases the run time even becomes slower. For all 96 instances, the difference (plus or minus) between the run time with and without warm starts is at most 30 seconds.

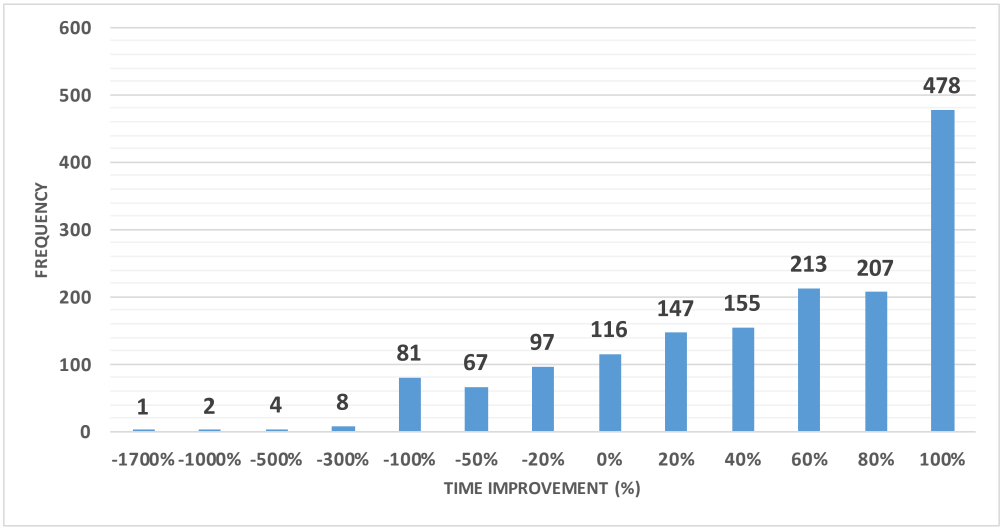
For the remaining 114 instances, i.e., those that require more than three minutes of run time, there are three that actually perform faster without the use of warm starts. However, the time loss is not significantly different since it just gets 2.72% slower even in the worst case. For the remaining 111, the benefits range from 0.11% to 65.75% reduction in time. The largest absolute change was 6 hours 12 minutes. The average time savings was 33.44%. Figure 3.3 summarizes the comparisons over all 114 instances. Further information can be found in Figure B.1 in the Appendix.



Histogram of the ratio of the CPU time saved by the use of warm starts to without the use of warm starts for 114 instances taking more than 3 minutes to find the Pareto frontier without the use of warm starts. (i.e., Warm-Starts Improvement (%) $= \frac{A-B}{A}$ where $A =$ CPU time without warm starts, $B =$ CPU time with warm starts)

Figure 3.3: Histogram of CPU time improvement ratio of warm starts to no warm start for the non-trivial instances

Given the 30-minute time limit for individual IP runs, there exists one instance which can be solved without a warm start, but cannot be solved using warm starts because warm starts do not always save CPU time in the IP instance level. Figure 3.4 presents a histogram of time improvement ratio of warm starts over all non-trivial comparable IPs, where comparable IP is defined as an IP instance that can be solved both with and without warm starts and non-trivial IP means an IP requiring more than 60 seconds. In Figure 3.4, the warm starts can make run time extremely slower than without the use of warm starts even though it rarely happens statistically. Thus, it is possible to terminate by the time limit imposed on individual IP runs even if total time could be shorter.



Histogram of CPU time ratio of warm starts to no warm start for total 1576 comparable individual IPs taking greater than or equal to 60 seconds. The percentage in x-axis represents the ratio of time saved by warm starts to no warm start CPU time. Here, a comparable individual IP means an IP that is solved by both with and without warm starts.

Figure 3.4: Histogram of CPU time improvement ratio of warm starts to no warm start for the non-trivial IPs.

3.4.4 Size of Pareto Frontier

In addition to assessing the run time of our algorithm, we also sought to understand the size of the resulting Pareto Frontiers found in realistic problem instances. There are 214 problem instances that we were able to solve to completion, either with or without the use of warm starts. For each, we counted the number of Pareto points. These are summarized in Figure 3.5.

64 instances have five or fewer Pareto points, the next 26 have between 6 and 10 Pareto points, and the next 45 have between 11 and 20 Pareto points. We observe that 17 instances have more than 100 Pareto (with a maximum of 387). However, we see that 14 instances out of these 17 are instances in category V3, i.e., they have the largest number of vacation requests. This number is actually unrealistic in real-world practice but was considered to test the boundaries of performance.

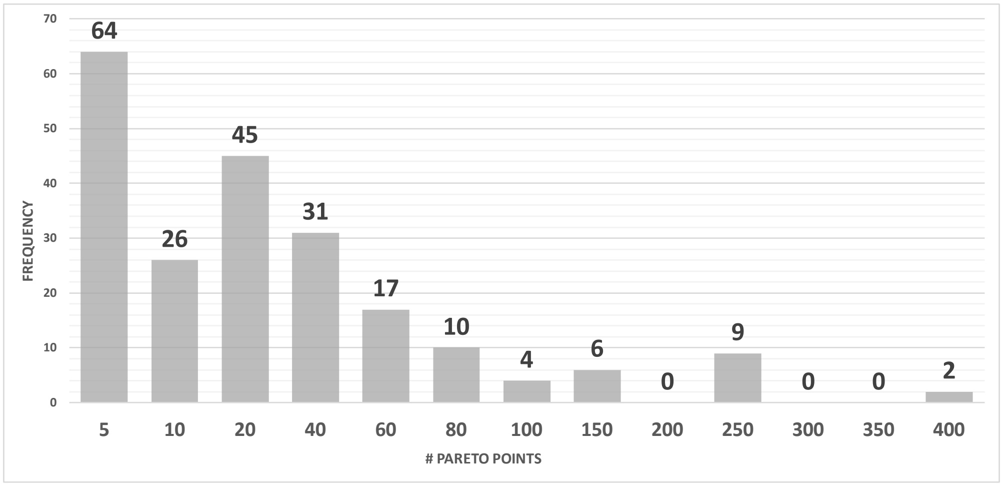


Figure 3.5: Histogram of the number of Pareto points

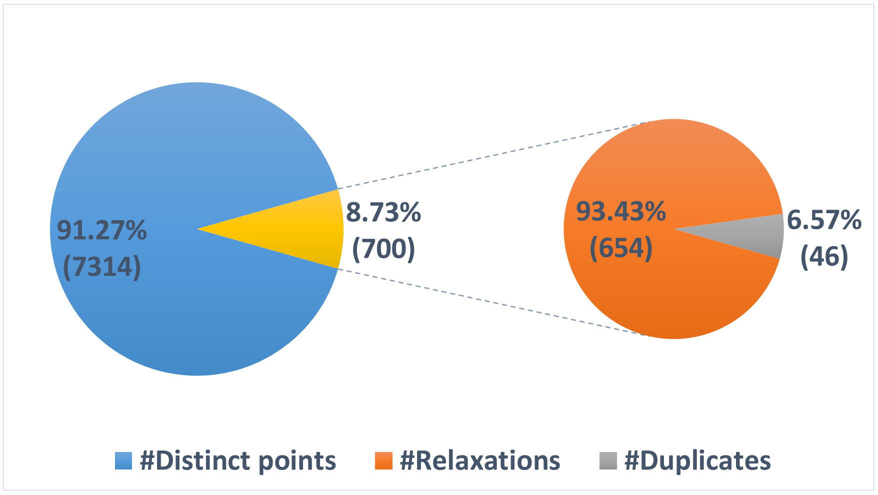
3.4.5 Efficiency

As seen in [110], the number of IPs solved by the recursive method is bounded by $\mathcal{O}(\mid \mathcal{P} \mid^n)$ where $\mid \mathcal{P} \mid$ is the size of the Pareto frontier and n is the number of metrics. This implies that the recursive method could solve duplicate IPs that yield the same Pareto point repeatedly throughout the algorithm. However, we have found that the inefficient calculations rarely happened in our problem instances.

Table 3.6 shows that 700 iterations are duplicated out of total 8014 iterations. Since 654 iterations out of 700 are saved by the relaxation improvement, only 46 iterations are inefficient calculations. Therefore, this implies that our approach is very efficient to find the Pareto frontier for the PEDS problem.

3.4.6 Case Study

We conclude this section with a discussion of how Chief Residents might use the output from our approach. Note that for (64+26+45=) 135 instances, there were fewer than 20 Pareto points as shown in Figure 3.5, which in our experience is a small enough number for the Chiefs to evaluate explicitly. At the other extreme, there were 21 instances with more than 80. We view this number



There were total 8014 iterations to find Pareto points. Among them, 700 iterations was repeatations that yield the duplicate Pareto points that has been solved before. The number of relaxations is the number of trials that are reduced by the relaxation improvement. Only 46 out of 700 repeatations yielded duplicated calculations inefficiently.

Figure 3.6: The number of relaxations and duplicates in the sequential method for the PEDS

as too large to directly evaluate. However, we have observed that most of these instances are generated by V3, exaggerated vacation requests considered to test the boundaries of tractability, and these extreme cases rarely happen in practice. Thus, we focus on the instances with more than 20 but at most 80 Pareto points.

In our collaborations with several Chief Residents, we have observed that while one metric is never fully dominant, nonetheless they typically have a rough prioritization of the metrics. For example, suppose that there are four metrics and the vector $(M1, M2, M3, M4)$ represents the value of four metrics in order. If the first metric was the most important and there are two candidates $P1 = (2, 3, 5, 7)$ and $P2 = (3, 2, 4, 6)$, Chief Residents would prefer $P1$ over $P2$ since they would be willing to give up a little from each of the other metrics to improve the first metric. However, they would not choose $(2, 30, 50, 70)$ over $P2$. We call this *weak* priority, where there exists a highest-priority metric, but it does not strictly dominate in all cases, as in the example above.

In the PEDS problem, what we have observed is that the Chiefs typically begin by reviewing only those schedules for which the highest-priority metric has its best possible value. If they find a high-quality schedule within this space, they are satisfied. Alternatively, they may also choose to review those schedules for which the highest-priority metric achieves its second-best value and quickly evaluate those schedules as well to see if significant benefit can be gained with minor sacrifice of the first metric.

Table 3.3 presents the number of Pareto points for each highest-priority metric over (31+17+10=) 58 instances with more than 20 but at most 80 Pareto points. We observe that, even though the total number of Pareto points may be as high as 80 in these instances, the total number of solutions relative to a single metric's best possible value is rarely more than 20 and thus manageable for the Chiefs to evaluate.

3.5 Conclusions

In this paper, we present an IP-based approach that generates the exhaustive set of Pareto-dominant schedules from which the Chief Residents can choose and demonstrated the tractability and practicality of this new approach to solving a real-world residency scheduling problem. This approach provided substantial benefits to the chief residents, providing them with a high-quality set of schedules to evaluate in a short period of time, and facilitating their qualitative decision making.

The contributions of this research are: (a) we suggest a way to handle ill-defined preferences between conflicting objectives; (b) the first application of a recursive method to generate all Pareto-dominant schedules; (c) showing tractability and usability of it; (d) a strategy to use a warm start.

Index	Year	Month	Levels	#Pareto Points	DVR			UFS			PCC			BSP		
					TOP1	TOP2	TOP3	TOP1	TOP2	TOP3	TOP1	TOP2	TOP3	TOP1	TOP2	TOP3
6	2012-2013	July	S2V3	38	15	27	34	15	26	34	9	17	26	12	21	28
9	2012-2013	July	S3V3	38	15	27	34	15	26	34	9	17	26	12	21	28
10	2012-2013	August	S1V1	48	6	13	18	17	34	44	4	8	11	24	28	39
11	2012-2013	August	S1V2	48	6	13	18	17	34	44	4	8	11	24	28	39
13	2012-2013	August	S2V1	48	6	13	18	17	34	44	4	8	11	24	28	39
14	2012-2013	August	S2V2	48	6	13	18	17	34	44	4	8	11	24	28	39
16	2012-2013	August	S3V1	48	6	13	18	17	34	44	4	8	11	24	28	39
17	2012-2013	August	S3V2	48	6	13	18	17	34	44	4	8	11	24	28	39
24	2012-2013	September	S2V3	42	2	5	12	4	10	18	1	5	10	7	16	24
27	2012-2013	September	S3V3	36	2	5	12	4	10	18	1	3	6	3	10	18
30	2012-2013	October	S1V3	22	6	12	18	12	19	22	6	12	18	12	19	22
55	2012-2013	April	S1V1	70	10	25	40	20	40	55	5	14	26	19	24	40
56	2012-2013	April	S1V2	24	3	6	9	11	18	21	5	9	12	10	11	18
57	2012-2013	April	S1V3	48	6	12	18	19	33	42	5	14	21	18	19	33
58	2012-2013	April	S2V1	70	10	25	40	20	40	55	5	14	26	19	24	40
59	2012-2013	April	S2V2	24	3	6	9	11	18	21	5	9	12	10	11	18
60	2012-2013	April	S2V3	48	6	12	18	19	33	42	5	14	21	18	19	33
61	2012-2013	April	S3V1	70	10	25	40	20	40	55	5	14	26	19	24	40
62	2012-2013	April	S3V2	24	3	6	9	11	18	21	5	9	12	10	11	18
63	2012-2013	April	S3V3	48	6	12	18	19	33	42	5	14	21	18	19	33
100	2013-2014	September	S1V1	50	8	17	26	18	35	50	7	15	24	17	34	35
101	2013-2014	September	S1V2	24	6	12	18	12	20	24	3	8	14	12	20	24
103	2013-2014	September	S2V1	37	7	14	21	13	24	33	7	14	21	20	22	36
106	2013-2014	September	S3V1	37	7	14	21	13	24	33	7	14	21	20	22	36
111	2013-2014	October	S1V3	37	3	10	19	2	6	12	1	4	8	2	7	13
114	2013-2014	October	S2V3	36	3	10	19	2	6	12	1	3	7	2	6	12
117	2013-2014	October	S3V3	36	3	10	19	2	6	12	1	3	7	2	6	12
127	2013-2014	March	S1V1	29	14	29	29	5	11	17	1	4	9	9	19	29
128	2013-2014	March	S1V2	29	14	29	29	5	11	17	1	4	9	9	19	29
172	2014-2015	August	S1V1	24	4	9	14	4	9	14	1	3	6	14	19	22
173	2014-2015	August	S1V2	24	4	9	14	4	9	14	1	3	6	14	19	22
174	2014-2015	August	S1V3	80	10	24	39	17	35	53	1	4	12	25	47	63
175	2014-2015	August	S2V1	24	4	9	14	4	9	14	1	3	6	14	19	22
176	2014-2015	August	S2V2	24	4	9	14	4	9	14	1	3	6	14	19	22
177	2014-2015	August	S2V3	60	11	23	35	14	28	41	1	4	9	21	36	43
178	2014-2015	August	S3V1	24	4	9	14	4	9	14	1	3	6	14	19	22
179	2014-2015	August	S3V2	24	4	9	14	4	9	14	1	3	6	14	19	22
180	2014-2015	August	S3V3	60	11	23	35	14	28	41	1	4	9	21	36	43
195	2014-2015	October	S2V3	54	8	20	31	18	34	45	12	22	32	18	30	42
198	2014-2015	October	S3V3	55	8	20	30	19	36	47	14	23	33	18	29	41
199	2014-2015	November	S1V1	22	3	8	13	9	18	22	5	10	15	11	14	20
200	2014-2015	November	S1V2	35	4	10	17	13	26	35	8	16	24	12	24	27
201	2014-2015	November	S1V3	31	3	7	11	10	20	31	6	12	18	12	21	26
202	2014-2015	November	S2V1	22	3	8	13	9	18	22	5	10	15	11	14	20
203	2014-2015	November	S2V2	39	5	12	20	13	26	35	9	18	27	15	28	31
204	2014-2015	November	S2V3	43	3	9	16	14	28	38	9	18	27	15	28	32
205	2014-2015	November	S3V1	22	3	8	13	9	18	22	5	10	15	11	14	20
206	2014-2015	November	S3V2	39	5	12	20	13	26	35	9	18	27	15	28	31
207	2014-2015	November	S3V3	52	3	9	16	17	34	46	9	18	27	19	35	39
217	2014-2015	April	S1V1	61	3	11	24	17	39	61	1	5	13	9	21	24
218	2014-2015	April	S1V2	61	3	11	24	17	39	61	1	5	13	9	21	24
220	2014-2015	April	S2V1	75	6	16	31	19	39	57	4	13	25	17	27	42
221	2014-2015	April	S2V2	75	6	16	31	19	39	57	4	13	25	17	27	42
223	2014-2015	April	S3V1	75	6	16	31	19	39	57	4	13	25	17	27	42
224	2014-2015	April	S3V2	75	6	16	31	19	39	57	4	13	25	17	27	42
237	2014-2015	June	S1V3	21	6	12	17	11	18	21	5	10	13	8	9	14
240	2014-2015	June	S2V3	24	7	14	20	11	18	21	6	12	15	11	12	17
243	2014-2015	June	S3V3	24	7	14	20	11	18	21	6	12	15	11	12	17

The number of Pareto points for the highest-priority metrics over instances with more than 20 but at most 80 Pareto points. The three columns TOP1, TOP2, and TOP3 under the highest-priority metrics represents the number of Pareto points when we consider the best possible value only, or the best and the second-best value together, or the best and the second and the third-best value respectively.

Table 3.3: The number of Pareto points reduced by the highest-priority metrics

CHAPTER 4

Vehicle Routing Problem

4.1 Introduction

In this chapter, we present models and algorithms to solve the time-constrained heterogeneous vehicle routing problem (TCHVRP). In this variant of the classical VRP, first posed by Dantzig and Ramser [67], we allow vehicles to vary not only by route time limit (in our case, corresponding to a limit on total travel and service time), but by cost on each arc as well. This research is motivated by the gradual introduction of hybrid vehicles into existing fleets [81]. In such cases, one vehicle (e.g. with a traditional combustion engine) might get better mileage (and thus have lower cost) on an arc primarily comprised of highway driving, whereas a hybrid vehicle might dominate in cost over an arc primarily comprised of city driving. Likewise, highway distances can be longer between two points than a city route, but travel times shorter due to less traffic and higher speeds. Thus, of particular importance to this research is the fact that we do not assume Pareto dominance of one vehicle type over another. Nor do we assume any correlation between arc distances and arc costs or arc times.

We consider two ways to model the TCHVRP, one a vehicle flow formulation and the other a set partitioning formulation. We find the explicit vehicle flow formulation to be computationally impractical for all but very small problem instances. We therefore focus on a set partitioning formulation, where *delayed column generation* [74] can be used to address the very large number of columns. We introduce and analyze a dynamic programming (DP) approach and different relaxations for generating these columns effectively in the pricing problem.

4.2 Time-Constrained Heterogeneous Vehicle Routing Problem (TCHVRP)

4.2.1 Problem Description

The classical VRP is to find an optimal (i.e. minimum-cost) set of routes for a fleet of vehicles so as to serve a given set of customers; the TSP is a special case of VRP, in which only one vehicle must serve all customers [121]. Many other variants of the VRP have been considered, including: CVRP, in which the vehicles have limited capacity [80, 122, 123], DVRP, where all vehicles have upper bounds on the length of the route in terms of mileage or time [124], multi-depot VRP (MDVRP), in which there are different depots that can be used for each vehicle [125], VRP with time windows (VRPTW), where customers have time windows within which the deliveries must be made [126, 127], and HVRP, in which there are different types of vehicles characterized by different capacities and costs [72].

Within HVRP, problems can be further classified. For example, some have finite numbers of vehicles for each type and some are unlimited. Some have fixed costs for each vehicle and some do not. In some cases, the capacity is constant across all vehicle types and in some cases it varies across types. Furthermore, arc-costs may be constant or may vary by vehicle type. These problems are further described, and the literature reviewed, in Baldacci, Battarra, and Vigo [96]. In addition, Laporte et al. [128] considered an extended version of DVRP involving both capacity and distance constraints simultaneously.

We consider a combination of HVRP and DVRP, which we label the Time-Constrained Heterogeneous Vehicle Routing Problem (TCHVRP). In this variant, the cost and travel time of any given arc may vary by vehicle type within a heterogeneous fleet. In particular, we make no assumptions about Pareto dominance across vehicles. Thus, one vehicle may be the least-cost option on certain arcs but the most costly on others. Similarly, one vehicle may be fastest over some arcs, but slowest over others. Furthermore, we do not assume that cost and time are correlated. This allows us to incorporate characteristics of mixed fleets that combine vehicles with both traditional combustion and hybrid engines.

Specifically, TCHVRP is defined as follows:

- We are given a fixed depot and a known set of customers.
- We are given a set of vehicle types, with a finite number of vehicles within each type.
- We are given arcs connecting each customer with the depot and each pair of customers. Each vehicle type/arc pair has a given cost. There is no fixed cost per vehicle.
- Each vehicle type has its own route time limit on the amount of work that vehicles of that type can perform. Each vehicle type/arc pair has a given time duration, and there is a given time associated with servicing each customer. Note that this resource (time) is consumed over the arcs as well as at the nodes (i.e. customers).
- The goal is to find the minimum-cost assignment of vehicle types to routes such that each customer is visited exactly once, no vehicle exceeds its time limit, and we do not use more vehicles of a given type than are available.

4.2.2 Data Sets

Throughout the paper, we present computational experiments to assess the performance of different approaches to solving TCHVRP. Our computational experiments are loosely based on the data sets of Golden et al. [72]. Specifically, we use the network topology of these data sets, defined by coordinates in a Euclidean space for each node in the network. Our arc costs and times cannot be drawn from the data sets, however, as it is the variation in these parameters that we are specifically exploring.

Our data sets range in size from 20 nodes to 35 nodes, with exhaustive sets of arcs connecting all node pairs. These correspond to Problem Instances 3, 4 and 15 through 18 of Golden et al. [72]. Specifically, we use the first 20 nodes from Problem Instances 3 and 4, the first 25 nodes from Problem Instances 17 and 18, the first 30 nodes from Problem Instances 15 and 16, and the first 35 nodes from Problem Instances 17 and 18.

We consider three different cases of problem instances derived from these original data sets. In the first case, which serves as a base case, we construct instances where there is only one vehicle type and arc costs and times are fully correlated with distance. Specifically, we define a constant cost per mile and a constant speed, and apply these to all arcs in the network for all vehicle types, using the Euclidiean distance between a given pair of nodes as their arc length.

In the second case, we allow cost per mile and speed to vary by vehicle type, but arc costs and speeds are still fully correlated with distance, and there is thus Pareto dominance with respect to cost, i.e. whichever vehicle type is least-cost on one arc will be least-cost on all arcs.

In the third case, we consider the case of interest, in which costs and times are neither Pareto-dominant nor fully correlated with each other (and thus, implicitly, with the arc distance), although we do recognize that there is at least some degree of correlation to be reasonably expected. To generate arc costs and times, we again start with the same arc lengths between each pair of customers as in the first two cases. We then assign each vehicle type a baseline cost per mile and a baseline speed, but we also randomly generate a perturbing error factor for each arc to avoid Pareto dominance and full correlation between cost and time. Given that we are randomly generating the perturbation factors, we create ten instances for each set of network topologies in this case. [See Appendix Table C.1 for full details.]

Finally, we assume three types of vehicles for problem instances in the second and third cases. For 20-node instances, we assume two vehicles of each type; for 25-node instances, we assume three vehicles of each type; and for 30- and 35-node instances, we assume four vehicles of each type. For the sake of exposition, we assign the same service time (thirty minutes) to each customer and the same route time limit constraint (eight hours) to each vehicle type. It is trivial, however, to specify vehicle- and customer-specific values within our proposed model.

All computational experiments were performed on an Intel Xeon 3.20 GHz processor with 32 GB of memory using CPLEX 12.2 C++ API with an optimality gap of 0.01%.

4.3 Arc-Based Model

As with other variants of VRP, TCHVRP can be formulated as an arc-based network flow model, using subtour elimination constraints such as those found in Golden et al. [72], which in turn builds on the work of Miller, Tucker, and Zemlin (MTZ) for the TSP [129]. Such an approach to TCHVRP, however, suffers from poor computational performance, as we will demonstrate below. Thus, this section motivates us to instead pose a column generation approach to solving a path-based formulations of the TCHVRP.

4.3.1 Vehicle Flow Formulation

<u>**Notation**</u>

Parameters and Sets

N	set of customers in the network, where index 0 represents the depot
T	set of vehicles types
c_{ij}^t	cost to travel from customer i to customer j for vehicle type t,$\forall i, j \in N, \forall t \in T$
d_{ij}^t	travel time from customer i to customer j for vehicle type t, $\forall i, j \in N, \forall t \in T$
M_t	number of available vehicles of type t, $\forall t \in T$
Q_t	route time limit for vehicle type t, $\forall t \in T$
L_i	service time at customer i, $\forall i \in N \setminus 0$

Variables

x_{ij}^t	binary variable that takes value 1 if a vehicle of type t is assigned to travel between customers i and j, $\forall i, j \in N, \forall t \in T$
y_i^t	cumulative travel and service time up through customer i for vehicle type t, $\forall i \in N, \forall t \in T$

Arc-Based Model (ABM):

$$\min \sum_{t\in T}\sum_{i\in N}\sum_{j\in N} c_{ij}^t x_{ij}^t \tag{4.1a}$$

$$\text{subject to: } \sum_{t\in T}\sum_{i\in N} x_{ij}^t = 1 \quad \forall j \in N \setminus 0 \tag{4.1b}$$

$$\sum_{i\in N} x_{ij}^t - \sum_{i\in N} x_{ji}^t = 0 \quad \forall j \in N, \forall t \in T \tag{4.1c}$$

$$\sum_{j\in N} x_{0j}^t \le M_t \quad \forall t \in T \tag{4.1d}$$

$$Q_t x_{ij}^t + (d_{ij}^t + L_j)x_{ij}^t + y_i^t \le y_j^t + Q_t \quad \forall i,j \in N, i \ne j, \forall t \in T \tag{4.1e}$$

$$y_j^t + d_{j0}^t x_{j0}^t \le Q_t \quad \forall j \in N \setminus 0, \forall t \in T \tag{4.1f}$$

$$x_{ij}^t = \begin{cases} \in \{0,1\} & , \quad i \ne j, \forall t \in T \\ 0 & , \quad i = j, \forall t \in T \end{cases} \tag{4.1g}$$

$$y_j = \begin{cases} \in \mathbb{R}^+ & , \quad j \in N \\ 0 & , \quad j = 0 \end{cases} \tag{4.1h}$$

The objective (4.1a) minimizes the total routing cost, which is the sum of the costs associated with each arc used. Constraint set (4.1b) ensures that exactly one vehicle type and one incoming arc is assigned to customer j. Constraint set (4.1c) specifies flow conservation at each customer. Constraint set (4.1d) specifies that the number of arcs out of the depot (and thus the number of routes) assigned to a vehicle type does not exceed the available number of vehicles of that type. Constraint set (4.1e) provides the subtour elimination constraints. The variable y_i^t is the accumulated time associated with whatever vehicle type t services customer i. Constraint set (4.1e) defines the relationship between any two customers that are sequential on a route to have respective y values that differ by at least the travel and service time associated with moving between them. This ensures that a subtour cannot exist, otherwise the y variable associated with any customer appearing in a subtour would ultimately be required to be strictly greater than itself. In particular, when $x_{ij}^t = 1$, some vehicle type t travels from customer i to customer j. Therefore, y_j^t must be at least equal to y_i^t plus the time between them and the service time at customer j ($(d_{ij}^t + L_j)x_{ij}^t$). Conversely, when $x_{ij}^t = 0$, there is no known relationship between y_i^t and y_j^t. The equation reduces to $y_i^t \leq y_j^t + Q_t$ which will also hold because Q is the limit on accumulated time for any single vehicle, i.e. any single route. Constraint (4.1f) ensures that the time incurred by a vehicle type t through servicing of customer j, plus the time that it would take to return to the depot if this were the last customer on the route, cannot exceed the total time limit for vehicles of the associated type. This enforces the capacity constraint.

4.3.2 Computational Experiments and Analysis

The arc-based model (ABM) is difficult to solve in practice, both because it is highly fractional and because the LP relaxation provides an inherently weak lower bound on the optimal objective value. To demonstrate this, we evaluated a randomly chosen subset of our non-Pareto data instances, limiting each run to at most two hours of run time. Tables 4.1 and 4.2 show our computational results. Specifically, for each of the four instances considered, Table 4.1 provides the number of nodes in the network, the realization number (recall that for each of the original networks, we randomly generated 10 sets of arc distances, costs, and times), the number of constraints, and the number of variables. This table also includes information about the branch-and-bound results after two hours of run time. Specifically, we see the number of branch-and-bound nodes solved, the number of pending nodes remaining in the branch-and-bound tree, and the optimality gap at the end of two hours. Table 4.1 shows that ABM failed to solve any of the instances to provable optimality (the optimality tolerance is 0.01%) within the two hour time limit, and in the case of the 35-node instances, could not even find an integer-feasible solution. In addition, the non-trivial optimality gaps and remaining high numbers of pending nodes suggest that the algorithm is far from terminating at the two-hour time limit.

Table 4.2 enables us to compare the run times and best objective values for three variations of the problem — the LP relaxation of the integrality constraints (ABM-LPR), the original problem

Data Set				ABM (2hours)		
#Nodes	Realization	Number of constraints	Number of variables	Number of nodes solved	Number of pending nodes	Optimality gap
20	8	~500	~1200	914,203	789,919	4.66%
25	5	~750	~1900	314,918	305,205	12.26%
30	7	~1050	~2700	190,431	188,429	13.27%
35	5	~1400	~3700	73,857	72,455	N/A

Table 4.1: Computational results after two hours (ABM)

with a two-hour limit on run time, and the original IP with the relaxation of the subtour elimination constraints (ABM-SER). As Table 4.2 indicates, the run time of the LP relaxation is close to 0, suggesting that the problem size is not the source of computational challenge. The LP solutions, however, are highly fractional, as demonstrated in Table 4.1 by the large number of branch-and-bound nodes solved and the nearly equivalent number of pending nodes yet to be solved. Furthermore, the optimality gap at termination is quite weak, with three of the instances having gaps between 4.66% and 13.27% and one of the instances unable to find a single integer-feasible solution within the two-hour time limit.

The key to this fractionality appears to be the subtour elimination constraints. When we remove these, the remaining IP solves very quickly (almost as fast as ABM-LPR). As seen in Table 4.2, however, the objective value for ABM-SER is even lower than the objective value for the LP relaxation, providing an even worse optimality gap.

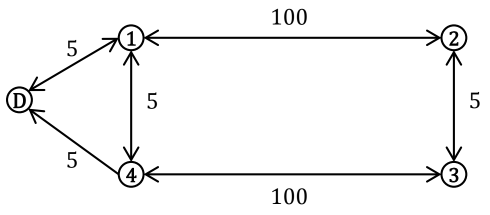
Data Set		ABM-LPR		ABM (2hours)		ABM-SER	
#Nodes	Realization	Run Time (sec)	Objective Value	Run Time (sec)	Objective Value	Run Time (sec)	Objective Value
20	8	< 1	375.46	7200	477.91	< 1	369.51
25	5	< 1	464.79	7200	544.89	1	462.04
30	7	< 1	472.99	7200	544.10	< 1	467.91
35	5	< 1	532.77	7200	613.84	< 1	526.73

Table 4.2: Optimal cost of relaxations (ABM)

To help understand why ABM-LPR and ABM-SER provide such weak lower bounds, consider the example from Figure 4.1. In this example, we see a simple network with four nodes (plus the depot). Arc costs are depicted; for the sake of exposition, we assume all vehicles are identical and have infinitely long time limits. In this instance, the optimal IP solution is to follow the path D-

1-2-3-4-D, with each corresponding arc having flow of one, and an objective value of 215. When the subtour elimination constraints are relaxed, however, it is valid to assign one unit of flow to the arcs in path D-1-4-D and likewise to the arcs in the path 2-3-2. This reduces the objective value by 88 percent, from 215 to 25.

Finally, when the subtour constraints are included but the integrality requirements are relaxed, we can assign α units of flow to each arc in the path D-1-2-3-4-D (the solution to ABM), and $1-\alpha$ units to the arcs in the path D-1-4-D and 2-3-2 (the solution to ABM-SER), i.e. ABM-LPR is a convex combination of ABM and ABM-SER, where α approaches 0 as Q goes to infinity.



Models	Paths	Cost
ABM	r_{IP}: D-1-2-3-4-D (1)	$5+100+5+100+5 = 215$
ABM-SER	r_{SER}: D-1-4-D, 2-3-2 (1)	$(5+5+5)+(5+5) = 25$
ABM-LPR $(0 \leq \alpha \leq 1)$	r_{LPR}: D-1-2-3-4-D (α) D-1-4-D , 2-3-2 ($1-\alpha$)	$\alpha \times 215 + (1-\alpha) \times 25$ $= \alpha \times r_{IP} + (1-\alpha) \times r_{SER}$

Figure 4.1: Example of weak lower bounds by relaxations (ABM)

4.4 Path-Based Model

Motivated by the computational experiments described in Section 4.3, we seek an approach that is not hampered by the need for subtour elimination. In this section, we introduce a path-based model that, by explicitly constructing paths as an input to the model, eliminates the need for such constraints.

4.4.1 Set Partitioning Formulation

Notation

Parameters and Sets

N	set of customers in the network, where index 0 represents the depot
T	set of vehicles types
R_t	set of all feasible routes (i.e. paths) for vehicle type $t \in T$
c_r^t	travel cost of route r for vehicle type t, $\forall t \in T, \forall r \in R_t$
δ_{ir}^t	binary coefficient that takes value 1 if customer i belongs to route r for vehicle type t, else $0, \forall i \in N \setminus 0, t \in T, r \in R_t$
M_t	number of available vehicles of type t, $\forall t \in T$
Q_t	time limit for vehicle type t, $\forall t \in T$
L_i	service time at customer i, $\forall i \in N \setminus 0$

Variables

x_r^t	binary variable that takes value 1 if route r is assigned to a vehicle of type t, else 0, $\forall t \in T, \forall r \in R_t$

<u>Path-Based Model (PBM):</u>

$$\min \sum_{t \in T} \sum_{r \in R_t} c_r^t x_r^t \tag{4.2a}$$

$$\text{subject to: } \sum_{t \in T} \sum_{r \in R_t} \delta_{ir}^t x_r^t = 1 \quad \forall i \in N \setminus 0 \qquad (\pi_i) \tag{4.2b}$$

$$\sum_{r \in R_t} x_r^t \leq M_t \quad \forall t \in T \qquad (\mu_t) \tag{4.2c}$$

$$x_r^t \in \{0, 1\} \quad \forall t \in T, \forall r \in R_t \tag{4.2d}$$

The objective (4.2a) minimizes the total routing cost. Constraint set (4.2b) requires that each customer must be covered by exactly one route. Constraint set (4.2c) specifies that at most M_t routes are assigned to vehicles of type $t \in T$. In addition, we associate dual variables π_i and μ_t with (4.2b) and (4.2c), respectively.

4.4.2 Heuristic Column Generation

The path-based model (PBM) contains too many variables to solve explicitly except for very small problem instances. We therefore propose to use *column generation* [130, 131] to solve the LP relaxation of PBM (PBM-LPR). Specifically, we begin with an initial subset of the feasible columns (i.e. routes) generated by the sweep algorithm [84], which we call the *restricted master problem* (RPBM-LPR). We then use a *pricing problem* to identify promising new routes to add to RPBM-LPR, based on the dual values of the current optimal solution. If such columns (i.e. routes) are found, they are used to augment RPBM-LPR and the process repeats. An optimal solution to the LP relaxation is guaranteed when no new negative reduced cost routes can be found.

Reduced Cost:

For a given feasible route $r_t \in R_t$ where γ^t_{ijr} is a binary parameter specifying whether arc (i, j) is included in route r_t, the corresponding reduced cost equation is:

$$\begin{aligned}
RC^t_r &= c^t_r - \sum_{i \in N \setminus 0} \pi_i \delta^t_{ir} - \mu_t \\
&= \sum_{i \in N} \sum_{j \in N} c^t_{ij} \gamma^t_{ijr} - \sum_{i \in N \setminus 0} \pi_i (\sum_{j \in N} \gamma^t_{ijr}) - \mu_t \\
&= \sum_{i \in N} \sum_{j \in N} c^t_{ij} \gamma^t_{ijr} - \sum_{i \in N} \pi_i (\sum_{j \in N} \gamma^t_{ijr}) - \mu_t \\
&= \sum_{i \in N} \sum_{j \in N} (c^t_{ij} - \pi_i) \gamma^t_{ijr} - \mu_t
\end{aligned}$$

where we define $\pi_0 = 0$ so as to combine terms in the final representation.

To find a negative reduced cost column for vehicles of type t, or to prove that no such columns exist, we can formulate the following optimization problem.

Pricing Problem:

$$\min \sum_{i \in N} \sum_{j \in N} (c_{ij}^t - \pi_i)\gamma_{ijr}^t - \mu_t \tag{4.4a}$$

$$\text{subject to: } r_t \in R_t \tag{4.4b}$$

For the remainder of this section, we focus primarily on solving PBM-LPR and, in particular, the pricing problem. This problem can be posed as an elementary shortest path problem with resource constraints (ESPPRC) on graphs with negative cost cycles and it has been shown that it is strongly NP-hard [132]. Thus, when we solve the integer version of the problem, we do so heuristically. That is, for the simplicity of implementation, we solve the LP to optimality and then find the best IP solution relative to the current set of columns in the restricted master. In practice, it would be necessary to generate new columns at each subsequent nodes of the branch-and-bound tree in order to ensure a provably optimal solution.

4.5 Dynamic Programming Approaches to solving the Pricing Problem

The key challenge in solving PBM-LPR is in solving the pricing problem, i.e. in generating candidate routes (negative reduced cost pivot variables). In theory, this problem ((4.4a) and (4.4b)) could be formulated as a MIP. However this will result in the same difficulties observed in the arc-based formulation, such as fractionality in the LP relaxation and a weak lower bound, due to the sub-tour elimination (4.1e) and time limit constraints (4.1f), which must now be shifted into the pricing problem. We therefore instead take a dynamic programming (DP) approach in which, by constructing the routes dynamically, we naturally avoid sub-tours. Throughout the remainder of this section, we present and compare several DP algorithms for solving the pricing problem.

4.5.1 Elementary Shortest Path Problem with Resource Constraints (ESPPRC)

The pricing problem for PBM-LPR identifies minimum-cost *elementary paths* (i.e. paths without any embedded cycles) that do not violate some sort of resource constraint. In our case, the resource is time (which is consumed over both arcs and nodes), and the cost is the reduced cost associated with the route when included in the restricted master. That is, we consider the true cost (which is the sum of all the arc costs) minus the dual value associated with each node minus the dual value associated with the specific vehicle type. In particular, one reduced cost problem can be solved for each vehicle type, in which the corresponding costs (true and dual) are assigned to individual arcs within the network.

Feillet et al. [133] were the first to propose an exact dynamic programming algorithm for solving the pricing problem of the VRPTW where paths with embedded cycles are forbidden. Other refinements for the DP algorithm have also been suggested by Chabrier [134], Righini and Salani [135, 136] and Lozano et al. [137]. Chabrier [134] embeded ESPPRC into a branch-and-price scheme for VRPTW and improved the quality of bounds by cutting planes and several other improvements. Righini and Salani [135, 136] proposed resource-based bounding, bidirectional search and a decremental state-space relaxation. Lozano et al. [137] introduced a recursive algorithm with novel pruning strategies and tested the proposed algorithm by solving the linear relaxation of the VRPTW at the root node via column generation approach. While DP can be a good candidate approach for solving the pricing problem for some VRPs [138], the exponential rise in the state-space can be burdensome in other cases. For example, the success of Feillet et al. [133] depends on the use of narrow time windows to facilitate pruning. On the other hand, when time windows are too wide, the computational performance suffers significantly.

In TCHVRP, we encounter computational issues associated with the limited opportunity to prune. Although we can use the vehicle time limits and the fact that nodes cannot be repeated to prune, these are not sufficient. Ideally, we also want to prune whenever two partial paths meet at the same node but with different costs and consumed times. Specifically, given two partial routes p_1 and p_2 that terminate at the same node, p_1 is dominated by p_2 and therefore p_1 can be pruned, if: 1) p_1 is more costly than p_2; 2) p_1 takes longer than p_2. We also need a third criterion, however: 3) the set of nodes included in p_1 is a superset of the nodes included in p_2.

The third criteria is necessary to ensure equivalent remaining opportunities to capture the benefit of negative dual values. For example, consider partial paths p_1: $D-X-Y-Z$ and p_2: $D-W-Z$. We are tempted to prune p_1 if it is higher cost and consumes more time than path p_2. However, it may be the case that the optimal (i.e. most negative reduced cost) path is $D-X-Y-Z-W-D$. If we prune p_1, we will not encounter this path. Conversely, the path $D-W-Z-W-D$ (which should be both cheaper and less time consuming than $D-X-Y-Z-W-D$) is not actually a valid path because it repeats node W (and thus accumulates the dual value associated with the cover constraint for node W twice). Thus we can only prune p_1 if it is not only more costly and more time consuming than p_2 but also if it includes a superset of the nodes found in partial path p_2. This criteria greatly reduces the pruning opportunities, with significant negative impact on computational performance, as we demonstrate below.

To evaluate the ESPPRC approach, considering both run time and solution quality, we conducted two experiments, using the data sets described in Section 4.2.2 and detailed in Appendix Table C.1. First, we compared the optimal objective values of the LP relaxations of the arc-based model to the path-based model. Second, we compared the heuristic objective values found by (a) solving the LP relaxation of the path-based model to optimality and then finding the optimal integer solution with respect to the given columns in the restricted master problem versus (b) allowing the arc-based integer model to run for the same amount of time as the path-based model and taking the best integer feasible solution.

Consistently, we observed that the LP relaxation of the path-based model is substantially higher than that of the arc-based model, thus yielding a tighter lower bound. As seen in Columns ABM-LPR and ESPPRC-LPR of Table 4.3, the LP relaxation of the path-based model ranges from 18.33% to 32.75% higher than the arc-based model, with an average increase of 25.68%.

To compare the objective values, we solved the path-based LP relaxation to optimality, then found the best integer solution relative to those columns. We subsequently ran the arc-based integer model for the equivalent amount of time and took the lowest integer solution found during that time. In the best case, the path-based approach found a solution that was 69.37% lower than the ABM solution. In addition, in many cases, the arc-based approach found no integer-feasible solution (i.e. N/A), while the path-based approach always did. Finally, in all but three instances, the PBM solution was strictly better than the ABM solution, and all three of those instances were instances of the homogeneous version of the problem. Results for this experiment can be seen in Table 4.3, Columns ESPPRC-IP and ABM-IP.

Based on the network topology of Golden et al. [72], we conduct experiments on 4 different size networks, with 20, 25, 30, and 35 nodes respectively. For each network, we consider three different cases. The first case has only one vehicle type and Pareto dominance with respect to cost and time. The second case has multiple vehicle types but again has Pareto dominance. For the third case, in which we relax the Pareto dominance, we randomly generate ten instances.

Although the first two experiments show that the path-based approach yields a tighter lower bound and better integer solution in equivalent time, we suggest that the approach is still too slow for practical use. For example, although the path-based approach can be solved in roughly a minute for the heterogeneous non-Pareto instance with twenty nodes, the computational time grows exponentially for this approach as the number of nodes increases, taking about two hours to solve the instances with 35 nodes. As seen in Table 4.4, Columns RPBM-LPR & Pricing Problem and Branch & Bound for IP Heuristics, neither the LP relaxations of the restricted master nor the ultimate IP itself take long to solve; total run time depends primarily on the time required to solve the DPs at each iteration of the pricing problem. We therefore consider alternative approaches to solving the DP in the following sections.

#Nodes	#VehTypes	Pareto	ABM-LPR	ESPPRC-LPR	Δ LPR (%)	ESPPRC-IP	time(sec)	ABM-IP	time(sec)	Δ IP (%)
20	1 (6)	Yes	**438.072**	**548.778**	**25.27%**	**570.69**	**27**	**570.69**	**27**	**0.00%**
20	3 (2)	Yes	**383.637**	**488.224**	**27.26%**	**525.44**	**123**	**553.4**	**123**	**5.05%**
20	3 (2)	No	371.78	466.91	25.59%	489.82	49	526.94	66	7.04%
20	3 (2)	No	374.5	473.504	26.44%	514.54	68	535.86	66	3.98%
20	3 (2)	No	373.955	471.317	26.04%	501.17	92	582.21	66	13.92%
20	3 (2)	No	375.452	460.342	22.61%	493.58	102	529.69	66	6.82%
20	3 (2)	No	364.397	464.748	27.54%	494.77	67	522.39	66	5.29%
20	3 (2)	No	382.248	479.287	25.39%	506.86	35	517.52	66	2.06%
20	3 (2)	No	372.486	461.86	23.99%	505.58	47	513.26	66	1.50%
20	3 (2)	No	375.455	470.413	25.29%	501.74	28	508.85	66	1.40%
20	3 (2)	No	371.301	466.656	25.68%	500.96	110	546.96	66	8.41%
20	3 (2)	No	364.683	453.665	24.40%	483.85	57	529.42	66	8.61%
25	1 (9)	Yes	**559.204**	**661.708**	**18.33%**	**674.79**	**47**	**674.79**	**47**	**0.00%**
25	3 (3)	Yes	**489.413**	**606.384**	**23.90%**	**616.48**	**255**	**922.24**	**255**	**33.15%**
25	3 (3)	No	475.899	569.747	19.72%	598.79	160	764.29	233	21.65%
25	3 (3)	No	464.149	567.087	22.18%	572.85	154	631.79	233	9.33%
25	3 (3)	No	478.001	570.983	19.45%	595.24	192	683.24	233	12.88%
25	3 (3)	No	484.074	582.357	20.30%	594.7	281	929.81	233	36.04%
25	3 (3)	No	464.791	572.237	23.12%	577.96	245	829.94	233	30.36%
25	3 (3)	No	486.371	583.196	19.91%	595	305	1942.52	233	69.37%
25	3 (3)	No	475.794	574.853	20.82%	584.11	239	807.77	233	27.69%
25	3 (3)	No	481.448	584.172	21.34%	594.61	280	925.87	233	35.78%
25	3 (3)	No	463.837	566.54	22.14%	577.07	186	940	233	38.61%
25	3 (3)	No	483.451	594.391	22.95%	602.03	287	1381.03	233	56.41%
30	1 (12)	Yes	**556.388**	**690.789**	**24.16%**	**733.58**	**348**	**713.07**	**348**	**-2.88%**
30	3 (4)	Yes	**487.081**	**636.173**	**30.61%**	**650.24**	**1798**	**789.43**	**1798**	**17.63%**
30	3 (4)	No	480.186	605.767	26.15%	642.18	2814	742.4	2306	13.50%
30	3 (4)	No	478.29	607.222	26.96%	631.58	2422	653.76	2306	3.39%
30	3 (4)	No	491.584	620.204	26.16%	659	1679	743.2	2306	11.33%
30	3 (4)	No	464.098	599.535	29.18%	643.53	2372	835.23	2306	22.95%
30	3 (4)	No	467.434	603.99	29.21%	603.99	2895	697.17	2306	13.37%
30	3 (4)	No	468.403	611.449	30.54%	646.09	3016	841.7	2306	23.24%
30	3 (4)	No	472.99	604.492	27.80%	644.12	1464	709.8	2306	9.25%
30	3 (4)	No	478.938	614.081	28.22%	623.81	2821	N/A	2306	N/A
30	3 (4)	No	480.524	609.289	26.80%	624.22	1820	746.59	2306	16.39%
30	3 (4)	No	483.211	613.911	27.05%	645.02	1753	780.29	2306	17.34%
35	1 (12)	Yes	**640.683**	**801.635**	**25.12%**	**821.23**	**1708**	**821.23**	**1708**	**0.00%**
35	3 (4)	Yes	**560.893**	**744.589**	**32.75%**	**766.09**	**8003**	**N/A**	**8003**	**N/A**
35	3 (4)	No	572.354	724.176	26.53%	750.18	8497	852.84	8253	12.04%
35	3 (4)	No	556.932	716.294	28.61%	749.49	11436	802.4	8253	6.59%
35	3 (4)	No	533.847	692.536	29.73%	707.57	7945	N/A	8253	N/A
35	3 (4)	No	565.321	712.613	26.05%	735.07	9706	N/A	8253	N/A
35	3 (4)	No	532.77	701.591	31.69%	725.4	6040	N/A	8253	N/A
35	3 (4)	No	546.034	699.645	28.13%	728.95	9672	795.19	8253	8.33%
35	3 (4)	No	529.464	680.76	28.58%	706.56	2800	N/A	8253	N/A
35	3 (4)	No	551.074	709.405	28.73%	732.58	8407	N/A	8253	N/A
35	3 (4)	No	546.031	697.333	27.71%	717.77	10155	N/A	8253	N/A
35	3 (4)	No	554.51	704.135	26.98%	727.66	7866	N/A	8253	N/A

#VehTypes: number of vehicle types (number of vehicles per type in parenthesis),
ABM-LPR: Linear relaxation of ABM at the root node,
ESPPRC-LPR: Linear relaxation of ESPPRC at the root node,
ESPPRC-IP: heuristic integer solution at the root node,
ABM-IP: exact integer solution by branch & bound
$\Delta\mathrm{LPR}(\%) = [\mathrm{RSPPRC} - \mathrm{LPR}] - [\mathrm{ABM} - \mathrm{LPR}]/[\mathrm{ABM} - \mathrm{LPR}]$,
$\Delta\mathrm{IP}(\%) = ([\mathrm{ABM} - \mathrm{IP}] - [\mathrm{ESPPRC} - \mathrm{IP}])\,/[\mathrm{ABM} - \mathrm{LPR}]$

Table 4.3: Solution quality (ESPPRC)

Data Set			RPBM-LPR & Pricing Problem	Branch & Bound for IP	Total
#Nodes	#VehTypes	Pareto	time(sec)	time(sec)	time(sec)
20	1 (6)	Yes	**27**	**0**	**27**
20	3 (2)	Yes	**122**	**1**	**123**
20	3 (2)	No	49	0	49
20	3 (2)	No	67	1	68
20	3 (2)	No	92	0	92
20	3 (2)	No	101	1	102
20	3 (2)	No	67	0	67
20	3 (2)	No	35	0	35
20	3 (2)	No	46	1	47
20	3 (2)	No	28	0	28
20	3 (2)	No	109	1	110
20	3 (2)	No	56	1	57
25	1 (9)	Yes	**47**	**0**	**47**
25	3 (3)	Yes	**254**	**1**	**255**
25	3 (3)	No	158	2	160
25	3 (3)	No	152	2	154
25	3 (3)	No	189	3	192
25	3 (3)	No	280	1	281
25	3 (3)	No	244	1	245
25	3 (3)	No	304	1	305
25	3 (3)	No	237	2	239
25	3 (3)	No	278	2	280
25	3 (3)	No	184	2	186
25	3 (3)	No	283	4	287
30	1 (12)	Yes	**345**	**3**	**348**
30	3 (4)	Yes	**1786**	**12**	**1798**
30	3 (4)	No	2811	3	2814
30	3 (4)	No	2414	8	2422
30	3 (4)	No	1662	17	1679
30	3 (4)	No	2352	20	2372
30	3 (4)	No	2894	1	2895
30	3 (4)	No	3004	12	3016
30	3 (4)	No	1451	11	1464
30	3 (4)	No	2816	5	2821
30	3 (4)	No	1813	7	1820
30	3 (4)	No	1747	6	1753
35	1 (12)	Yes	**1663**	**44**	**1708**
35	3 (4)	Yes	**7859**	**144**	**8003**
35	3 (4)	No	8250	247	8497
35	3 (4)	No	11136	299	11436
35	3 (4)	No	7796	148	7945
35	3 (4)	No	9338	366	9706
35	3 (4)	No	5803	236	6040
35	3 (4)	No	9547	124	9672
35	3 (4)	No	2669	130	2800
35	3 (4)	No	8268	139	8407
35	3 (4)	No	9945	209	10155
35	3 (4)	No	7754	111	7866

Table 4.4: Computing time (ESPPRC)

4.5.2 Cycle Relaxation of ESPPRC (RSPPRC)

The DP is hard to solve when elementary paths are required because the dominance requirements significantly limit pruning opportunities. As observed by Desrochers et al. [127], removing the no-cycles restriction can greatly reduce run times. Furthermore, even if cycles are allowed, the equality restriction in the cover constraints will prevent such routes from being included in the optimal integer solution [134]. In fact, prior to the work of Feillet et al. [133], most existing research relaxed the no-cycle constraint in order to simplify the pricing problem. For example, Desrochers et al. [127] presented a pseudo-polynomial primal-dual labeling algorithm to effectively solve this problem. Recently, Baldacci et al. [79] proposed the use of *ng*-routes, which yields a compromise between elementary routes and non-elementary routes with cycles. Martinelli et al. [139] expanded further upon this idea by combining the decremental state-space relaxation [136] and completion bounds to accelerate the algorithm.

In this section, we consider alternative ways to solve the DP without requiring elementary paths, that is, by allowing cycles. We analyze the impact of this relaxation on both the time required to solve the pricing problem and also the strength of the LP relaxation. We refer to the variation of the pricing problem in which cycles are allowed as the cycle relaxation of ESPPRC (RSPPRC). In our approach to solving RSPPRC, we recognize that when cycles are allowed, it is no longer required that the set of nodes in partial route p_1 be a superset of the nodes in partial route p_2 in order for p_2 to dominate p_1 and thus for p_1 to be pruned. It is trivial to modify our original DP algorithm for solving ESPPRC accordingly, by simply eliminating the third pruning criterion.

Data Set			ESPPRC	RSPPRC	Δ / ESPPRC
#Nodes	#VehTypes	Pareto	time(sec)	time(sec)	%
20	1 (6)	Yes	**27**	**4**	**-85.19%**
20	3 (2)	Yes	**123**	**10**	**-91.87%**
20	3 (2)	No	49	7	-85.71%
20	3 (2)	No	68	7	-89.71%
20	3 (2)	No	92	8	-91.30%
20	3 (2)	No	102	8	-92.16%
20	3 (2)	No	67	9	-86.57%
20	3 (2)	No	35	6	-82.86%
20	3 (2)	No	47	8	-82.98%
20	3 (2)	No	28	7	-75.00%
20	3 (2)	No	110	7	-93.64%
20	3 (2)	No	57	8	-85.96%
25	1 (9)	Yes	**47**	**6**	**-87.23%**
25	3 (3)	Yes	**255**	**14**	**-94.51%**
25	3 (3)	No	160	10	-93.75%
25	3 (3)	No	154	13	-91.56%
25	3 (3)	No	192	9	-95.31%
25	3 (3)	No	281	12	-95.73%
25	3 (3)	No	245	13	-94.69%
25	3 (3)	No	305	12	-96.07%
25	3 (3)	No	239	13	-94.56%
25	3 (3)	No	280	12	-95.71%
25	3 (3)	No	186	13	-93.01%
25	3 (3)	No	287	13	-95.47%
30	1 (12)	Yes	**348**	**9**	**-97.41%**
30	3 (4)	Yes	**1798**	**23**	**-98.72%**
30	3 (4)	No	2814	24	-99.15%
30	3 (4)	No	2422	20	-99.17%
30	3 (4)	No	1679	21	-98.75%
30	3 (4)	No	2372	20	-99.16%
30	3 (4)	No	2895	25	-99.14%
30	3 (4)	No	3016	21	-99.30%
30	3 (4)	No	1464	21	-98.57%
30	3 (4)	No	2821	21	-99.26%
30	3 (4)	No	1820	20	-98.90%
30	3 (4)	No	1753	17	-99.03%
35	1 (12)	Yes	**1708**	**15**	**-99.12%**
35	3 (4)	Yes	**8003**	**48**	**-99.40%**
35	3 (4)	No	8497	40	-99.53%
35	3 (4)	No	11436	48	-99.58%
35	3 (4)	No	7945	37	-99.53%
35	3 (4)	No	9706	43	-99.56%
35	3 (4)	No	6040	48	-99.21%
35	3 (4)	No	9672	42	-99.57%
35	3 (4)	No	2800	45	-98.39%
35	3 (4)	No	8407	35	-99.58%
35	3 (4)	No	10155	46	-99.55%
35	3 (4)	No	7866	47	-99.40%

($\Delta = [\text{RSPPRC}] - [\text{ESPPRC}]$)

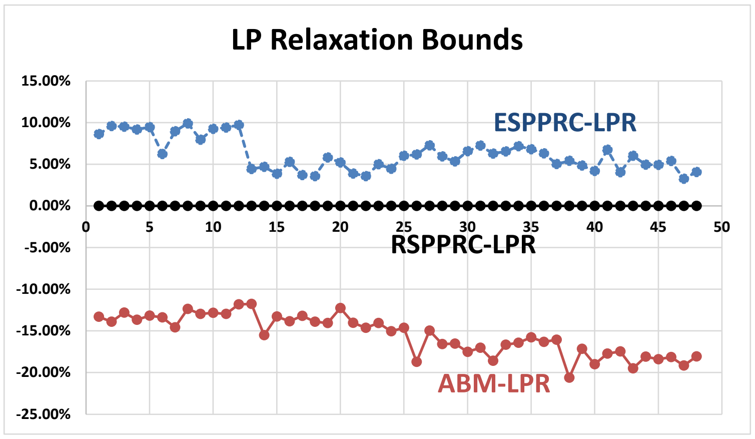
Table 4.5: Computing time (RSPPRC)

We conducted computational experiments on the same data as in 4.5.1 to evaluate run time, strength of the LP relaxation, and IP-heuristic solution quality. Figure 4.2 presents the resulting

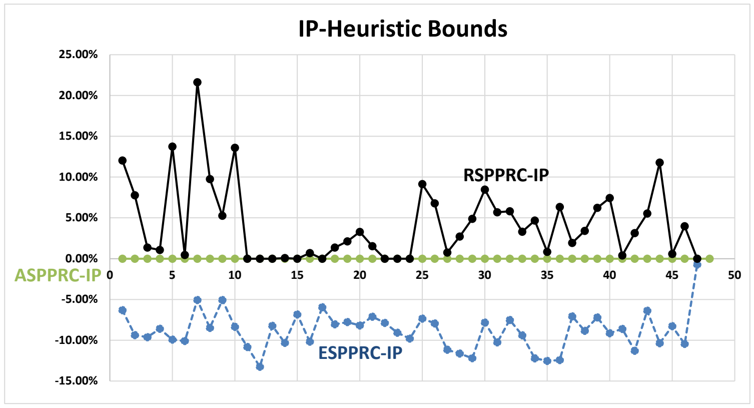
LP relaxation bounds and objective values for the IP-heuristic approaches to RSPPRC, ASPPRC (Augmented RSPPRC defined later) and ESPPRC. Note that the first column of Table 4.3, 4.4 and 4.5 provides a unique index for each instance and x axis in Figure 4.2 correspond to these indices. We observe that, by simply enabling a greater degree of pruning, we can solve RSPPRC much more quickly than ESPPRC, as demonstrated in Table 4.5. This table presents the total time, for both ESPPRC and RSPPRC, needed to solve the LP relaxation to optimality and then to solve "to optimality" the associated IP (note that we solve to optimality with respect to the available columns but we do not implement branch-and-price). As seen in Figure 4.2, however, the LP relaxation is significantly worse relative to when using ESPPRC (although still significantly better than the LP relaxation of the arc-based model, ABM-LPR). In turn, as seen in Figure 4.2, the quality of the heuristic IP solution (found when branching just on those columns found when solving for the LP relaxation) is worse than the ESPPRC approach as well.

To understand why the LP relaxation when cycles are allowed is worse than ESPPRC, consider the following example, as depicted in Figure 4.3. The true cost of path $D - A - B - C - D$ is 206 and this path covers nodes A,B, and C. When cycles are allowed, however, we can replace this variable with a variable corresponding to the path $D - A - B - C - A - B - C - D$, with value 1/2. Again, this covers nodes A,B, and C, but now with cost 106. The reduction is because we have reduced the travel distance from the depot to the cluster of nodes. Extending this idea, consider traversing the cycle $A - B - C - A$ m times with corresponding value $x = \frac{1}{m}$. Again, in the LP relaxation, this would satisfy coverage of A,B, and C, but with a cost of $\frac{6m}{m+1} + \frac{206}{m+1}$. As m approaches infinity, the cost approaches six, the cost of the cycle $A - B - C - A$. The only limiting factor here is the total time limit Q allowed for the path.

In addition, the IP heuristic objective is also worse because many of the columns identified when solving PBM-LPR using RSPPRC will not in fact be valid columns for the integer version of the problem because, as motivated in the previous example, many will contain cycles which are invalid under the equality constraints. Since we are not generating columns beyond the root node, this is particularly problematic.



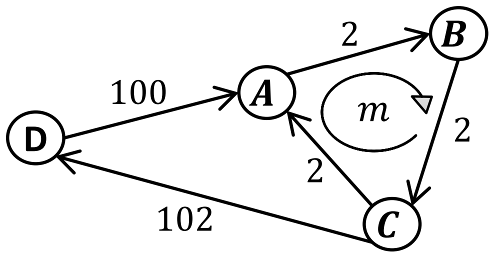
(Percentage difference of ESPPRC-LPR and ABM-LPR based on SPPRC-LPR)



(Percentage difference of RSPPRC-IP and ESPPRC-IP based on ASPPRC-IP)

Figure 4.2: Solution quality (RSPPRC)

Motivated by this, we augmented the RSPPRC approach to include the acyclic equivalent of each newly-generated column. That is, given a column where some node occurs more than once, we also create a new column in which we delete from the route all occurrences of that node except the first. We refer to this augmented RSPPRC as ASPPRC. Note that this does not impact the LP relaxation of RSPPRC (because these were not chosen to be included in the optimal LP solution), nor does it have any significant impact on the run time over solving RSPPRC. Therefore, ASPPRC is equivalent in run time to RSPPRC. For solution quality, as seen in Figure 4.2, ASPPRC is always better than RSPPRC (given that it has a superset of the columns of RSPPRC), however, it still has worse objective values than ESPPRC.



	Route Cost(c_r)	Value(x)	Total Cost
No cycle ($m = 0$)	$D - A - B - C - D$: 100+2+2+102 = 206	1	206
One cycle ($m = 1$)	$D - (A - B - C) - (A - B - C) - D$: 100+($\underline{2+2+2}$)+2+2+102 = 212	$\frac{1}{2}$	106
⋮	⋮	⋮	⋮
m cycle	$D - (A - B - C) \cdots (A - B - C) - D$: 100+m × ($\underline{2+2+2}$)+2+2+102 = 206+m × 6	$\frac{1}{m+1}$	$\frac{6m}{m+1} + \frac{206}{m+1}$

Figure 4.3: Example of weak lower bounds by the relaxation of no-cycle constraint

4.5.3 Weak Dominance Relaxation of ESPPRC (WESPPRC)

In the previous sections, we observed that allowing cycles made the DP approach much faster because we were not limited to pruning only when one path contained a superset of the nodes in another. On the other hand, allowing cycles greatly weakened the lower bound. Motivated by these facts, we propose a final alternative approach in which we do not allow cycles, but we eliminate the third pruning criterion. For example, if path p_1 is less costly and takes less time than path p_2, we allow path p_2 to be pruned, even if it does not contain a superset of the nodes in path p_1. We refer to this ESPPRC with weak dominance rule as the weak dominance relaxation of ESPPRC (WESPPRC). Note that this new approach no longer guarantees an optimal solution to the LP relaxation because we may prune a path that is in fact part of the optimal solution.

We observe, as seen in Table 4.6, that the result is a significant improvement in run time - in one instance, from more than three hours to roughly 34 seconds. We reiterate that the LP relaxation of WESPPRC is not guaranteed to be optimal (i.e. we may prune relevant columns). As seen in Table 4.6, however, the impact on the LP relaxation is very small, with a maximum increase of 0.94%. Finally, we note that the IP objective value of WESPPRC is sometimes worse and sometimes better than ESPPRC. However, we observe the difference in objective value to be within 5.24% in all cases, and often much less. Furthermore, it is as good or better than ASPPRC in all but one of the large problem instances, and faster than that approach in most cases.

Data Set			ESPPRC	WESPPRC	LPR objective ratio	IP objective ratio		Time ratio
#Nodes	#VehTypes	Pareto	time(sec)	time(sec)	WESPPRC-LPR / ESPPRC-LPR	WESPPRRC-IP / ESPPRC-IP	WESPPRRC-IP / ASPPRC-IP	WESPPRRC-IP / ASPPRC-IP
20	1 (6)	Yes	**27**	**3**	**100.20%**	**100.00%**	**93.69%**	**0.43**
20	3 (2)	Yes	**123**	**5**	**100.00%**	**100.83%**	**91.39%**	**0.45**
20	3 (2)	No	49	5	100.41%	100.29%	90.65%	0.71
20	3 (2)	No	68	5	100.94%	99.56%	91.00%	0.71
20	3 (2)	No	92	4	100.33%	101.74%	91.65%	0.50
20	3 (2)	No	102	6	100.04%	101.82%	91.55%	0.86
20	3 (2)	No	67	6	100.20%	101.45%	96.31%	0.67
20	3 (2)	No	35	5	100.44%	100.86%	92.31%	0.83
20	3 (2)	No	47	5	100.81%	98.29%	93.32%	0.63
20	3 (2)	No	28	4	100.03%	99.43%	91.12%	0.50
20	3 (2)	No	110	5	100.19%	101.68%	90.68%	0.63
20	3 (2)	No	57	4	100.50%	100.33%	87.05%	0.44
25	1 (9)	Yes	**47**	**4**	**100.28%**	**102.61%**	**94.16%**	**0.80**
25	3 (3)	Yes	**255**	**7**	**100.00%**	**101.11%**	**90.69%**	**0.47**
25	3 (3)	No	160	10	100.00%	104.31%	97.18%	1.00
25	3 (3)	No	154	6	100.22%	101.24%	90.94%	0.43
25	3 (3)	No	192	8	100.11%	100.94%	94.94%	0.80
25	3 (3)	No	281	7	100.04%	101.06%	92.93%	0.54
25	3 (3)	No	245	7	100.05%	101.12%	93.26%	0.54
25	3 (3)	No	305	7	100.40%	102.11%	93.76%	0.58
25	3 (3)	No	239	7	100.00%	100.71%	93.56%	0.58
25	3 (3)	No	280	8	100.42%	103.50%	95.36%	0.62
25	3 (3)	No	186	6	100.62%	100.33%	91.23%	0.43
25	3 (3)	No	287	7	100.20%	105.21%	94.90%	0.54
30	1 (12)	Yes	**348**	**6**	**100.16%**	**99.97%**	**92.64%**	**0.67**
30	3 (4)	Yes	**1798**	**12**	**100.11%**	**105.24%**	**96.90%**	**0.48**
30	3 (4)	No	2814	13	100.00%	101.07%	89.78%	0.57
30	3 (4)	No	2422	10	100.39%	103.21%	91.21%	0.48
30	3 (4)	No	1679	13	100.23%	98.19%	86.21%	0.59
30	3 (4)	No	2372	13	100.73%	98.66%	90.94%	0.59
30	3 (4)	No	2895	10	100.25%	100.25%	89.97%	0.40
30	3 (4)	No	3016	12	100.07%	101.35%	93.74%	0.57
30	3 (4)	No	1464	12	100.32%	97.73%	88.55%	0.55
30	3 (4)	No	2821	13	100.43%	105.24%	92.39%	0.59
30	3 (4)	No	1820	12	100.52%	102.46%	89.63%	0.60
30	3 (4)	No	1753	13	100.25%	103.64%	90.75%	0.76
35	1 (12)	Yes	**1708**	**11**	**100.02%**	**101.28%**	**94.12%**	**0.69**
35	3 (4)	Yes	**8003**	**25**	**100.06%**	**100.40%**	**91.53%**	**0.50**
35	3 (4)	No	8497	32	100.11%	99.64%	92.46%	0.76
35	3 (4)	No	11436	34	100.16%	103.93%	94.44%	0.69
35	3 (4)	No	7945	25	100.28%	100.49%	91.85%	0.64
35	3 (4)	No	9706	43	100.05%	100.01%	88.74%	1.00
35	3 (4)	No	6040	28	100.07%	98.02%	91.78%	0.56
35	3 (4)	No	9672	26	100.36%	97.84%	87.72%	0.60
35	3 (4)	No	2800	21	100.83%	100.66%	92.34%	0.45
35	3 (4)	No	8407	30	100.47%	99.67%	89.25%	0.83
35	3 (4)	No	10155	30	100.07%	101.27%	100.54%	0.63
35	3 (4)	No	7866	28	100.16%	100.00%	90.18%	0.57
Max					**100.94%**	**105.24%**	**100.54%**	**1.00**
Min					**100.00%**	**97.73%**	**86.21%**	**0.40**
Average					**100.26%**	**101.06%**	**92.11%**	**0.61**

Table 4.6: Computing time & solution quality (WESPPRC)

Finally, we present the optimality gap for WESPPRC in Table 4.7. For future research, we could implement and evaluate the total time and solution quality of using branch-and-price with WESPPRC embedded in each node of the branch-and-bound tree.

#Nodes	#VehTypes	Pareto	WESPPRC-LPR	WESPPRC-IP	Gap (%)
20	1 (6)	Yes	549.897	570.69	3.78%
20	3 (2)	Yes	488.245	529.82	8.52%
20	3 (2)	No	468.814	491.26	4.79%
20	3 (2)	No	477.948	512.26	7.18%
20	3 (2)	No	472.887	509.91	7.83%
20	3 (2)	No	460.533	502.58	9.13%
20	3 (2)	No	465.672	501.92	7.78%
20	3 (2)	No	481.374	511.21	6.20%
20	3 (2)	No	465.602	496.94	6.73%
20	3 (2)	No	470.577	498.89	6.02%
20	3 (2)	No	467.541	509.39	8.95%
20	3 (2)	No	455.922	485.47	6.48%
25	1 (9)	Yes	663.535	692.39	4.35%
25	3 (3)	Yes	606.392	623.35	2.80%
25	3 (3)	No	569.747	624.6	9.63%
25	3 (3)	No	568.332	579.95	2.04%
25	3 (3)	No	571.638	600.84	5.11%
25	3 (3)	No	582.572	600.98	3.16%
25	3 (3)	No	572.519	584.41	2.08%
25	3 (3)	No	585.549	607.58	3.76%
25	3 (3)	No	574.853	588.25	2.33%
25	3 (3)	No	586.631	615.44	4.91%
25	3 (3)	No	570.038	578.98	1.57%
25	3 (3)	No	595.57	633.38	6.35%
30	1 (12)	Yes	691.928	733.35	5.99%
30	3 (4)	Yes	636.857	684.33	7.45%
30	3 (4)	No	605.792	649.03	7.14%
30	3 (4)	No	609.594	651.88	6.94%
30	3 (4)	No	621.651	647.04	4.08%
30	3 (4)	No	603.928	634.9	5.13%
30	3 (4)	No	605.48	605.48	0.00%
30	3 (4)	No	611.847	654.79	7.02%
30	3 (4)	No	606.452	629.5	3.80%
30	3 (4)	No	616.733	656.48	6.44%
30	3 (4)	No	612.475	639.59	4.43%
30	3 (4)	No	615.416	668.47	8.62%
35	1 (12)	Yes	801.829	831.78	3.74%
35	3 (4)	Yes	745.024	769.16	3.24%
35	3 (4)	No	724.998	747.45	3.10%
35	3 (4)	No	717.455	778.93	8.57%
35	3 (4)	No	694.448	711.04	2.39%
35	3 (4)	No	712.935	735.15	3.12%
35	3 (4)	No	702.082	711.03	1.27%
35	3 (4)	No	702.151	713.22	1.58%
35	3 (4)	No	686.392	711.25	3.62%
35	3 (4)	No	712.741	730.13	2.44%
35	3 (4)	No	697.844	726.9	4.16%
35	3 (4)	No	705.27	727.66	3.17%

Table 4.7: Optimality gap (WESPPRC)

4.6 Conclusions

In this paper, we present a variation of VRP, TCHVRP, in which arc times and costs vary by vehicle type, and in particular, Pareto dominance is not assumed. We demonstrate the challenges of solving this problem with an arc-based model and consider a path-based model as a viable alternative. In particular, we focus on dynamic programming-based approaches to solving the pricing problem when solving the path-based formulation via column generation. We show that allowing cycles within the routes makes DP approaches to generating routes much faster, but at the expense of a weaker LP relaxation and poorer integer solutions when routes are generated only at the root node. We also show that prohibiting cycles, with fully-defined pruning, is prohibitive slow. As an alternative, we propose a heuristic where cycles are not allowed, but pruning is expanded to allow those cases where set of the nodes covered by one partial route is not necessarily a superset of the nodes covered by another. Although we can no longer guarantee the optimality of the LP relaxation, we show that this markedly improves solution time over the pure case and markedly improves solution quality over the case in which cycles are allowed.

CHAPTER 5

Conclusions / Future Research

5.1 Conclusions / Future Research

In this dissertation, we study the role of dominance in the multi-criteria optimization problem and the vehicle routing problem. Chapter 2 introduces an iterative and interactive IP-based approach to clarify ill-defined or multiple objectives in healthcare provider scheduling problem. In Chapter 3, we study the concept of dominance and develop an exact algorithm to find all points on the Pareto frontier. Chapter 4 focuses on a column generation approach to solve the vehicle routing problem. We propose dynamic programming-based approaches using dominance and evaluate several relaxation methods to address the pricing problem.

In Chapter 3, the contributions of our research are in developing, implementing, and evaluating an IP-based approach for generating the exhaustive set of Pareto-dominant solutions to a multi-criteria residency scheduling problem. We discuss both the tractability and the practicality of our approach as applied to the University of Michigan Pediatric Emergency Department. In the future, we plan to extend this research in the following ways:

- Apply the proposed method to other multi-objective problems with discrete metric values.
- Accelerate the run time of individual IPs when it takes too long time to find a Pareto point. We have shown that the use of warm starts is helpful to reduce the run time of the original IPs, and thus the overall run time. More research on the impact of various warm start strategies

on running time or a way to use the similarity of IPs in the lexicographic optimization or a perturbation of upper bounds to avoid extremely slow IP instances is a logical next step.

- Finally, we seek to elicit hidden information about the Chiefs' preferences through an interactive precedure in which they evaluate the Pareto schedules and provide feedback toward their true preference. We could proactively use the information in the future decision process in order to forecast the subset area where the preferred schedule exists instead of exploring whole solution space.

In Chapter 4, we introduce the dominance rule and dynamic programming-based approaches to solving the ESPPRC which is the pricing problem when solving the TCHVRP via column generation. There are more topics to investigate this network-based column generation approach and its applications further. Several directions for future research are described as below:

- We have observed that the superset condition in the dominance rule reduces the ability to prune non-optimal states in a dynamic programming, and the relaxation of the superset condition improves solution time significantly without losing huge optimality gap. In future research, instead of relaxing the superset condition strictly, we could develop intelligent heuristics for identifying non-optimal states by considering cost, time, and superset information comprehensively.

- We could parallelize the dynamic programming approach to solve the pricing problem. Since the shortest path problem for different types of vehicles and the state for each subpath in dynamic programming could be solved independently, the parallel computing power could accelerate the processing time.

- We could integrate the dynamic programming-based approach with an exact branch-and-price algorithm. We could use our method that yields tighter bounds than the case in which cycles are allowed to solve the linear relaxation of the set partitioning problem in each node of the branch-and-bound procedure.

- Recent healthcare provider scheduling studies suggest a column generation based decomposition algorithm to solving these more complex healthcare personnel scheduling problems [6]. We could apply this approach to the challenging healthcare personnel scheduling such as the long-term or multi-year scheduling problems [140], scheduling problems integrating different provider schedules [141, 142], and flexible shift scheduling problems that allow flexible start times and variable shift lengths [4].

APPENDIX A

Appendix for Chapter 2

A.1 Metric Formulation

Notation

G	set of resident *groups* where $g \subset G$ is the set of residents who have the same number of working days, $g = \{r \in R : \lvert W_r \rvert \text{ is the same}\} \subset R$
I	set of intern-undesirable shifts, $I \in S$; In our data instances, shift 1 and 7 are in I
P	set of shifts that are defined as the post-continuity clinic shifts, $P \subset S$; In our data instances shifts 6 and 7 (i.e., 8 PM – 5 AM and 11 PM – 8 AM) are in P
E	set of day-shift pairs that are defined as the optional shifts around the end of the planning horizon, $E \subset D \times S$
U	set of bad (undesirable) sleep patterns where $u \in U$ is a combination of shift offsets on multiple days. Note that $\mid u \mid$ represents the number of shift offsets in u
$U_{(d,u)}$	set of date-shift pairs associated with bad sleep pattern u on day d, $U_{(d,u)} \subset D \times S$ $\forall (d,u) \in D \times U$

Input parameters

$\underline{S}_g, \overline{S}_g$	lower and upper bounds on the number of total shifts for a resident in group g, $\forall g \in G$
$\underline{N}_g$, $\overline{N}_g$	lower and upper bounds on the number of night shifts for a resident in group g, $\forall g \in G$

$\underline{U}_g$, $\overline{U}_g$	lower and upper bounds on the number of bad sleep patterns for a resident in a group $g, \forall g \in G$
$\underline{P}_g$, $\overline{P}_g$	lower and upper bounds on the number of post-continuity clinic shifts for a resident in a group $g, \forall g \in G$
$\underline{I}, \overline{I}$	lower and upper bounds on the number of intern residents assigned to intern-prohibited shifts in the planning horizon
$\underline{E}, \overline{E}$	lower and upper bounds on the number of covered optional shifts in the planning horizon
$\underline{F}, \overline{F}$	lower and upper bounds on the number of uncovered flex shifts in the planning horizon

Metric variables

y_{rdu}	binary variable, equals 1 if resident r is assigned to bad sleep pattern u on day d; otherwise 0 $\forall r \in R, \forall d \in D, \forall u \in U$
z_{rd}	binary variable, equals 1 if resident $r \in R$ is assigned to work a post-continuity clinic shift on day $d \in C_r$; otherwise 0 $\forall r \in R, \forall d \in D$
s_r	number of total shifts for resident r, $s_r = \sum_{d \in D} \sum_{s \in S} x_{rds}$ for $\forall r \in R$
n_r	number of night shifts for resident r, $n_r = \sum_{d \in D} \sum_{s \in N} x_{rds}$ for $\forall r \in R$
u_r	number of bad sleep patterns for resident r, $u_r = \sum_{d \in D} \sum_{u \in U} y_{rdu}$ for $\forall r \in R$
p_r	number of post-continuity clinic shifts for resident r, $p_r = \sum_{d \in C_r} \sum_{s \in P} x_{rds}$ for $\forall r \in R$
i	total number of intern residents assigned to intern-undersirable shifts in the planning horizon, $i = \sum_{r \in \{r: l_r = \text{interns}\}} \sum_{d \in D} \sum_{s \in I} x_{rds}$
e	total number of covered optional shifts in the planning horizon, $e = \sum_{r \in R} \sum_{(d,s) \in E} x_{rds}$
f	total number of uncovered flex shifts in the planning horizon, $f = \mid D \mid - \sum_{r \in R} \sum_{d \in D} \sum_{s \in F} x_{rds}$

Metric constraints

$$y_{rds} \leq x_{rij} \qquad \forall r \in R, \forall d \in D, \forall u \in U, \forall (i,j) \in U_{(d,u)} \tag{A.1}$$

$$y_{rds} + \mid u \mid \geq \sum_{(d,s) \in U_{(d,u)}} x_{rds} + 1 \qquad \forall r \in R, \forall d \in D, \forall u \in U \tag{A.2}$$

$$z_{rd} \geq x_{rds} \qquad \forall r \in R, \forall d \in C_r, \forall s \in P \tag{A.3}$$

$$z_{rd} \leq \sum_{s \in P} x_{rds} \qquad \forall r \in R, \forall d \in C_r \tag{A.4}$$

$$s_r = \sum_{d \in D} \sum_{s \in S} x_{rds} \qquad \forall r \in R \tag{A.5}$$

$$n_r = \sum_{d \in D} \sum_{s \in N} x_{rds} \qquad \forall r \in R \tag{A.6}$$

$$u_r = \sum_{d \in D} \sum_{u \in U} y_{rdu} \qquad \forall r \in R \tag{A.7}$$

$$p_r = \sum_{d \in C_r} \sum_{s \in P} x_{rds} \qquad \forall r \in R \tag{A.8}$$

$$i = \sum_{r \in R} \sum_{d \in D} \sum_{s \in I} x_{rds} \tag{A.9}$$

$$e = \sum_{r \in R} \sum_{(d,s) \in OPT} x_{rds} \tag{A.10}$$

$$f = \mid D \mid - \sum_{r \in R} \sum_{d \in D} \sum_{s \in F} x_{rds} \tag{A.11}$$

$$\underline{S}_g \leq s_r \leq \overline{S}_g \qquad \forall g \in G, \forall r \in g \tag{A.12}$$

$$\underline{N}_g \leq n_r \leq \overline{N}_g \qquad \forall g \in G, \forall r \in g \tag{A.13}$$

$$\underline{U}_g \leq u_r \leq \overline{U}_g \qquad \forall g \in G, \forall r \in g \tag{A.14}$$

$$\underline{P}_g \leq p_r \leq \overline{P}_g \qquad \forall g \in G, \forall r \in g \tag{A.15}$$

$$\underline{I} \leq i \leq \overline{I} \tag{A.16}$$

$$\underline{E} \leq e \leq \overline{E} \tag{A.17}$$

$$\underline{F} \leq f \leq \overline{F} \tag{A.18}$$

$$y_{rdu} \in \{0,1\} \qquad \forall r \in R, \forall d \in D, \forall u \in U \tag{A.19}$$

$$z_{rd} \in \{0,1\} \qquad \forall r \in R, \forall d \in C_r \tag{A.20}$$

Constraints (A.1) and (A.2) link decision variables x_{rds} and auxiliary variables y_{rds} to count the number of bad sleep patterns. Constraints (A.3) and (A.4) link decision variables x_{rds} and auxiliary variables z_{rd} to count the number of post-continuity clinic shifts.

Metric variables (A.5) measure the total number of shifts assigned to resident r, metric variables (A.6) measure the the total number of night shifts assigned to resident r, metric variables (A.7) measure the total number of BSPs assigned to resident r, metric variables (A.8) measure the total number of PCCs assigned to resident r, metric variables (A.9) measure the total number of intern residents assigned to intern-prohibited shifts in the planning horizon, metric variables (A.10) measure the number of covered optional shifts in the planning horizon, metric variables (A.11) measure the total number of uncovered flex shifts (UFSs) in the planning horizon.

Metric constraints (A.12) and (A.13) control the upper and lower bounds on the number of shifts and night shifts that is assigned to each resident r in a group $g \in G$, metric constraints (A.14) control the number of BSPs that are assigned to each resident r in a group $g \in G$, metric constraints (A.15) control that the number of PCCs that are assigned to each resident r in a group $g \in G$, metric constraints (A.16) control the total number of interns assigned to intern residents assigned to intern-undersirable shifts in the planning horizon, metric constraints (A.17) control the total number of covered optional shifts in the planning horizon, metric constraint (A.18) control the total number of UFSs in the planning horizon.

APPENDIX B

Appendix for Chapter 3

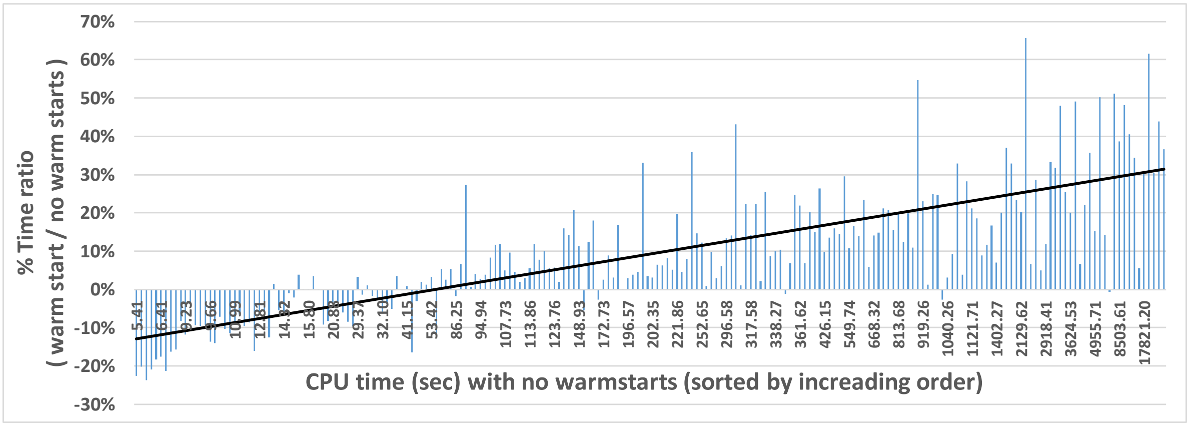
B.1 Computational Results

2012-2013								2013-2014								2014-2015							
Index	Month	Levels	Infeasible	Time(h:m:s)	#Pareto Points	#Repeats	Timeout	Index	Month	Levels	Infeasible	Time(h:m:s)	#Pareto Points	#Repeats	Timeout	Index	Month	Levels	Infeasible	Time(h:m:s)	#Pareto Points	#Repeats	Timeout
1	July	S1V1	No	0:00:39	3	0	No	82	July	S1V1	No	0:00:59	5	2	No	163	July	S1V1	No	0:02:35	9	0	No
2	July	S1V2	No	0:00:37	3	0	No	83	July	S1V2	No	0:02:17	10	2	No	164	July	S1V2	No	0:02:52	9	0	No
3	July	S1V3	No	0:48:44	9	2	Yes	84	July	S1V3	No	0:04:04	15	3	No	165	July	S1V3	No	4:55:12	82	0	No
4	July	S2V1	No	0:00:32	3	0	No	85	July	S2V1	No	0:00:52	5	2	No	166	July	S2V1	No	0:01:42	8	1	No
5	July	S2V2	No	0:00:30	3	0	No	86	July	S2V2	No	0:01:48	10	2	No	167	July	S2V2	No	0:01:49	8	1	No
6	July	S2V3	No	0:54:26	38	1	No	87	July	S2V3	No	0:02:26	14	0	No	168	July	S2V3	No	3:27:14	215	5	No
7	July	S3V1	No	0:00:30	3	0	No	88	July	S3V1	No	0:00:43	5	2	No	169	July	S3V1	No	0:01:29	8	1	No
8	July	S3V2	No	0:00:28	3	0	No	89	July	S3V2	No	0:01:30	10	2	No	170	July	S3V2	No	0:01:38	8	1	No
9	July	S3V3	No	0:28:49	38	1	No	90	July	S3V3	No	0:02:04	14	0	No	171	July	S3V3	No	2:03:05	212	4	No
10	August	S1V1	No	0:12:25	48	5	No	91	August	S1V1	No	0:00:15	1	0	No	172	August	S1V1	No	0:06:16	24	0	No
11	August	S1V2	No	0:13:17	48	5	No	92	August	S1V2	No	0:00:15	1	0	No	173	August	S1V2	No	0:07:24	24	0	No
12	August	S1V3	No	1:35:30	204	15	No	93	August	S1V3	No	0:00:41	3	0	No	174	August	S1V3	No	0:41:38	80	0	No
13	August	S2V1	No	0:10:46	48	5	No	94	August	S2V1	No	0:00:12	1	0	No	175	August	S2V1	No	0:05:55	24	0	No
14	August	S2V2	No	0:10:50	48	5	No	95	August	S2V2	No	0:00:13	1	0	No	176	August	S2V2	No	0:05:19	24	0	No
15	August	S2V3	No	0:58:46	203	13	No	96	August	S2V3	No	0:00:34	3	0	No	177	August	S2V3	No	0:18:29	60	6	No
16	August	S3V1	No	0:09:34	48	5	No	97	August	S3V1	No	0:00:11	1	0	No	178	August	S3V1	No	0:04:26	24	0	No
17	August	S3V2	No	0:09:34	48	5	No	98	August	S3V2	No	0:00:11	1	0	No	179	August	S3V2	No	0:05:04	24	0	No
18	August	S3V3	No	0:48:15	203	13	No	99	August	S3V3	No	0:00:27	3	0	No	180	August	S3V3	No	0:14:45	60	6	No
19	September	S1V1	No	0:01:27	6	0	No	100	September	S1V1	No	0:09:22	50	6	No	181	September	S1V1	Yes	0:00:01	0	0	No
20	September	S1V2	No	0:01:32	6	0	No	101	September	S1V2	No	0:04:32	24	3	No	182	September	S1V2	Yes	0:00:01	0	0	No
21	September	S1V3	No	0:02:58	11	0	No	102	September	S1V3	No	1:16:54	96	7	No	183	September	S1V3	Yes	0:00:01	0	0	No
22	September	S2V1	No	0:03:22	19	0	No	103	September	S2V1	No	0:05:34	37	4	No	184	September	S2V1	No	0:00:15	2	0	No
23	September	S2V2	No	0:03:29	19	0	No	104	September	S2V2	No	0:01:44	12	1	No	185	September	S2V2	No	0:00:16	2	0	No
24	September	S2V3	No	0:08:06	42	7	No	105	September	S2V3	No	0:33:59	143	5	No	186	September	S2V3	No	0:04:27	18	16	No
25	September	S3V1	No	0:03:09	19	0	No	106	September	S3V1	No	0:05:00	37	4	No	187	September	S3V1	No	0:00:07	1	0	No
26	September	S3V2	No	0:03:09	19	0	No	107	September	S3V2	No	0:01:28	12	1	No	188	September	S3V2	No	0:00:07	1	0	No
27	September	S3V3	No	0:06:25	36	1	No	108	September	S3V3	No	0:28:18	143	2	No	189	September	S3V3	No	0:01:21	10	0	No
28	October	S1V1	No	0:00:14	1	0	No	109	October	S1V1	No	0:08:56	14	0	No	190	October	S1V1	Yes	0:00:01	0	0	No
29	October	S1V2	No	0:00:15	1	0	No	110	October	S1V2	No	0:17:41	19	0	No	191	October	S1V2	Yes	0:00:01	0	0	No
30	October	S1V3	No	0:06:08	22	0	No	111	October	S1V3	No	0:31:30	37	0	No	192	October	S1V3	Yes	0:00:01	0	0	No
31	October	S2V1	No	0:00:11	1	0	No	112	October	S2V1	No	0:05:30	13	0	No	193	October	S2V1	No	0:01:02	5	3	No
32	October	S2V2	No	0:00:12	1	0	No	113	October	S2V2	No	0:10:19	18	0	No	194	October	S2V2	No	0:01:46	10	0	No
33	October	S2V3	No	0:02:12	10	0	No	114	October	S2V3	No	0:16:33	36	0	No	195	October	S2V3	No	0:12:17	54	6	No
34	October	S3V1	No	0:00:08	1	0	No	115	October	S3V1	No	0:05:08	13	0	No	196	October	S3V1	No	0:00:49	5	3	No
35	October	S3V2	No	0:00:08	1	0	No	116	October	S3V2	No	0:08:10	18	0	No	197	October	S3V2	No	0:01:32	10	0	No
36	October	S3V3	No	0:01:57	10	0	No	117	October	S3V3	No	0:16:17	36	0	No	198	October	S3V3	No	0:11:11	55	6	No
37	November	S1V1	No	0:00:16	1	0	No	118	November	S1V1	No	0:00:22	2	0	No	199	November	S1V1	No	0:05:11	22	0	No
38	November	S1V2	No	0:00:16	1	0	No	119	November	S1V2	No	0:00:22	2	0	No	200	November	S1V2	No	0:17:02	35	0	No
39	November	S1V3	No	0:01:07	5	0	No	120	November	S1V3	No	0:00:32	2	0	No	201	November	S1V3	No	0:12:31	31	0	No
40	November	S2V1	No	0:00:12	1	0	No	121	November	S2V1	No	0:00:10	1	0	No	202	November	S2V1	No	0:04:17	22	0	No
41	November	S2V2	No	0:00:11	1	0	No	122	November	S2V2	No	0:00:10	1	0	No	203	November	S2V2	No	0:17:33	39	0	No
42	November	S2V3	No	0:01:25	9	0	No	123	November	S2V3	No	0:00:22	2	0	No	204	November	S2V3	No	0:12:06	43	0	No
43	November	S3V1	No	0:00:10	1	0	No	124	November	S3V1	No	0:00:10	1	0	No	205	November	S3V1	No	0:03:31	22	0	No
44	November	S3V2	No	0:00:10	1	0	No	125	November	S3V2	No	0:00:10	1	0	No	206	November	S3V2	No	0:07:39	39	0	No
45	November	S3V3	No	0:01:15	9	0	No	126	November	S3V3	No	0:00:21	2	0	No	207	November	S3V3	No	0:11:47	52	0	No
46	March	S1V1	No	0:04:07	16	0	No	127	March	S1V1	No	0:12:11	29	0	No	208	March	S1V1	Yes	0:00:01	0	0	No
47	March	S1V2	No	0:04:18	16	0	No	128	March	S1V2	No	0:12:34	29	0	No	209	March	S1V2	Yes	0:00:01	0	0	No
48	March	S1V3	No	0:04:05	16	0	No	129	March	S1V3	No	1:26:56	142	35	No	210	March	S1V3	Yes	0:00:01	0	0	No
49	March	S2V1	No	0:03:21	16	0	No	130	March	S2V1	No	0:03:11	19	0	No	211	March	S2V1	No	0:04:30	14	0	No
50	March	S2V2	No	0:03:37	16	0	No	131	March	S2V2	No	0:03:12	19	0	No	212	March	S2V2	No	0:05:13	14	0	No
51	March	S2V3	No	0:03:35	16	0	No	132	March	S2V3	No	0:52:23	134	37	No	213	March	S2V3	No	7:19:33	387	10	No
52	March	S3V1	No	0:03:06	16	0	No	133	March	S3V1	No	0:02:55	19	0	No	214	March	S3V1	No	0:05:04	19	0	No
53	March	S3V2	No	0:03:16	16	0	No	134	March	S3V2	No	0:02:54	19	0	No	215	March	S3V2	No	0:06:16	19	0	No
54	March	S3V3	No	0:03:15	16	0	No	135	March	S3V3	No	0:41:58	129	14	No	216	March	S3V3	No	3:51:11	377	49	No
55	April	S1V1	No	2:18:13	70	0	No	136	April	S1V1	No	0:02:47	8	0	No	217	April	S1V1	No	0:34:33	61	6	No
56	April	S1V2	No	0:41:00	24	34	No	137	April	S1V2	No	0:02:40	8	0	No	218	April	S1V2	No	0:37:22	61	6	No
57	April	S1V3	No	0:42:54	48	30	No	138	April	S1V3	No	0:05:48	11	0	No	219	April	S1V3	No	1:07:12	87	13	No
58	April	S2V1	No	1:01:06	70	0	No	139	April	S2V1	No	0:01:57	8	0	No	220	April	S2V1	No	0:19:22	75	5	No
59	April	S2V2	No	0:16:47	24	34	No	140	April	S2V2	No	0:02:03	8	0	No	221	April	S2V2	No	0:19:44	75	5	No
60	April	S2V3	No	0:24:16	48	30	No	141	April	S2V3	No	0:04:22	13	0	No	222	April	S2V3	No	1:39:52	212	51	No
61	April	S3V1	No	0:45:36	70	0	No	142	April	S3V1	No	0:01:48	8	0	No	223	April	S3V1	No	0:18:55	75	5	No
62	April	S3V2	No	0:15:49	24	34	No	143	April	S3V2	No	0:01:48	8	0	No	224	April	S3V2	No	0:18:37	75	5	No
63	April	S3V3	No	0:21:43	48	30	No	144	April	S3V3	No	0:03:42	13	0	No	225	April	S3V3	No	1:09:57	204	28	No
64	May	S1V1	No	0:00:42	4	0	No	145	May	S1V1	No	3:42:37	77	8	Yes	226	May	S1V1	Yes	0:00:01	0	0	No
65	May	S1V2	No	0:00:42	4	0	No	146	May	S1V2	No	0:44:09	4	1	Yes	227	May	S1V2	Yes	0:00:01	0	0	No
66	May	S1V3	No	0:04:02	18	0	No	147	May	S1V3	No	0:31:39	2	1	Yes	228	May	S1V3	Yes	0:00:01	0	0	No
67	May	S2V1	No	0:00:34	4	0	No	148	May	S2V1	No	9:43:19	209	10	No	229	May	S2V1	No	0:00:15	1	0	No
68	May	S2V2	No	0:00:36	4	0	No	149	May	S2V2	No	2:53:11	37	9	Yes	230	May	S2V2	No	0:00:16	1	0	No
69	May	S2V3	No	0:03:16	18	0	No	150	May	S2V3	No	8:17:11	86	31	Yes	231	May	S2V3	No	0:00:17	1	0	No
70	May	S3V1	No	0:00:31	4	0	No	151	May	S3V1	No	6:59:47	209	10	No	232	May	S3V1	No	0:00:14	1	0	No
71	May	S3V2	No	0:00:30	4	0	No	152	May	S3V2	No	3:26:40	39	8	Yes	233	May	S3V2	No	0:00:14	1	0	No
72	May	S3V3	No	0:02:48	18	0	No	153	May	S3V3	No	1:31:16	37	19	Yes	234	May	S3V3	No	0:00:15	1	0	No
73	June	S1V1	Yes	0:00:01	0	0	No	154	June	S1V1	No	1:45:17	16	1	No	235	June	S1V1	No	0:01:57	10	2	No
74	June	S1V2	Yes	0:00:01	0	0	No	155	June	S1V2	No	0:41:22	1	1	Yes	236	June	S1V2	No	0:02:13	10	2	No
75	June	S1V3	Yes	0:00:01	0	0	No	156	June	S1V3	No	0:36:17	1	1	Yes	237	June	S1V3	No	0:06:39	21	4	No
76	June	S2V1	No	0:00:10	1	0	No	157	June	S2V1	No	2:34:25	49	2	Yes	238	June	S2V1	No	0:01:48	11	2	No
77	June	S2V2	No	0:00:10	1	0	No	158	June	S2V2	No	3:49:56	64	29	Yes	239	June	S2V2	No	0:01:42	11	2	No
78	June	S2V3	No	0:00:11	1	0	No	159	June	S2V3	No	24:04:21	306	50	Yes	240	June	S2V3	No	0:05:12	24	4	No
79	June	S3V1	No	0:00:07	1	0	No	160	June	S3V1	No	3:34:59	106	5	No	241	June	S3V1	No	0:01:32	11	2	No
80	June	S3V2	No	0:00:07	1	0	No	161	June	S3V2	No	2:52:43	75	19	Yes	242	June	S3V2	No	0:01:33	11	2	No
81	June	S3V3	No	0:00:07	1	0	No	162	June	S3V3	No	9:48:23	202	68	Yes	243	June	S3V3	No	0:04:43	24	4	No

Table B.1: Computational results (warm starts version)

					No (No warm start)		M (Mutiple warm starts)	
Index	Year	Month	Levels	Infeasible	Timeout	Max IP Time(sec)	Timeout	Max IP Time(sec)
3	2012-2013	July	S1V3	No	Yes	1800.535	Yes	1800.483
6	2012-2013	July	S2V3	No	Yes	1800.351	No	768.61
145	2013-2014	May	S1V1	No	Yes	1800.31	Yes	1800.445
146	2013-2014	May	S1V2	No	Yes	1800.39	Yes	1800.914
147	2013-2014	May	S1V3	No	Yes	1800.31	Yes	1801.002
149	2013-2014	May	S2V2	No	Yes	1800.60	Yes	1800.437
150	2013-2014	May	S2V3	No	Yes	1800.49	Yes	1800.728
152	2013-2014	May	S3V2	No	Yes	1800.93	Yes	1800.426
153	2013-2014	May	S3V3	No	Yes	1800.81	Yes	1801.478
154	2013-2014	June	S1V1	No	Yes	1800.91	No	1010.45
155	2013-2014	June	S1V2	No	Yes	1801.32	Yes	1801.707
156	2013-2014	June	S1V3	No	Yes	1800.31	Yes	1800.926
157	2013-2014	June	S2V1	No	No	925.37	Yes	1800.406
158	2013-2014	June	S2V2	No	Yes	1800.50	Yes	1800.887
159	2013-2014	June	S2V3	No	Yes	1800.31	Yes	1801.078
161	2013-2014	June	S3V2	No	Yes	1800.60	Yes	1801.203
162	2013-2014	June	S3V3	No	Yes	1801.28	Yes	1800.885
165	2014-2015	July	S1V3	No	Yes	1800.306	No	756.28

Table B.2: Timeout instances by either no warm start or warm starts



Total 210 comparable problem instances are sorted by no warm start CPU time where the comparable problem instance means a problem instance that is solved by both no warm start and warm starts. A black line represents linear regression to show the fact that warm starts strategy tends to improve a lot with a significant difference as running time takes a long.

Figure B.1: CPU time ratio of warm starts to no warm start for comparable problem instances

APPENDIX C

Appendix for Chapter 4

C.1 Problem Data

We generate TCHVRP instances based on the data sets of Golden et al. [72], which is popular benchmark test instances commonly used in the literature. Table C.1 gives the characteristics of the TCHVRP instances we have generated. In this table, we give the number N of customers and, for each vehicle type k, the number M_k of vehicles available, the variable cost factor α_k and time factor β_k, error terms for cost ϵ_α and time ϵ_β, and the number of instances. As global parameter settings, we use the constant capacity Q and loading time L over all vehicles. Finally, we generate three different data sets for TCHVRP: single fleet type with Pareto cost, heterogeneous fleet with Pareto cost, and heterogeneous fleet with non-Pareto cost.

For Pareto instances, we have a fixed cost per mile and distance per hour for each vehicle type. For Non-Pareto instances, we generate the cost per mile (α_{ij}) and distance per hour (β_{ij}) from a normal distribution with the means and standard deviations in Table C.1. More precisely, the travel cost c^k_{ij} and time t^k_{ij} between customers i and j for vehicles of type k are calculated by c^k_{ij} = $\alpha^k_{ij} d_{ij}$ and $t^k_{ij} = \frac{d_{ij}}{\beta^k_{ij}}$ when the travel is performed by a vehicle of type k. These are based off of a Euclidean distance matrix D_{ij} from the geographical data given in Golden et al. [72]. And then, we generate cost and time matrix such that cost = $\alpha_{ij} \times D_{ij}$ and time = $\frac{D_{ij}}{\beta_{ij}}$ given distance matrix D_{ij}.

For the time resources constraints, we have chosen to use a single value Q = 8 hours for all vehicles, and a loading time of $L = 30$ minutes for all customer nodes for all vehicle types. Finally, the error terms α_k and β_k have been chosen in such a way that no vehicle speed is less than 30 mph or more than 60 mph and cost is between 1.25 and 3.75 dollars per mile.

Time & Cost	**Vehicle 1**	**Vehicle 2**	**Vehicle 3**	**Instances**
Single Pareto ($\boldsymbol{\alpha_1}, \boldsymbol{\beta_1}$)	M_1: 6(20 nodes), 9(25 nodes), 12(30,35 nodes) $\boldsymbol{\alpha_1}$: 2 , $\boldsymbol{\beta_1}$: 45	None	None	1
Multiple Pareto ($\boldsymbol{\alpha_t}, \boldsymbol{\beta_t}$)	M_1: 2(20 nodes), 3(25 nodes), 4(30,35 nodes) $\boldsymbol{\alpha_1}$: 2.25 , $\boldsymbol{\beta_1}$: 40	M_2: 2(20 nodes), 3(25 nodes), 4(30,35 nodes) $\boldsymbol{\alpha_2}$: 2 , $\boldsymbol{\beta_2}$: 45	M_3: 2(20 nodes), 3(25 nodes), 4(30,35 nodes) $\boldsymbol{\alpha_3}$: 1.75, $\boldsymbol{\beta_3}$: 50	1
Multiple Non-Pareto ($\boldsymbol{\alpha_t + \epsilon_\alpha}, \boldsymbol{\beta_t + \epsilon_\beta}$)	$\boldsymbol{\epsilon_\alpha}$: $N(0,0.25)$[1.25,3.75] $\boldsymbol{\epsilon_\beta}$: $N(0,7)$[30,60]	$\boldsymbol{\epsilon_\alpha}$: $N(0,0.25)$[1.25,3.75] $\boldsymbol{\epsilon_\beta}$: $N(0,7)$[30,60]	$\boldsymbol{\epsilon_\alpha}$: $N(0,0.25)$[1.25,3.75] $\boldsymbol{\epsilon_\beta}$: $N(0,7)$[30,60]	10

Table C.1: Parameters for VRP instances

BIBLIOGRAPHY

[1] Cheang, B., Li, H., Lim, A., and Rodrigues, B., “Nurse rostering problems—-a bibliographic survey,” *European Journal of Operational Research*, Vol. 151, No. 3, 2003, pp. 447–460.

[2] Burke, E. K., De Causmaecker, P., Berghe, G. V., and Van Landeghem, H., “The state of the art of nurse rostering,” *Journal of scheduling*, Vol. 7, No. 6, 2004, pp. 441–499.

[3] Beaulieu, H., Ferland, J. A., Gendron, B., and Michelon, P., “A mathematical programming approach for scheduling physicians in the emergency room,” *Health care management science*, Vol. 3, No. 3, 2000, pp. 193–200.

[4] Brunner, J. O., Bard, J. F., and Kolisch, R., “Flexible shift scheduling of physicians,” *Health care management science*, Vol. 12, No. 3, 2009, pp. 285–305.

[5] Ozkarahan, I., “A scheduling model for hospital residents,” *Journal of Medical Systems*, Vol. 18, No. 5, 1994, pp. 251–265.

[6] Guo, J., Morrison, D. R., Jacobson, S. H., and Jokela, J. A., “Complexity results for the basic residency scheduling problem,” *Journal of Scheduling*, Vol. 17, No. 3, 2014, pp. 211–223.

[7] Pinedo, M. L., *Scheduling: Theory, Algorithms, and Systems*, Vol. 4, Springer, 2016.

[8] Graves, S. C., “A review of production scheduling,” *Operations research*, Vol. 29, No. 4, 1981, pp. 646–675.

[9] Vance, P. H., Barnhart, C., Johnson, E. L., and Nemhauser, G. L., “Airline crew scheduling: A new formulation and decomposition algorithm,” *Operations Research*, Vol. 45, No. 2, 1997, pp. 188–200.

[10] Ernst, A. T., Jiang, H., Krishnamoorthy, M., Nott, H., and Sier, D., “An integrated optimization model for train crew management,” *Annals of Operations Research*, Vol. 108, No. 1-4, 2001, pp. 211–224.

[11] Raff, S., “Routing and scheduling of vehicles and crews: The state of the art,” *Computers & Operations Research*, Vol. 10, No. 2, 1983, pp. 6369117149195–67115147193211.

[12] Gans, N., Koole, G., and Mandelbaum, A., “Telephone call centers: Tutorial, review, and research prospects,” *Manufacturing & Service Operations Management*, Vol. 5, No. 2, 2003, pp. 79–141.

[13] Taylor, P. E. and Huxley, S. J., "A break from tradition for the San Francisco police: Patrol officer scheduling using an optimization-based decision support system," *Interfaces*, Vol. 19, No. 1, 1989, pp. 4–24.

[14] Li, Y. and Kozan, E., "Rostering ambulance services," *Industrial engineering and management society*, 2009, pp. 795–801.

[15] Fry, M. J., Magazine, M. J., and Rao, U. S., "Firefighter staffing including temporary absences and wastage," *Operations research*, Vol. 54, No. 2, 2006, pp. 353–365.

[16] Edie, L. C., "Traffic delays at toll booths," *Journal of the operations research society of America*, Vol. 2, No. 2, 1954, pp. 107–138.

[17] Brucker, P., Qu, R., and Burke, E., "Personnel scheduling: Models and complexity," *European Journal of Operational Research*, Vol. 210, No. 3, 2011, pp. 467–473.

[18] Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., and De Boeck, L., "Personnel scheduling: A literature review," *European Journal of Operational Research*, Vol. 226, No. 3, 2013, pp. 367–385.

[19] Dantzig, G. B., "Letter to the Editor-A Comment on Edie's "Traffic Delays at Toll Booths"," *Journal of the Operations Research Society of America*, Vol. 2, No. 3, 1954, pp. 339–341.

[20] Ernst, A. T., Jiang, H., Krishnamoorthy, M., and Sier, D., "Staff scheduling and rostering: A review of applications, methods and models," *European journal of operational research*, Vol. 153, No. 1, 2004, pp. 3–27.

[21] Ernst, A. T., Jiang, H., Krishnamoorthy, M., Owens, B., and Sier, D., "An annotated bibliography of personnel scheduling and rostering," *Annals of Operations Research*, Vol. 127, No. 1-4, 2004, pp. 21–144.

[22] Carter, M. W. and Lapierre, S. D., "Scheduling emergency room physicians," *Health Care Management Science*, Vol. 4, No. 4, 2001, pp. 347–360.

[23] Topaloglu, S., "A multi-objective programming model for scheduling emergency medicine residents," *Computers & Industrial Engineering*, Vol. 51, No. 3, 2006, pp. 375–388.

[24] Gendreau, M., Ferland, J., Gendron, B., Hail, N., Jaumard, B., Lapierre, S., Pesant, G., and Soriano, P., "Physician scheduling in emergency rooms," *International Conference on the Practice and Theory of Automated Timetabling*, Springer, 2006, pp. 53–66.

[25] Cardoen, B., Demeulemeester, E., and Beliën, J., "Operating room planning and scheduling: A literature review," *European journal of operational research*, Vol. 201, No. 3, 2010, pp. 921–932.

[26] Levner, E., Kats, V., de Pablo, D. A. L., and Cheng, T. E., "Complexity of cyclic scheduling problems: A state-of-the-art survey," *Computers & Industrial Engineering*, Vol. 59, No. 2, 2010, pp. 352–361.

[27] Vanhoucke, M. and Maenhout, B., “On the characterization and generation of nurse scheduling problem instances,” *European Journal of Operational Research*, Vol. 196, No. 2, 2009, pp. 457–467.

[28] M’Hallah, R. and Alkhabbaz, A., “Scheduling of nurses: a case study of a Kuwaiti health care unit,” *Operations Research for Health Care*, Vol. 2, No. 1, 2013, pp. 1–19.

[29] Rousseau, L.-M., Pesant, G., and Gendreau, M., “A general approach to the physician rostering problem,” *Annals of Operations Research*, Vol. 115, No. 1, 2002, pp. 193–205.

[30] Carrasco, R. C., “Long-term staff scheduling with regular temporal distribution,” *Computer methods and programs in biomedicine*, Vol. 100, No. 2, 2010, pp. 191–199.

[31] Fazio, S. B. and Steinmann, A. F., “A New Era for Residency Training in Internal Medicine,” *JAMA internal medicine*, 2015, pp. 1–2.

[32] Shanafelt, T. and Habermann, T., “Medical residents’ emotional well-being,” *Jama*, Vol. 288, No. 15, 2002, pp. 1846–1847.

[33] Butterfield, P. S., “The stress of residency: a review of the literature,” *Archives of internal medicine*, Vol. 148, No. 6, 1988, pp. 1428–1435.

[34] Schwenk, T. L., Davis, L., and Wimsatt, L. A., “Depression, stigma, and suicidal ideation in medical students,” *Jama*, Vol. 304, No. 11, 2010, pp. 1181–1190.

[35] Sen, S., Kranzler, H. R., Krystal, J. H., Speller, H., Chan, G., Gelernter, J., and Guille, C., “A prospective cohort study investigating factors associated with depression during medical internship,” *Archives of general psychiatry*, Vol. 67, No. 6, 2010, pp. 557–565.

[36] Goldman, M. L., Shah, R. N., and Bernstein, C. A., “Depression and suicide among physician trainees: recommendations for a national response,” *JAMA psychiatry*, Vol. 72, No. 5, 2015, pp. 411–412.

[37] Shanafelt, T. D., Bradley, K. A., Wipf, J. E., and Back, A. L., “Burnout and self-reported patient care in an internal medicine residency program,” *Annals of internal medicine*, Vol. 136, No. 5, 2002, pp. 358–367.

[38] Lockley, S. W., Barger, L. K., Ayas, N. T., Rothschild, J. M., Czeisler, C. A., Landrigan, C. P., et al., “Effects of health care provider work hours and sleep deprivation on safety and performance,” *The Joint Commission Journal on Quality and Patient Safety*, Vol. 33, No. Supplement 1, 2007, pp. 7–18.

[39] Rogers, A. E., “The effects of fatigue and sleepiness on nurse performance and patient safety,” 2008.

[40] Lerner, B. H., “A life-changing case for doctors in training,” 2009.

[41] Philibert, I., Friedmann, P., Williams, W. T., et al., “New requirements for resident duty hours,” *Jama*, Vol. 288, No. 9, 2002, pp. 1112–1114.

[42] Sherali, H. D., Ramahi, M. H., and Saifee, Q. J., "Hospital resident scheduling problem," *Production Planning & Control*, Vol. 13, No. 2, 2002, pp. 220–233.

[43] Franz, L. S. and Miller, J. L., "Scheduling medical residents to rotations: solving the large-scale multiperiod staff assignment problem," *Operations Research*, Vol. 41, No. 2, 1993, pp. 269–279.

[44] Day, T. E., Napoli, J. T., and Kuo, P. C., "Scheduling the resident 80-hour work week: an operations research algorithm," *Current surgery*, Vol. 63, No. 2, 2006, pp. 136–141.

[45] Topaloglu, S., "A shift scheduling model for employees with different seniority levels and an application in healthcare," *European Journal of Operational Research*, Vol. 198, No. 3, 2009, pp. 943–957.

[46] Cohn, A., Root, S., Kymissis, C., Esses, J., and Westmoreland, N., "Scheduling medical residents at boston university school of medicine," *Interfaces*, Vol. 39, No. 3, 2009, pp. 186–195.

[47] Topaloglu, S. and Ozkarahan, I., "A constraint programming-based solution approach for medical resident scheduling problems," *Computers & Operations Research*, Vol. 38, No. 1, 2011, pp. 246–255.

[48] Güler, M. G., İdi, K., Güler, E. Y., et al., "A goal programming model for scheduling residents in an anesthesia and reanimation department," *Expert Systems with Applications*, Vol. 40, No. 6, 2013, pp. 2117–2126.

[49] Güler, M. G., "A hierarchical goal programming model for scheduling the outpatient clinics," *Expert Systems with Applications*, Vol. 40, No. 12, 2013, pp. 4906–4914.

[50] Bard, J. F., Shu, Z., and Leykum, L., "A network-based approach for monthly scheduling of residents in primary care clinics," *Operations Research for Health Care*, Vol. 3, No. 4, 2014, pp. 200–214.

[51] Bard, J. F., Shu, Z., Morrice, D. J., Leykum, L. K., and Poursani, R., "Annual block scheduling for family medicine residency programs with continuity clinic considerations," *IIE Transactions*, Vol. 48, No. 9, 2016, pp. 797–811.

[52] Bard, J. F., Shu, Z., Morrice, D. J., and Leykum, L. K., "Constructing Block Schedules for Internal Medicine Residents," *IISE Transactions on Healthcare Systems Engineering*, , No. just-accepted, 2016.

[53] Hwang, C. and Masud, A., "Multiple objective decision making-methods and applications: a state-of-the-art survey," *Lecture notes in economics and mathematical systems (*, , No. 164, 1979.

[54] Caramia, M. and Dell'Olmo, P., *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms*, Springer Science & Business Media, 2008.

[55] Mavrotas, G., "Effective implementation of the ε-constraint method in multi-objective mathematical programming problems," *Applied mathematics and computation*, Vol. 213, No. 2, 2009, pp. 455–465.

[56] Haimes, Y. Y., Lasdon, L. S., and Wismer, D. A., "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems Man and Cybernetics*, , No. 1, 1971, pp. 296–297.

[57] Bäck, T., Hammel, U., and Schwefel, H.-P., "Evolutionary computation: Comments on the history and current state," *IEEE transactions on Evolutionary Computation*, Vol. 1, No. 1, 1997, pp. 3–17.

[58] Zitzler, E., "Evolutionary algorithms for multiobjective optimization: Methods and applications," 1999.

[59] Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.

[60] Glover, F. and Kochenberger, G. A., *Handbook of Metaheuristics*, Boston, MA: Springer US, 2003.

[61] Rajabioun, R., "Cuckoo optimization algorithm," *Applied soft computing*, Vol. 11, No. 8, 2011, pp. 5508–5518.

[62] Esmaili, M., Amjady, N., and Shayanfar, H. A., "Multi-objective congestion management by modified augmented ε-constraint method," *Applied Energy*, Vol. 88, No. 3, 2011, pp. 755–766.

[63] Marler, R. T. and Arora, J. S., "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, Vol. 26, No. 6, 2004, pp. 369–395.

[64] Arthur, J. L. and Ravindran, A., "A multiple objective nurse scheduling model," *AIIE transactions*, Vol. 13, No. 1, 1981, pp. 55–60.

[65] Musa, A. and Saxena, U., "Scheduling nurses using goal-programming techniques," *IIE transactions*, Vol. 16, No. 3, 1984, pp. 216–221.

[66] Ozkarahan, I. and Bailey, J. E., "Goal programming model subsystem of a flexible nurse scheduling support system," *IIE transactions*, Vol. 20, No. 3, 1988, pp. 306–316.

[67] Dantzig, G. B. and Ramser, J. H., "The truck dispatching problem," *Management science*, Vol. 6, No. 1, 1959, pp. 80–91.

[68] Garvin, W., Crandall, H., John, J., and Spellman, R., "Applications of linear programming in the oil industry," *Management Science*, Vol. 3, No. 4, 1957, pp. 407–430.

[69] Gavish, B. and Graves, S. C., "The travelling salesman problem and related problems," 1978.

[70] Gavish, B. and Graves, S. C., "Scheduling and routing in transportation and distribution systems: formulations and new relaxations," *Management Science (accepted. subject to revision)*, 1982.

[71] Gheysens, F., Golden, B., and Assad, A., "A comparison of techniques for solving the fleet size and mix vehicle routing problem," *Operations-Research-Spektrum*, Vol. 6, No. 4, 1984, pp. 207–216.

[72] Golden, B., Assad, A., Levy, L., and Gheysens, F., "The fleet size and mix vehicle routing problem," *Computers & Operations Research*, Vol. 11, No. 1, 1984, pp. 49–66.

[73] Balinski, M. L. and Quandt, R. E., "On an integer program for a delivery problem," *Operations Research*, Vol. 12, No. 2, 1964, pp. 300–304.

[74] Rao, M. and Zionts, S., "Allocation of Transportation Units to Alternative Trips-A Column Generation Scheme with Out-of-Kilter Subproblems," *Operations Research*, Vol. 16, No. 1, 1968, pp. 52–63.

[75] Foster, B. A. and Ryan, D. M., "An integer programming approach to the vehicle scheduling problem," *Operational Research Quarterly*, 1976, pp. 367–384.

[76] Agarwal, Y., Mathur, K., and Salkin, H. M., "A set-partitioning-based exact algorithm for the vehicle routing problem," *Networks*, Vol. 19, No. 7, 1989, pp. 731–749.

[77] Baldacci, R., Christofides, N., and Mingozzi, A., "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts," *Mathematical Programming*, Vol. 115, No. 2, 2008, pp. 351–385.

[78] Baldacci, R. and Mingozzi, A., "A unified exact method for solving different classes of vehicle routing problems," *Mathematical Programming*, Vol. 120, No. 2, 2009, pp. 347–380.

[79] Baldacci, R., Mingozzi, A., and Roberti, R., "New route relaxation and pricing strategies for the vehicle routing problem," *Operations research*, Vol. 59, No. 5, 2011, pp. 1269–1283.

[80] Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F., "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical programming*, Vol. 106, No. 3, 2006, pp. 491–511.

[81] Juan, A. A., Goentzel, J., and Bektaş, T., "Routing fleets with multiple driving ranges: Is it possible to use greener fleet configurations?" *Applied Soft Computing*, Vol. 21, 2014, pp. 84–94.

[82] Subramanian, A., Penna, P. H. V., Uchoa, E., and Ochi, L. S., "A hybrid algorithm for the heterogeneous fleet vehicle routing problem," *European Journal of Operational Research*, Vol. 221, No. 2, 2012, pp. 285–295.

[83] Clarke, G. u. and Wright, J., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations research*, Vol. 12, No. 4, 1964, pp. 568–581.

[84] Gillett, B. E. and Miller, L. R., "A heuristic algorithm for the vehicle-dispatch problem," *Operations research*, Vol. 22, No. 2, 1974, pp. 340–349.

[85] Glover, F., "Tabu search-part I," *ORSA Journal on computing*, Vol. 1, No. 3, 1989, pp. 190–206.

[86] Kirkpatrick, S., Vecchi, M., et al., "Optimization by simmulated annealing," *science*, Vol. 220, No. 4598, 1983, pp. 671–680.

[87] Holland, J. H., *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press, 1975.

[88] Taillard, É. D., "A heuristic column generation method for the heterogeneous fleet VRP," *RAIRO-Operations Research*, Vol. 33, No. 01, 1999, pp. 1–14.

[89] Gendreau, M., Laporte, G., Musaraganyi, C., and Taillard, É. D., "A tabu search heuristic for the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, Vol. 26, No. 12, 1999, pp. 1153–1173.

[90] Nagata, Y., "Edge assembly crossover for the capacitated vehicle routing problem," *Evolutionary Computation in Combinatorial Optimization*, Springer, 2007, pp. 142–153.

[91] Prins, C., "A GRASP $\times$ evolutionary local search hybrid for the vehicle routing problem," *Bio-inspired algorithms for the vehicle routing problem*, Springer, 2009, pp. 35–53.

[92] Laporte, G., "Fifty years of vehicle routing," *Transportation Science*, Vol. 43, No. 4, 2009, pp. 408–416.

[93] Toth, P. and Vigo, D., *The vehicle routing problem*, Siam, 2001.

[94] Toth, P. and Vigo, D., *Vehicle Routing: Problems, Methods, and Applications*, Vol. 18, SIAM, 2014.

[95] Golden, B. L., Raghavan, S., and Wasil, E. A., *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, Vol. 43, Springer, 2008.

[96] Baldacci, R., Battarra, M., and Vigo, D., "Routing a heterogeneous fleet of vehicles," *The vehicle routing problem: latest advances and new challenges*, Springer, 2008, pp. 3–27.

[97] Center for Healthcare Engineering and Patient Safety (CHEPS), "Shift Scheduling Game," 2017.

[98] Marler, R. T. and Arora, J. S., "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, Vol. 41, No. 6, 2010, pp. 853–862.

[99] Perelstein, E., Rose, A., Hong, Y.-C., Cohn, A., and Long, M. T., "Automation Improves Schedule Quality and Increases Scheduling Efficiency for Residents," *Journal of graduate medical education*, Vol. 8, No. 1, 2016, pp. 45–49.

[100] Saaty, T. L., *What is the analytic hierarchy process?*, Springer, 1988.

[101] Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., and Poole, D., "CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *J. Artif. Intell. Res.(JAIR)*, Vol. 21, 2004, pp. 135–191.

[102] Domshlak, C., Prestwich, S., Rossi, F., Venable, K. B., and Walsh, T., "Hard and soft constraints for reasoning about qualitative conditional preferences," *Journal of Heuristics*, Vol. 12, No. 4-5, 2006, pp. 263–285.

[103] Sylva, J. and Crema, A., "A method for finding the set of non-dominated vectors for multiple objective integer linear programs," *European Journal of Operational Research*, Vol. 158, No. 1, 2004, pp. 46–55.

[104] Jaszkiewicz, A., "A metaheuristic approach to multiple objective nurse scheduling," *Foundations of Computing and Decision Sciences*, Vol. 22, No. 3, 1997, pp. 169–184.

[105] Burke, E. K., Li, J., and Qu, R., "A Pareto-based search methodology for multi-objective nurse scheduling," *Annals of Operations Research*, Vol. 196, No. 1, 2012, pp. 91–109.

[106] Klein, D. and Hannan, E., "An algorithm for the multiple objective integer linear programming problem," *European Journal of Operational Research*, Vol. 9, No. 4, 1982, pp. 378–385.

[107] Lokman, B. and Köksalan, M., "Finding all nondominated points of multi-objective integer programs," *Journal of Global Optimization*, Vol. 57, No. 2, 2013, pp. 347–365.

[108] Lemesre, J., Dhaenens, C., and Talbi, E.-G., "Parallel partitioning method (PPM): A new exact method to solve bi-objective problems," *Computers & operations research*, Vol. 34, No. 8, 2007, pp. 2450–2462.

[109] Dhaenens, C., Lemesre, J., and Talbi, E.-G., "K-PPM: A new exact method to solve multi-objective combinatorial optimization problems," *European Journal of Operational Research*, Vol. 200, No. 1, 2010, pp. 45–53.

[110] Dächert, K. and Klamroth, K., "A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems," *Journal of Global Optimization*, 2015, pp. 1–34.

[111] Laumanns, M., Thiele, L., and Zitzler, E., "An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method," *European Journal of Operational Research*, Vol. 169, No. 3, 2006, pp. 932–942.

[112] Kirlik, G. and Sayın, S., "A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems," *European Journal of Operational Research*, Vol. 232, No. 3, 2014, pp. 479–488.

[113] Boland, N., Charkhgard, H., and Savelsbergh, M., "The L-shape search method for triobjective integer programming," *Optimization Online*, 2014.

[114] Boland, N., Charkhgard, H., and Savelsbergh, M., "A Simple and Efficient Algorithm For Solving Three Objective Integer Programs," *Optimization Online*, 2014.

[115] Özlen, M. and Azizoğlu, M., "Multi-objective integer programming: a general approach for generating all non-dominated solutions," *European Journal of Operational Research*, Vol. 199, No. 1, 2009, pp. 25–35.

[116] Özlen, M., Burton, B. A., and MacRae, C. A., "Multi-objective integer programming: an improved recursive algorithm," *Journal of Optimization Theory and Applications*, Vol. 160, No. 2, 2014, pp. 470–482.

[117] Bérubé, J.-F., Gendreau, M., and Potvin, J.-Y., "An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits," *European Journal of Operational Research*, Vol. 194, No. 1, 2009, pp. 39–50.

[118] Isermann, H. and Steuer, R. E., "Computational experience concerning payoff tables and minimum criterion values over the efficient set," *European journal of operational research*, Vol. 33, No. 1, 1988, pp. 91–97.

[119] Reeves, G. R. and Reid, R. C., "Minimum values over the efficient set in multiple objective decision making," *European Journal of Operational Research*, Vol. 36, No. 3, 1988, pp. 334–338.

[120] Steuer, R. E., "Non-fully resolved questions about the efficient/nondominated set," *Multicriteria analysis*, Springer, 1997, pp. 585–589.

[121] Dantzig, G., Fulkerson, R., and Johnson, S., "Solution of a Large-Scale Traveling-Salesman Problem," *Operations Research*, Vol. 2, No. 4, 1954, pp. 393–410.

[122] Christofides, N., "The vehicle routing problem," *RAIRO-Operations Research-Recherche Opérationnelle*, Vol. 10, No. V1, 1976, pp. 55–70.

[123] Christofides, N., Mingozzi, A., and Toth, P., "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Mathematical programming*, Vol. 20, No. 1, 1981, pp. 255–282.

[124] Laporte, G., Desrochers, M., and Nobert, Y., "Two exact algorithms for the distance-constrained vehicle routing problem," *Networks*, Vol. 14, No. 1, 1984, pp. 161–172.

[125] Bettinelli, A., Ceselli, A., and Righini, G., "A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows," *Transportation Research Part C: Emerging Technologies*, Vol. 19, No. 5, 2011, pp. 723–740.

[126] Christofides, N., Mingozzi, A., and Toth, P., "State-space relaxation procedures for the computation of bounds to routing problems," *Networks*, Vol. 11, No. 2, 1981, pp. 145–164.

[127] Desrochers, M., Desrosiers, J., and Solomon, M., "A new optimization algorithm for the vehicle routing problem with time windows," *Operations research*, Vol. 40, No. 2, 1992, pp. 342–354.

[128] Laporte, G., Nobert, Y., and Desrochers, M., "Optimal routing under capacity and distance restrictions," *Operations research*, Vol. 33, No. 5, 1985, pp. 1050–1073.

[129] Miller, C. E., Tucker, A. W., and Zemlin, R. A., "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, Vol. 7, No. 4, 1960, pp. 326–329.

[130] Choi, E. and Tcha, D.-W., "A column generation approach to the heterogeneous fleet vehicle routing problem," *Computers & Operations Research*, Vol. 34, No. 7, 2007, pp. 2080–2095.

[131] Dantzig, G. B. and Wolfe, P., "Decomposition principle for linear programs," *Operations research*, Vol. 8, No. 1, 1960, pp. 101–111.

[132] Dror, M., "Note on the complexity of the shortest path models for column generation in VRPTW," *Operations Research*, Vol. 42, No. 5, 1994, pp. 977–978.

[133] Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C., "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks*, Vol. 44, No. 3, 2004, pp. 216–229.

[134] Chabrier, A., "Vehicle routing problem with elementary shortest path based column generation," *Computers & Operations Research*, Vol. 33, No. 10, 2006, pp. 2972–2990.

[135] Righini, G. and Salani, M., "Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints," *Discrete Optimization*, Vol. 3, No. 3, 2006, pp. 255–273.

[136] Righini, G. and Salani, M., "New dynamic programming algorithms for the resource constrained elementary shortest path problem," *Networks*, Vol. 51, No. 3, 2008, pp. 155–170.

[137] Lozano, L., Duque, D., and Medaglia, A. L., "An exact algorithm for the elementary shortest path problem with resource constraints," *Transportation Science*, 2015.

[138] Righini, G. and Salani, M., "Dynamic programming algorithms for the elementary shortest path problem with resource constraints," *Electronic Notes in Discrete Mathematics*, Vol. 17, 2004, pp. 247–249.

[139] Martinelli, R., Pecin, D., and Poggi, M., "Efficient elementary and restricted non-elementary route pricing," *European Journal of Operational Research*, Vol. 239, No. 1, 2014, pp. 102–111.

[140] Brunner, J. O. and Edenharter, G. M., "Long term staff scheduling of physicians with different experience levels in hospitals using column generation," *Health care management science*, Vol. 14, No. 2, 2011, pp. 189–202.

[141] Beliën, J. and Demeulemeester, E., "A branch-and-price approach for integrating nurse and surgery scheduling," *European journal of operational research*, Vol. 189, No. 3, 2008, pp. 652–668.

[142] Guerriero, F. and Guido, R., "Operational research in the management of the operating theatre: a survey," *Health care management science*, Vol. 14, No. 1, 2011, pp. 89–114.