UNIVERSITÉ DE MONTRÉAL

DECOMPOSITION-BASED INTEGER PROGRAMMING, STOCHASTIC PROGRAMMING, AND ROBUST OPTIMIZATION METHODS FOR HEALTHCARE PLANNING, SCHEDULING, AND ROUTING PROBLEMS

SEYED HOSSEIN HASHEMI DOULABI
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INDUSTRIEL)
MAI 2017

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

DECOMPOSITION-BASED INTEGER PROGRAMMING, STOCHASTIC PROGRAMMING, AND ROBUST OPTIMIZATION METHODS FOR HEALTHCARE PLANNING, SCHEDULING, AND ROUTING PROBLEMS

présentée par : HASHEMI DOULABI Seyed Hossein
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

DEDICATION

To my wife, Sahar, whose endless love and sacrifice made this journey possible

To my parents who brought me up with their eternal love

ACKNOWLEDGEMENTS

I wholeheartedly appreciate my wonderful advisors, Profs. Gilles Pesant and Louis Martin Rousseau for their excellent guidance, immense knowledge and motivations. They supported me very patiently by giving me the opportunity to explore many different research areas during my PhD. I could not have imagined more supportive advisors for my PhD Study. I consider them as role models in my future career in academia.

Besides my advisors, I would like to thank Prof. Patrick Jaillet for hosting me as a visiting student at *MIT*. It was really a unique opportunity to collaborate with Prof. Jaillet and be a part of his research group. In addition, I would like to thank Profs. Merve Bodur, Michel Gendreau, Bernard Gendron for accepting to join the jury.

Furthermore, I acknowledge *The Fonds de recherche du Québec - Nature et technologies* (*FR-QNT*) for providing full financial scholarships during the second and third year of my PhD study and also another financial scholarship that supported my visit at *MIT*. Likewise, I express my gratitude to *CIRRELT*, and *Institute of Industrial Engineers* (*IIE*) for supporting my research by several scholarships.

I also want to take this opportunity to thank all my great friends at *CIRRELT* for their encouragements.

RÉSUMÉ

Il existe de nombreuses applications de planification, d'ordonnancement et de confection de tournées dans les systèmes de santé. La résolution efficace de ces problèmes peut aider les responsables de la santé à fournir des services de meilleure qualité, en utilisant efficacement les ressources médicales disponibles. En raison de la nature combinatoire de ces problèmes, dans de nombreux cas, les algorithmes de programmation en nombres entiers standards dans les logiciels commerciaux de programmation mathématique tels que CPLEX et Gurobi ne peuvent pas résoudre efficacement les modèles correspondants. Dans cette thèse, nous étudions trois problèmes de planification, d'ordonnancement et de confection de tournées des soins de santé et proposons des approches à base de décomposition utilisant la programmation en nombres entiers, la programmation stochastique et une méthode d'optimisation robuste.

Le premier article de cette thèse présente un problème intégré de planification et d'ordonnancement dans le cadre des salles d'opération. Cette situation implique d'optimiser l'ordonnancement et l'affectation des chirurgies aux différentes salles d'opération, sur un horizon de planification à court terme. Nous avons pris en compte les heures de travail quotidiennes maximales des chirurgiens, le temps de nettoyage obligatoire alloué lors du passage de cas infectieux à des cas non infectieux et le respect des dates limites des chirurgies. Nous avons aussi empêché le chevauchement des chirurgies effectuées par le même chirurgien. Nous avons formulé le problème en utilisant un modèle de programmation mathématique et développé un algorithme «branch-and-price-and-cut» basé sur un modèle de programmation par contraintes pour le sous-problème. Nous avons mis en place des règles de dominance et un algorithme de détection d'infaillibilité rapide. Cet algorithme, basé sur le problème du sac à dos multidimensionnel, nous permet d'améliorer l'efficacité du modèle de programmation de contraintes. Les résultats montrent que notre méthode présente un écart à l'optimum moyen de 2,81%, ce qui surpasse de manière significative la formulation mathématique compacte dans la littérature.

Dans la deuxième partie de cette thèse, pour la première fois, nous avons étudié l'optimisation des problèmes de tournées de véhicules avec visites synchronisées (VRPS) en tenant compte de stochasticité des temps de déplacement et de service. En plus d'envisager un problème d'ordonnancement des soins de santé à domicile, nous introduisons un problème d'ordonnancement des salles d'opération avec des durées stochastiques qui est une nouvelle application de VRPS. Nous avons modélisé les VRPS qui ont des durées stochastiques en programmation stochastique à deux niveaux avec des variables entières dans les deux niveaux. L'avantage du modèle proposé est que, contrairement aux modèles déterministes de la littérature VRPS, il n'a pas de contraintes «big-M». Cet avantage entraine en contrepartie la présence d'un grand nombre de variables entières dans le second niveau. Nous avons prouvé que les contraintes d'intégralité sur les variables du deuxième niveau sont triviales ce qui nous permet d'appliquer l'algorithme «L-shaped» et son implémentation branch-and-and-cut pour résoudre le problème. Nous avons amélioré le modèle en développant des inégalités valides et une fonction de bornes inférieures. Nous avons analysé les sous-problèmes de l'algorithme en L et nous avons proposé une méthode de résolution qui est beaucoup plus rapide que les algorithmes de programmation linéaire standards. En outre, nous avons étendu notre modèle pour modéliser les VRPS avec des temps de déplacement et de service dépendant du temps.

Les résultats de l'optimisation montrent que, pour le problème stochastique de soins à domicile, l'algorithme «branch-and-cut» résout à l'optimalité les exemplaires avec 15 patients et 10% à 30% de visites synchronisées. Il trouve également des solutions avec un écart à l'optimum moyen de de 3,57% pour les cas avec 20 patients. De plus l'algorithme «branch-and-cut» résout à l'optimalité les problèmes d'ordonnancement stochastique des salles d'opération avec 20 chirurgies. Ceci est une amélioration considérable par rapport à la littérature qui fait état de cas avec 11 chirurgies. En outre, la modélisation proposée pour le problème dépendant du temps trouve des solutions optimales pour d'une grande portion des exemplaires d'ordonnancement de soins de santé à domicile avec 30 à 60 patients et différents taux de visites synchronisées.

Dans la dernière partie de cette thèse, nous avons étudié une catégorie de modèles d'optimisation robuste en deux étapes avec des variables entières du problème adversaire. Nous avons analysé l'importance de cette classe de problèmes lors de la modélisation à deux niveaux de problèmes de planification de ressources robuste en deux étapes où certaines tâches ont des temps d'arrivée et des durées incertains. Nous considérons un problème de répartition et d'affectation d'infirmières comme une application de cette classe de modèles robustes. Nous avons appliqué la décomposition de Dantzig-Wolfe pour exploiter la structure de ces modèles, ce qui nous a permis de montrer que le problème initial se réduit à un problème robuste à une seule étape. Nous avons proposé un algorithme Benders pour le problème reformulé. Étant donné que le problème principal et le sous-problème dans l'algorithme Benders sont des programmes à nombres entiers mixtes, il requiert une quantité de calcul importante à chaque itération de l'algorithme pour les résoudre de manière optimale. Par conséquent, nous avons développé de nouvelles conditions d'arrêt pour ces programmes à nombres entiers mixtes et fourni des preuves de convergence. Nous avons développé également un algorithme heuristique appelé «dual algorithm». Dans cette heuristique, nous dualisons la relaxation linéaire du problème adversaire dans le problème reformulé et générons des coupes itérativement pour façonner l'enveloppe convexe de l'ensemble d'incertitude. Nous avons combiné cette heuristique avec l'algorithme Benders pour créer un algorithme plus efficace appelé algorithme «Benders-dual algorithm». De nombreuses expériences de calcul sur le problème de répartition et d'affectation d'infirmières sont effectuées pour comparer ces algorithmes.

ABSTRACT

There are many applications of planning, scheduling, and routing problems in healthcare systems. Efficiently solving these problems can help healthcare managers provide higher-quality services by making efficient use of available medical resources. Because of the combinatorial nature of these problems, in many cases, standard integer programming algorithms in commercial mathematical programming software such as CPLEX and Gurobi cannot solve the corresponding models effectively. In this dissertation, we study three healthcare planning, scheduling, and routing problems and propose decomposition-based integer programming, stochastic programming, and robust optimization methods for them.

In the first essay of this dissertation, we study an integrated operating room planning and scheduling problem that combines the assignment of surgeries to operating rooms and scheduling over a short-term planning horizon. We take into account the maximum daily working hours of surgeons, prevent the overlapping of surgeries performed by the same surgeon, allow time for the obligatory cleaning when switching from infectious to noninfectious cases, and respect the surgery deadlines. We formulate the problem using a mathematical programming model and develop a branch-and-price-and-cut algorithm based on a constraint programming model for the subproblem. We also develop dominance rules and a fast infeasibility-detection algorithm based on a multidimensional knapsack problem to improve the efficiency of the constraint programming model. The computational results show that our method has an average optimality gap of 2.81% and significantly outperforms a compact mathematical formulation in the literature.

As the second essay of this dissertation, for the first time, we study vehicle routing problems with synchronized visits (VRPS) and stochastic/time-dependent travel and service times. In addition to considering a home-health care scheduling problem, we introduce an operating room scheduling problem with stochastic durations as a novel application of VRPS. We formulate VRPS with stochastic times as a two-stage stochastic programming model with integer variables in both stages. An advantage of the proposed model is that, in contrast to the deterministic models in the VRPS literature, it does not have any big-M constraints. This advantage comes at the cost of a large number of second-stage integer variables. We prove that the integrality constraints on second-stage variables are trivial, and therefore we can apply the L-shaped algorithm and its branch-and-cut implementation to solve the problem. We enhance the model by developing valid inequalities and a lower bounding functional. We analyze the subproblems of the L-shaped algorithm and devise a solution method for them that is much faster than standard linear programming algorithms. Moreover, we extend our model to formulate VRPS with time-dependent travel and service times.

Computational results show that, in the stochastic home-health care scheduling problem, the branch-and-cut algorithm optimally solves instances with 15 patients and 10% to 30% of synchronized visits. It also finds solutions with an average optimality gap of 3.57% for instances with 20 patients. Furthermore, the branch-and-cut algorithm optimally solves stochastic operating room scheduling problems with 20 surgeries, a considerable improvement over the literature that reports on instances with 11 surgeries. In addition, the proposed formulation for the time-dependent problem solves a large portion of home-health care scheduling instances with 30 to 60 patients and different rates of synchronized visits to optimality.

For the last essay of this dissertation, we also study a class of two-stage robust optimization models with integer adversarial variables. We discuss the importance of this class of problems in modeling two-stage robust resource planning problems where some tasks have uncertain arrival times and duration periods. We consider a two-stage nurse planning problem as an application of this class of robust models. We apply Dantzig-Wolfe decomposition to exploit the structure of these models and show that the original problem reduces to a single-stage robust problem. We propose a Benders algorithm for the reformulated single-stage problem. Since the master problem and subproblem in the Benders algorithm are mixed integer programs, it is computationally demanding to solve them optimally at each iteration of the algorithm. Therefore, we develop novel stopping conditions for these mixed integer programs and provide the relevant convergence proofs. We also develop a heuristic algorithm called dual algorithm. In this heuristic, we dualize the linear programming relaxation of the adversarial problem in the reformulated problem and iteratively generate cuts to shape the convex hull of the uncertainty set. We combine this heuristic with the Benders algorithm to create a more effective algorithm called Benders-dual algorithm. Extensive computational experiments on the nurse planning problem are performed to compare these algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

There is a wide range of operations research applications in healthcare. Researchers have applied various operations research methods in disease prevention, disease screening and surveillance, treatment, and designing and organization of healthcare systems (Denton et al. 2011). The recent area, also referred to as healthcare system management, focuses on providing high quality health services to patients as economically as possible. In the remainder of this chapter, we provide three sections to categorize different problems in operating room, home-health care and nurse-staffing management. These categorizations help readers have better insights about the healthcare planning, scheduling and routing that we have studied in this dissertation.

1.1 Categorization of operating room management

Around 60–70% of patients are admitted to hospitals for some type of surgical operation (Guerriero and Guido 2011). Therefore, efficient management of operating theatres, which include operating and recovery rooms, is very important to hospital managers.

As suggested by Guerriero and Guido (2011), we can categorize problems in operating room management into three groups :

1. *Case mix planning* : This is the strategic stage of operating room management where the total available operating room time over a long term (usually a year) is allocated to surgical departments. The main objective in this stage is to minimize the total deviations from a target time allocation to departments or to maximize the total weighted number of scheduled surgeries.
2. *Master surgery scheduling* : After dividing the total available operating room time to surgical departments, a cyclic schedule, over a medium term planning horizon, must be determined. The schedule that is referred to as *master surgery schedule* determines the number of daily available operating rooms, work hours of operating rooms and the priority of surgeons or surgical departments in using them. This stage is the tactical level of operating room management and the obtained *master surgery schedule* does not change during the first three months after its creation.
3. *Elective case scheduling* : This stage is the operational level of operating room management where after deciding about the *master surgical schedule*, surgeries must be scheduled on a daily basis. In this stage, we determine all details required to perform

a surgery including the corresponding surgeon, the start time of the surgery, required equipments, etc. This stage can be divided itself into the following two steps :

(a) *Advance scheduling (Operating room planning)* : Surgeries are assigned to particular operating rooms and days.

(b) *Allocation scheduling (Operating room scheduling)* : Surgeries are sequenced and scheduled in operating rooms.

In addition to the above definitions, there are two strategies affecting decision making in the tactical and operational levels :

1. *Block Scheduling Strategy* : In this strategy, a set of time blocks are reserved for surgeons or surgical departments to perform their surgeries. Each block time is a time interval in a specific day in a specific operating room. In this strategy, it is assumed that over the planning horizon, time blocks are reserved for surgeons (surgical groups) and they cannot be used by other surgeons even though they remains unused because of unpredicted events.
2. *Open Scheduling Strategy* : In this strategy, surgeries from different surgical departments can be performed in the same time block in the same operating room. Surgeons must choose the surgeries to perform the next day, and the operating theatre manager will schedule those surgeries in one or more rooms. This is a relaxation of the block-scheduling strategy, and it is expected to give a more efficient schedule. However, the management of such schedules is more difficult because of unexpected events such as emergency cases and the stochastic nature of surgical durations.

In the first article presented in Chapter 4, we focus on the operational level of the operating room management and study an integrated operating room planning and scheduling problem under an open scheduling strategy. Moreover, one of the healthcare applications that we have studied in the second article in Chapter 5 is an operating room scheduling problem (allocation scheduling).

1.2 Categorization of home-health care management

Considering the increase in aged populations around the world and their health service needs, the home-health care industry has grown significantly in the past decades. The main goal of companies in this industry is to provide high quality health services to elderly people and also to patients recovering from injuries or illnesses at their homes in the most cost-effective way.

According to Hulshof et al. (2012), decision making in home-health care services can be categorized as follows :

1. Strategic stage : At this stage, the following strategic decisions must be made :
    (a) *Regional Coverage* : In this step, the home-health care organization must decide about the number, types and locations of home-health care centers.
    (b) *Service mix* : The home-health care organization must decide about the types of services to provide. Home care includes helping patients with their basic personal needs such as walking, feeding, and dressing, while home-health care is involved in providing health services such as wound dressing, administering medication, monitoring the general health of the patient, pain control, and other health supports.
    (c) *Case mix* : Based on the output of the *service mix* step, the home-health care organization must decide about the types and the approximate number of patients to serve.
    (d) *Districting* : In this step, the region is partitioned into a number of districts. Patients in each district are usually assigned to one care team. The goal of districting is to limit the travel distances and times for caregivers.
    (e) *Capacity dimensioning* : For each skill category of caregivers, the organization decides if it must employ staff or hire them temporarily from other agencies. Similar decisions must be made about the different types of equipment that caregivers need for serving patients.
2. Tactical stage : This stage is divided into the following steps :
    (a) *Capacity allocation* : Resource capacities fixed in the *Capacity dimensioning* step are allocated to different patient groups over identified districts. The goal in this step is to fairly distribute workload, access times and quality of care for caregivers and patients.
    (b) *Admission control* : The home-health care organization must determine the rule based on which patients are admitted. The main objective of admission control policies is to select patients such that access times are minimized while resources are used as efficiently as possible. Admission control policies are determined based on various factors such as patient locations, the degree of urgency, the required weekly number of visits, and service times.
    (c) *Staff-shift scheduling* : In this step, the organization decide about the start and end times of different shifts and determines the number of caregivers of different types for working in various shifts in each district.

3. Operational stage : This stage is divided into the following steps :
   (a) *Staff-to-shift assignment* : The organization assigns caregivers to shifts over a medium planning horizon (several weeks). The goal of this step is to meet the staffing levels fixed in the staff-shift scheduling step in the tactical level, while satisfying constraints corresponding to work regulations and caregivers' preferences.
   (b) *Visit scheduling* : In this step, the home-health care organization determines all details required for visiting patients over the scheduling horizon. This step is divided into the following sub-steps :
      i. *Short-term care plan* : For each patient, the organization must decide about the visit days and times.
      ii. *Staff-to-visit assignment* : Specific caregiver(s) are assigned to each visit.
      iii. *Routing* : For each caregiver, the sequence of patients to be visited is determined.

In the second article presented in Chapter 5, we study a vehicle routing and scheduling problem with applications in operating rooms and home-health care scheduling problems. The home-health care scheduling problem that we define in the second article is an integration of the *Staff-to-visit assignment* and *Routing* steps presented in the previous categorization.

1.3 Categorization of nurse-staffing management

Human resource management is one of the most critical decision making areas in hospitals. Nurse staffing has been the subject of many studies in the literature. This is because nurses' salaries form a major part of hospitals' budgets and therefore it is very important for hospital managers to have an efficient nurse-staffing plan in order to run hospitals in a cost-effective manner.

According to Punnakitikashem et al. (2008), we can consider four phases of nurse planning and scheduling in hospitals as follows :

1. *Nurse budgeting* : Financial analysts allocate budgets for hiring nurses and determine how many permanent and temporary nurses must be hired.
2. *Nurse rostering* : The nurse manager creates a schedule that determines on which days and which shifts each nurse must be working in the hospital. In this scheduling problem, there are many constraints corresponding to work regulations and demand satisfaction. A prerequisite of this step is *data modeling* in which a data analyst forecasts the number of patients that will enter hospital units over the next four to six weeks.

3. *Nurse rescheduling* : A rescheduling may happen 90 minutes before the start of each shift. A nurse supervisor compares the number and skill levels of scheduled nurses with the expected workload in the shift and if there is a shortage ask on-call nurses to work in the shift.
4. *Nurse assignment* : At the beginning of each shift, the nurse supervisor assigns each patient to a nurse such that 1) the nurse's skill level is appropriate for serving the patient, and 2) the workload of nurses is balanced.

In the third article presented in Chapter 6, we study a two-stage robust optimization problem with an application in general resource planning problems. As an application of the developed model, we study a two-stage nurse planning problem in a hospital's ward based on an uncertain number of patients brought from ICUs to the ward. *Nurse planning* refers to the part of *nurse rostering* where, based on the data obtained from *data modeling*, the daily number of nurses required in the planning horizon are determined without deciding on detailed schedules.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we provide literature reviews on operating room planning and scheduling, home-health care routing and scheduling, and nurse planning problems separately.

2.1 Operating room planning and scheduling

In recent years, some researchers have provided precious surveys on operating room management with different perspectives (Cardoen et al. 2010, Guerriero and Guido 2011, May et al. 2011, Samudra et al. 2016). Cardoen et al. (2010) presented a survey by focusing on the problem setting (e.g., objective functions or different types of patients) and the methodological aspects (e.g., solution method or uncertainty). Guerriero and Guido (2011) presented a more useful literature review based on the categorization scheme presented in Section 1.1. May et al. (2011) provided a literature review where papers are categorized to six categories based on dividing the planning horizon into six stages. Samudra et al. (2016) classified papers in the literature considering patient type, objective functions, decisions to make, up- and downstream resources, uncertainty, research methodology and test instances.

In this section, we have used the structure proposed by Guerriero and Guido (2011) to review relevant papers. In addition to papers referred to by Guerriero and Guido (2011), we have studied the relevant papers published after 2011. Since the focus of the first article in this dissertation is on the operational stage of operating room management, we limit the literature review to papers dealing with this stage. In the Section 2.1.1, we review papers addressing both operating room planning and operating room scheduling. Then, in sections 2.1.2 and 2.1.3, we review papers that studied these problems separately.

2.1.1 Integrated operating room planning and scheduling

In papers published by 2012, researchers rarely addressed integrated operating room planning and scheduling due to the intrinsic complexity of the synchronization. Even, in this few number of papers, most authors have proposed two-step methods to solve the planning and scheduling problems rather than an exact method capable of finding the optimal solution in a unified manner (Jebali et al. 2006, Guinet and Chaabane 2003, Fei et al. 2006, 2010, Liu et al. 2011). However, after 2012, there has been an emergent interest in integrated planning and scheduling of operating rooms.

Jebali et al. (2006) developed a two-step method where, in the first step, a mathematical formulation minimizes the sum of daily hospitalization cost, overtime cost, and under time cost by assigning surgeries to days and operating rooms. Then, in the second step, another mixed integer programming formation (MIP) decides about the sequencing of surgeries considering the possibility of modifying the assignment of surgeries to operating rooms. The main advantage of this work is that authors considered many details in scheduling including prevention from surgeon overlapping (a surgeon cannot work in more than one operating room simultaneously), anesthesia and cleaning times, time window constraints for surgeons, and the possibility of blockage in recovery rooms. Because of the complexity of the developed models, they only solved small-sized instances with three operating rooms, four surgeons and four beds in recovery rooms.

Guinet and Chaabane (2003) developed an assignment formulation for the operating room planning problem with side constraints such as surgeon availability, maximum daily workloads for surgeons, and maximum overtime hours in operating room. This model minimizes the sum of hospitalization cost (waiting time of patients) and overtime cost. They also proposed a primal-dual heuristic based on a path augmenting method in a bipartite graph to assign surgeries to operating rooms in the planning horizon. Then, they rescheduled patients on each day in order to satisfy other staff and material constraints. It is possible that the solution obtained by this approach violates the no-overlapping constraint for surgeons.

Fei et al. (2006) studied an operating room planning and scheduling problem in the case of a block scheduling strategy that makes the scheduling part of the studied problem simpler since surgeon overlapping will not happen at all. They presented a two-step approach where a column generation algorithm finds some patient assignments for operating rooms. In the scheduling phase, they considered the possibility of blockage due to the limited number of beds in the recovery room that makes the problem similar to a two-stage flow shop. They applied a hybrid genetic algorithm-tabu search method for this phase. The main novelty of this work is the column generation method in which pattern variables decide about the patient assignments to operating rooms. Fei et al. (2010) applied the proposed method for the case of open scheduling strategy where the first step maximizes operating room utilization, and the second step minimizes overtime and under time cost. Liu et al. (2011) developed a heuristic to solve the operating room planning and scheduling problem introduced by Fei et al. (2010). The proposed heuristic relies on the idea of aggregating states in a dynamic programming model in order to deal with the curse of dimensionality.

In contrast to previously mentioned papers, some works in the literature have proposed a unified approach for operating room planning and scheduling simultaneously (Roland et al.

(2006, 2010), Marques et al. (2012), Vijayakumar et al. (2013), Di Martinelly et al. (2014), Molina-Pariente et al. (2015a), Zhou et al. (2016)). To the best of our knowledge, Roland et al. (2006) is the first work that formulated the operating room planning and scheduling problem in a unified way. Inspired from resource constrained project scheduling models, they proposed an MIP model based on four-index binary variables to minimize the sum of operating rooms fixed cost and overtime cost. The proposed four-index binary variables determine the assignment of operating rooms, surgery dates, and start times to surgeries. Roland et al. (2010) extended this model by considering different types of human resource constraints including surgeons and nurses. They showed that the proposed formulation is capable of solving only small-sized instances. In both previous papers, genetic algorithms were applied to solve the problem due to the intractable nature of the developed formulations.

Marques et al. (2012) used the same four-index variables to formulate the integrated operating room planning and scheduling. They applied this formulation to study a real case in a hospital in Lisbon. They considered several types of priorities for patients. Because of the huge dimension of the MIP model, they scheduled emergency surgeries first, and then decided about other surgeries. Finally, they applied a heuristic algorithm to improve the obtained solution. One of the drawbacks of this work is that, to reduce the number of variables, the scheduling horizon is discretized to 15-min time slots, which could be a source of inaccuracy in the obtained solution. Most researchers considered 5-minute time slots in operating room scheduling and planning (e.g., Lamiri et al. 2008b).

Vijayakumar et al. (2013) considered the operating room planning and scheduling as a bin-packing problem with some side constraints and proposed an MIP formulation based on five-index variables that determine the assignment of patients to surgeons, operating rooms, days, and time slots. This variable definition is different from the one presented by Roland et al. (2006), Roland et al. (2010), and Marques et al. (2012). The developed MIP model is just capable of solving very small instances due to the huge number of variables. Therefore, they proposed a heuristic approach based on the well-known First Fit Decreasing rule in the bin packing problem.

Di Martinelly et al. (2014) developed an MIP model for integrating operating room planning and scheduling with nurse scheduling. To formulate the operating room planning and scheduling part of the model, they used the four-index variable proposed by Roland et al. (2006). They concluded that there is no relation between the required number of nurses and the number of opened operating rooms, while the simultaneous scheduling of operating rooms and nurses decreases other costs including overtime costs and the service costs of nurses. The main deficiency of this work is that due to the large number of four-index variables, each day is discretized to 15-minute time slots, which results in significant inaccuracy in start times of surgeries.

Molina-Pariente et al. (2015a) studied an integrated operating room planning and scheduling problem where teams of surgeons perform surgeries and the surgical durations depend on the teams' experience. They proposed an MIP model that can solve small-sized instances. They also developed an iterative constructive method to solve larger instances.

Zhou et al. (2016) developed an MIP model for an integrated operating room planning and scheduling with limited resources in pre-operative and post-operative stages. In contrast to most models in the literature, they did not use the four-index variable introduced by Roland et al. (2006) and instead used continuous variables for the start and finish times of surgeries. They developed a Lagrangian relaxation algorithm and proposed a branch-and-bound algorithm for solving the subproblem. They also enhanced the proposed Lagrangian algorithm by a lower bound method and some dominance rules.

The main shortcoming of MIPs proposed by Vijayakumar et al. (2013), Molina-Pariente et al. (2015a), and Zhou et al. (2016) is that they involve big-M constraints that result in weak linear programming relaxations and significant slower convergence to the optimal solution in a branch-and-bound algorithm.

Integration of optimization and simulation methods has been another research trend in the area of operating room planning and scheduling (Persson and Persson 2009, M'Hallah and Al-Roomi 2014, Wang et al. 2016b). Persson and Persson (2009) formulated an optimization model to evaluate the effect of a law passed in Sweden stating that no elective surgery can remain in the waiting list for the intervention more than 90 days. They studied a surgery planning problem with consideration of recovery bed constraints in a Swedish hospital, and proposed an MIP and a simulation model. Their computational results showed that the new law would lead to longer waiting times for medium prioritized patients.

M'Hallah and Al-Roomi (2014) applied a simulation approach to evaluate three surgery planning and scheduling strategies in a block scheduling environment. They assumed that in each block surgeries are sequenced in a non-increasing order of their expected surgical times. Their objective function was to minimize the total under time and overtime.

Wang et al. (2016b) applied a discrete event simulation to evaluate the performance of surgical schedules obtained from solving a deterministic MIP model. Within the simulation model, they considered the uncertainty of surgical durations, dynamic arrivals of emergency cases, and limited downstream resources. They progressively solved the deterministic model for each week and evaluated the obtained schedule by the simulation model. They considered surgeries that were cancelled due to overtime and limited downstream resources for rescheduling within next weeks. They applied the proposed approach on a network of collaborating hospitals and demonstrated its superiority to the actual policy used in these hospitals.

2.1.2 Operating room planning

Many researchers have studied operating room planning problems. Most of them applied MIP models to formulate deterministic problems (Ozkarahan 2000, Ogulata and Erol 2003, Perdomo et al. 2006, Lamiri et al. 2008a,b, Fei et al. 2008, 2009, Augusto et al. 2008, 2010, Wang et al. 2014a, Choi and Wilhelm 2014, Jebali and Diabat 2015, Roshanaei et al. 2017, Marques and Captivo 2017. Ozkarahan (2000) studied the sequential allocation of surgeries to operating rooms during the planning horizon. They aimed to optimize some conflicting objectives including maximization of surgeon preferences and operating rooms utilization, and minimization of under time and overtime costs simultaneously. To this end, she developed an MIP model based on assignment variables and set the objective function to be a weighted summation of the deviations from goals. To underline the superiority of the model, she compared the obtained solution with the actual plan used by hospital's administrators in a case study. The main flaw of this work is that under the open scheduling strategy, there is a possibility that an obtained allocation does not result in a feasible schedule with the same cost due to surgeon overlapping.

Ogulata and Erol (2003) presented a hierarchical multiple criteria model based on mathematical programming formulations to maximize balanced distribution of operations among surgical groups and to minimize patients' waiting times, overtimes and under times. They divided decision making into three separate phases and for each phase developed an MIP model. In the first phase, they selected patients considering priorities and arrival dates. In the second phase, they assigned the selected patients to surgical groups. Finally, they determined the surgery dates of patients. They evaluated hierarchical model using data sets in a Turkish hospital.

Fei et al. (2008) studied an operating room planning problem with the objective of minimizing overtime and under time costs and proposed a branch-and-price algorithm. They developed a heuristic algorithm to obtain feasible upper bounds in the branch-and-bound tree. Moreover, they discussed several branching rules and node selection strategies. They evaluated the proposed method by solving some randomly generated instances. Later, Fei et al. (2009) used a slightly modified version of the proposed column generation to solve another operating planning problem with constraints on the maximum workload of surgeons.

Molina-Pariente et al. (2015b) considered an operating room planning problem in a university hospital in Spain. In this problem, the decision maker must assign intervention dates and operating rooms for surgeries on the waiting list. The objective function is to minimize the access time of patients. They implemented 83 heuristic algorithms from the literature and computationally showed that their proposed metaheuristic algorithm outperforms other heuristics.

Roshanaei et al. (2017) studied an operating room planning problem in a network of hospitals where their objective was to minimize the sum of patients waiting times and opening cost of operating rooms and surgical suites. They developed a logic-based Benders' decomposition (LBBD) to solve an integer programming model. They evaluated four different implementations of the proposed LBBD and demonstrated that their best implementation is significantly faster than solving the original integer programming model.

In a few papers in the area of operating room planning, researchers have considered the recovery room capacity constraint (Perdomo et al. 2006, Augusto et al. 2008, 2010). Perdomo et al. (2006) studied an operating theatre scheduling problem and developed a formulation that decides on the start time of surgeries without assigning surgeries to operating rooms. The objective function of the developed model was to minimize the sum of surgery completion times. They proposed a Lagrangian relaxation algorithm and a heuristic model to find lower and upper bounds respectively. Augusto et al. (2008) enhanced the recent work by considering a constraint on the number of transporters between operating and recovery rooms. Augusto et al. (2010) studied the same problem with the extra assumption that when a patient's surgery is finished and there is no available bed in the recovery room, the recovery process starts in the operating room. They modeled the problem as a four-stage flow shop scheduling problem and proposed a Lagrangian relaxation.

Some researchers considered uncertainties in operating room planning problems (Gerchak et al. 1996, Lamiri et al. 2008a,b, Hans et al. 2008, Wang et al. 2014a, Choi and Wilhelm 2014, Jebali and Diabat 2015, Marques and Captivo 2017). Gerchak et al. (1996) studied an operating room planning with stochastic surgery durations. They developed a stochastic dynamic programming model to determine the number of patients to be scheduled in a finite planning horizon. If the number of all available surgeries are more than the number of scheduled surgeries, the model postpones the remaining surgeries to next days by paying some penalties in the objective function. They have considered the problem as an open scheduling where the daily capacity is the sum of total available operating room times.

Lamiri et al. (2008a) studied the planning of elective and emergency cases in a block scheduling environment. This work extended the problem studied by Gerchak et al. (1996) through specifying the surgery date for individuals and also by considering emergency cases. They developed a stochastic model based on Monte Carlo simulation to approximate the expected value of overtime cost. After sampling from the probability distribution of emergency surgeries, they used an MIP model to solve the problem. The output of the proposed model is an assignment for elective cases and keeping some capacity free for emergency cases. The obtained solutions might turn out to be infeasible in practice because the operating rooms are considered as a single resource with a capacity equal to the total available times of operating rooms. Lamiri et al. (2008b) overcame this inaccuracy through developing a stochastic model based on a column generation method for the same problem. In this work, in addition to under time and overtime costs, a second level of overtime cost after a predetermined threshold is considered.

Hans et al. (2008) addressed the assignment of surgeries and planning of slacks to operating rooms in order to free some additional capacity to cope with uncertainty of surgery durations. Their objective function maximizes capacity utilization and at the same time minimizes the risk of overtime. They designed some heuristic algorithms and tested them on some real data provided by Erasmus MC hospital in Netherlands. The main drawback of this work is that it just dealt with expected values of surgery durations instead of considering them as random variables.

Wang et al. (2014a) formulated an operating room planning problem with scenario-based stochastic durations and emergency cases. In their formulation, the model seeks a trade-off between the opening costs of operating rooms and overtime costs. They also included a chance constraint to restrict the level of cancellations. They proposed a column generation algorithm and multiple branching rules. The main issue about their proposed model is that the chance constraint limit the probability of having cancellation without taking into account the level of constraint violations.

Choi and Wilhelm (2014) studied a capacity planning problem where surgeries from different surgical specialties are allocated to operating rooms on different days. The objective function is to minimize the penalties of unplanned surgeries, and under time and over time of operating rooms. They proposed a stochastic programming model and developed four simplified versions of it, which are more convenient in numerical computations.

Jebali and Diabat (2015) formulated an operating room planning problem with stochastic surgery durations as a two-stage stochastic programming model. In the proposed model, they considered bed constraints in ICU and wards. They used sample average approximation to solve the problem aiming to minimize the waiting, under time, and overtimes costs.

Marques and Captivo (2017) proposed an MIP model for an operating room planning problem. The decisions considered in their model are 1) the selection of patients from a waiting list and 2) the assignment of dates, operating rooms, and time blocks to them. As an extension, they proposed a robust optimization approach to deal with uncertain surgery durations. They applied the proposed methods in a Portuguese hospital.

2.1.3 Operating room scheduling

Most authors have considered operating room scheduling problems with deterministic parameters (Sier et al. 1997, Hsu et al. 2003, Cardoen et al. 2009a,b, Zhong et al. 2014, Wang et al. 2015, Monteiro et al. 2015, Hachicha and Mansour 2016, Latorre-Núñez et al. 2016).

Sier et al. (1997) was one of the first works that addressed the scheduling and sequencing of patients in operating rooms. They considered a weighted multi-objective function taking into account patients age, availability of equipment in operating rooms and possible overlapping of schedules in operating rooms. They modeled the objective function as a nonlinear integer formulation and proposed a simulated annealing algorithm.

Hsu et al. (2003) studied patients scheduling in an ambulatory surgical center. They considered the patient scheduling process as a two-stage no-wait flow shop and minimized the number of nurses in the post-anesthesia care unit and the makespan of the last recovered patient. They assumed that the number of required nurses depends on the peak of patients in the recovery room. They also supposed that surgeons' schedules are back to back with no idle time. To solve the problem, firstly, authors presented a greedy heuristic to find an initial solution, and then applied a tabu search method to improve the solution. They evaluated the quality of obtained solutions by comparing them with actual schedules in a hospital university.

Cardoen et al. (2009a) considered surgery sequencing in a block scheduling environment and proposed an MIP model with a multi-objective function inspired from a real case in Belgium. In the objective function, they considered criteria such as giving priority to children and patients that had cancellations before, penalties for scheduling some surgeries before a predetermined time due to the travel time of patients, overtime of staff in recovery rooms after the closing time of the day-care center, and peaks of bed usages in post-anesthesia care units. Cardoen et al. (2009b) presented a branch-and-price algorithm for the same problem. They developed a dynamic programming method for the pricing phase that constructs operating room schedules. They also proposed some speeding-up techniques to improve the branch-and-price algorithm.

Zhong et al. (2014) studied an operating room scheduling problem as a multi-machine scheduling problem by considering surgeons, nurses, anesthesiologists and surgical equipment as machines. They proposed a two-stage heuristic approach that, in the first stage, assigns surgeries to operating rooms using the longest processing time rule. Then, in the second stage, the proposed algorithm sequences surgeries in each operating room based on the weighted shortest processing time rule in order to minimize the weighted completion times of surgeries.

Wang et al. (2015) addressed an operating room scheduling problem considering patient priority, affinities between surgical team members, different types of resources, and availability of surgical teams. They proposed a constraint programming model and also a MIP model and compared them in a real case study from a Belgian University Hospital. They concluded that the MIP model provides better solutions in the case of weighted sum objective functions than makespan minimization, while the constraint programming model behaves conversely.

Monteiro et al. (2015) studied a new multi-objective function in operating room scheduling. This multi-objective function aims to increase the affinities between surgical team members and at the same time tries to increase the range of skills acquired by nurses in a long term. They proposed an MIP model to formulate and solve the problem.

Hachicha and Mansour (2016) considered a scheduling problem in a private healthcare facility. They considered the problem as three-stage flow shop scheduling problem. In the first-stage, patients are admitted and prepared for surgeries. The second stage starts when patients are transferred to operating rooms, and in the third stage, patients are allocated to hospital beds. They developed two MIP models for this problem. Similar to many other papers in this area, the main deficiency of this work is that there are many big-M constraints in the proposed formulations and therefore MIPs stopped before finding optimal solutions in many instances.

Latorre-Núñez et al. (2016) addressed a daily operating room scheduling problem considering operating rooms, post-anesthesia recovery and the arrival of emergency cases. They developed an MIP model, a constraint programming model, and several heuristic algorithms.

A few researchers have studied the operating room scheduling with some stochastic surgery durations (Denton et al. 2007, Batun et al. 2011, Mancilla and Storer 2012, Saadouli et al. 2015, Xiao et al. 2016). Denton et al. (2007) studied the effect of surgery sequencing in a single operating room on a particular day. Considering waiting, idling and overtime costs in the objective function, they presented a two-stage stochastic programming model and showed that determining the start time of surgeries is of essence and can improve a surgery plan significantly. Their two-stage stochastic programming model includes sequencing variables that increase the complexity of the problem. Therefore, they proved the optimal policy for a single operating room scheduling problem with two surgical cases and then, inspired from this special optimal case, they developed some heuristics for surgery sequencing.

Batun et al. (2011) developed an interesting two-stage stochastic programming model for an operating room scheduling problem. The objective function included opening cost of operating rooms, waiting times of surgeons and the overtime costs. They proposed an L-shaped algorithm and enhanced it by developing a lower bounding functional.

Mancilla and Storer (2012) extended Denton et al. (2007) through presenting position-based variables to determine the schedule of the single operating room. They proposed a heuristic solution approach based on Benders decomposition which outperformed the simple heuristic proposed by Denton et al. (2007).

Saadouli et al. (2015) proposed a two-step algorithm for an operating room scheduling problem with uncertainty in surgery durations and recovery times. In the first step, while considering slack times as a part of surgery durations, they solved a bin packing model in order to select some surgeries for scheduling on a selected day. Then, in the second step, they applied an MIP model to assign selected surgeries to operating rooms. Finally, they compared the proposed approach with the current approach in a hospital in Tunisia and showed some improvements.

Xiao et al. (2016) considered a single operating room scheduling problem and proposed a stochastic programming model and an L-shaped algorithm as the solution method. The main novelty of this work is that they have considered the possibility of cancellations.

Some researchers used simulation methods to evaluate surgery sequencing in operating rooms (Marcon and Dexter 2006, Arnaout and Kulbashian 2008, Saremi et al. 2013). Marcon and Dexter (2006) studied the effect of various scheduling rules taken from machine scheduling on the performance of hospitals such as operating rooms utilization, waiting time of patients for recovery beds, and the required number of staff in the recovery room. They limited this research to the case that surgeons work in only one operating room on each day. Using discrete event simulation, they showed that the most significant effect of sequencing is on the waiting time for recovery beds. The other interesting output of their work is to demonstrate that the well-known longest processing time rule results in a high quality utilization of operating rooms but requires more nurses in recovery rooms during a week.

Similarly, Arnaout and Kulbashian (2008) addressed the evaluation of scheduling rules in surgery sequencing. They modelled the operating theatre scheduling as a parallel machine scheduling with the objective of minimizing the makespan. They compared the well-known shortest processing time and longest processing time rules with a new one developed by themselves and showed the superiority of the last one using simulation. The main drawback of this work is that it has not considered the possibility of surgeon overlapping in different operating rooms.

Saremi et al. (2013) studied a multi-stage operating room scheduling problem with stochastic service times. In this problem, they considered multiple classes of patients, the availability of multiple resources, compatibility of patients and surgeons, and time window constraints for surgeons. They proposed three simulation-based optimization method. The main difference of this work with the others in the literature is that they have assumed that patients from the same class have similar service times. Moreover, as another simplification they have aggregated the total available times of operating rooms as the available capacity for performing surgeries.

2.2 Home-health care routing and scheduling

Fikar and Hirsch (2017) is the only survey that specifically addresses home-health care routing and scheduling problems. They categorized papers in the literature into two groups ; papers that studied single period and those that considered multi-period problems. Within each category, they discussed different objectives, constraints and solution methodologies. Moreover, Castillo-Salazar et al. (2016) provided a literature review on workforce scheduling and routing problems (WSRP) and shortly discussed some characteristics of home-health care problems as an example of WSRP. In the following, we review some of the most significant papers in home-health care routing and scheduling.

Eveborn et al. (2006) focused on staff scheduling in the public home-health system in Sweden. They formulated the problem as a set partitioning model and repetitively used a matching algorithm to assign visiting schedules to staff. In generating staff routes, they considered many constraints including time windows constraints, qualification of caregivers for assigned visits, and planned breaks for staff members.

Akjiratikarl et al. (2007) proposed a particle swarm algorithm for the scheduling of caregivers in the United Kingdom, where local authorities are responsible for providing home care services. In this problem, they aimed to minimize the total traveled distance providing that constraints on time windows and working hours must be satisfied.

Trautsamwieser et al. (2011) addressed a daily home care scheduling in the case of natural disasters. They formulated the problem as an MIP model minimizing the weighted sum of driving and waiting times, and the dissatisfaction levels of clients and nurses. The proposed model is only capable of solving small-sized instances. Therefore, they devised a variable neighbourhood search algorithm for larger instances. They evaluated the proposed algorithms on data sets from Austrian Red Cross.

Hiermann et al. (2015) studied a multimodal home-healthcare scheduling problem in an Austrian home-health care provider. In this problem, “multimodal” refers to different transportation modes that caregivers can use to travel. The objective was to assign caregivers to patients and find efficient routes for them while addressing caregivers and patients preferences. They proposed a two-step method where, in the first step, a constraint programming algorithm finds initial solutions and, in the second-step, four metaheuristic algorithms improve the solutions.

Mankowska et al. (2014) addressed a daily home-health care routing and scheduling problem with interdependent services. In home-health care context, interdependent services refer to services that caregivers must provide to patients simultaneously or with some time lags. They also took into account individual service requirements of patients, qualifications of nurses. They proposed an MIP model to minimize the sum of total traveled distance, and total tardiness in serving patients. They also proposed some constructive, local search, and variable neighborhood search heuristics to find solutions for large instances.

Bowers et al. (2015) studied routing and scheduling of midwives to visit mothers at homes. They emphasized that the continuity of case by the same midwife ensures a better relationship between mothers and health staff. They applied a variant of a multiple traveling salesman algorithm incorporating staff and mother preferences.

Braekers et al. (2016) studied a home care routing and scheduling problem considering many practical details such as skills, working regulations and overtime for nurses, travel costs depending on the transportation mode, hard time windows, and patients’ preferences on nurses and visit times. They proposed a bi-objective MIP model to minimize operational costs and maximize the service level simultaneously. For solving large instances, they also devised a metaheuristic algorithm that applies a large neighborhood search algorithm in a multi-directional local search structure.

Some authors treated home-health care routing and scheduling as a vehicle routing problem (VRP) and formulated the problem in a VRP setting (Bredström and Rönnqvist 2008, Rasmussen et al. 2012, Mısır et al. 2015). Bredström and Rönnqvist (2008) studied a vehicle routing and scheduling problem with temporal precedence and synchronization constraints. The temporal constraint means that, for some customers, more than one visit is required and there must be a precedence order with time lags between them. They considered the application of this problem in the home-health care context and proposed an MIP and a heuristic solution algorithm.

Rasmussen et al. (2012) considered a home care scheduling problem with temporal, and soft preference constraints as a generalization of vehicle routing problem with time windows. They formulated the problem as a set partitioning problem and proposed a branch-and-price algorithm. In the proposed algorithm, temporal constraints are enforced within the branching procedure. Considering the soft preference constraints, they also introduced a visit clustering approach and showed that it decreases run time significantly.

Mısır et al. (2015) evaluated the performance of generalized heuristics in solving routing and rostering problem. They considered a home-health care routing and scheduling problem as one of the relevant problems in this category. In this problem, they emphasized on the patients preferences to nurses and vice versa. They also considered task synchronization for serving some patients and referred to it as “connected activities”. All constraints that they considered in this problem are soft constraints.

Some researchers extended the home-health care routing and scheduling by considering the weekly planning of visits (Nickel et al. 2012, Shao et al. 2012, Bard et al. 2014a,b, Trautsamwieser and Hirsch 2014, Cappanera and Scutellà 2015). Nickel et al. (2012) addressed a home-health care planning and scheduling in Germany. In the planning horizon, they decide on the visiting periods for patients during a week. In the scheduling horizon, they fix the staff routes and schedules. They also took into account the fact that operational decisions must be consistent as much as possible with the master schedule already fixed for a medium term. In this work, the objective function minimizes the weighted sum of the number of unscheduled patients, the patient-nurse loyalty penalty, caregivers’ overtimes, and the total traveled distance. They proposed metaheuristic algorithms that were combined with a constraint programming model.

Shao et al. (2012) studied home-care planning, scheduling, and routing problem where a number of multi-skilled therapists visits patients over a week. The objective was to find weekly schedules for therapists such that the travel and administrative costs were minimized. They formulated the problem as an MIP model and because of its failure they devised a greedy randomized adaptive search procedure (GRASP) algorithm that generates tours for therapists in parallel. Bard et al. (2014a) studied the same problem and stated that the parallel GRASP developed by Shao et al. (2012) failed in instances with tight constraints. They addressed this issue by devising a GRASP algorithm that generates tours sequentially. Bard et al. (2014b) took a step forward and developed a branch-and-price-and-cut algorithm for the same problem. Their algorithm finds near optimal solutions for instances with up to 162 visits and 5 therapists.

Trautsamwieser and Hirsch (2014) addressed a weekly home-health care planning and scheduling problem considering working regulations such as breaks, maximum daily working time, and weekly rest times. They developed an MIP model. Then they reformulated it to a master problem and subproblems framework and applied a branch-and-price solution algorithm. They showed that the proposed branch-and-price algorithm is capable of solving instances with 45 patients, and 203 visits in the week.

Cappanera and Scutellà (2015) proposed an MIP model for simultaneously 1) assigning appropriately skilled operators to patients, 2) scheduling of visits, and 3) routing of operators. They aimed to optimize two balancing objective functions for palliative and terminal patients. The first objective function maximizes the minimum utilization factors of operators, while the second one minimizes the maximum utilization of operators.

Papers cited above studied static home-health care routing and scheduling problems. As an exception, Bennett and Erera (2011) considered a home-health care routing and scheduling where patients arrive dynamically and nurses must visit them several times a week over a predetermined number of weeks. Appointment times for each visit must be selected from a list of available options. Visits must repeat at the same date and times during the service duration. They proposed a rolling horizon myopic planning approach to maximize the number of served patients in a special case that a single nurse serves all patients.

Yuan et al. (2015) is the only paper in the literature that studied home-health care routing and scheduling with stochastic service times. They proposed a two-stage stochastic programming model and then reformulated it to a set partitioning model and applied column generation and branch-and-price solution algorithms. They reported computational results for instances with up to 50 patients.

2.3 Nurse rostering

As nurse planning is a part of nurse rostering and there is not any significant literature addressing nurse planning problems solely, in this section, we provide a literature review on nurse rostering problems. Several researchers have provided surveys focusing on nurse rostering problems (Cheang et al. 2003, Burke et al. 2004). Cheang et al. (2003) reviewed papers in the literature by focusing on the modeling aspects (e.g., decision variables, constraints, and objective functions), and solution approaches (e.g., mathematical programming, heuristics, and artificial intelligence algorithms). Burke et al. (2004) reviewed different steps of nurse staffing and scheduling and then provided a descriptive literature review based on solution methods. In addition, there are some surveys on personnel scheduling problems that discuss nurse rostering problems as one of the relevant applications (Ernst et al. 2004, Brucker et al.

2011, Van den Bergh et al. 2013).

Most researchers proposed heuristic and metaheuristic algorithms for nurse rostering problems (Aickelin and Dowsland 2000, Dowsland and Thompson 2000, Bellanti et al. 2004, Burke et al. 2008, Tsai and Li 2009, Burke et al. 2010b,a, 2011, Lü and Hao 2012, Valouxis et al. 2012, Martin et al. 2013, Wu et al. 2015, Tassopoulos et al. 2015, Rahimian et al. 2017a,b). Aickelin and Dowsland (2000) proposed a genetic algorithm for nurse rostering in a hospital in UK. They used problem-specific knowledge in designing their algorithm. Downsland and Thompson (2000) devised a hybrid heuristic-integer programming algorithm for nurse rostering in a UK hospital. In order to satisfy some rostering constraints, they hybridized their algorithm with a tabu search algorithm, and knapsack and network flow models. Bellanti et al. (2004) developed local and tabu search algorithms for a nurse rostering problem in an Italian hospital where considering holiday planning and labor constraints were essential. Burke et al. (2008) proposed an approach that hybridizes heuristic ordering with variable neighborhood search for solving a nurse rostering problem with commercial data. They computationally showed that their algorithm outperforms a commercial algorithm.

Burke et al. (2010b) presented a multi-objective mathematical programming model and applied a hybrid integer programming (IP)-variable neighborhood search (VNS) as the solution method. In their algorithm, IP solves a relaxed version of the problem with all hard constraints and some soft constraints, while VNS improves the solution obtained from IP. Burke et al. (2010a) developed a scatter search algorithm to minimize the violations of soft constraints. Burke et al. (2011) proposed a hybrid local search and a diversification procedure. Lü and Hao (2012) devised an adaptive neighborhood search with two neighborhood moves and three intensification and diversification search strategies. Martin et al. (2013) proposed a cooperative search algorithm that combines multiple metaheuristics where each of them was useful in optimizing a different fairness objective function.

Wu et al. (2015) proposed an MIP model and a particle swarm optimization (PSO) for a nurse rostering problem. They also proposed a simple procedure for refining the solution obtained from the PSO algorithm. They showed that the PSO algorithm finds optimal solutions in all real instances when the objective function is to maximize the fairness. Rahimian et al. (2017a) developed a hybrid integer and constraint programming approach to solve a nurse rostering problem. They extracted useful information such as the computational difficulty of instances and constraints from the constraint programming model in order to adaptively adjust the search parameters. Later, they proposed a hybrid algorithm combining integer programming and variable neighborhood search for a nurse scheduling problem (Rahimian et al. 2017b). They demonstrated that their algorithm outperforms two state-of the-art algorithms in a recent benchmark of instances.

Some researchers also proposed two-phase metaheuristic algorithms where in the first phase daily workloads of nurses are determined and in the second phase, daily shifts are assigned to nurses (Tsai and Li (2009), Valouxis et al. (2012), Tassopoulos et al. (2015)).

Another group of researchers developed mathematical programming approaches for nurse rostering problems (Jaumard et al. (1998), Bard and Purnomo (2005, 2006), Beliën and Demeulemeester (2008), Maenhout and Vanhoucke (2010), He and Qu (2012)). Jaumard et al. (1998) was among the firsts to apply column generation and branch-and-price algorithms for nurse rostering. In their algorithm, the subproblem deal with individuals constraints (workload, rotation, and days off) while the master problem takes into account demand constraints.

Bard and Purnomo (2005) developed a column generation algorithm with master problem and subproblems that are similar to those proposed by Jaumard et al. (1998). They considered a multi-objective nurse rostering problem and improved the column generation algorithm using a swapping heuristic and a refinement procedure for improving the generated columns. They tested the proposed algorithm on a data set from a hospital in the US.

Bard and Purnomo (2006) studied a nurse rostering problem where agency nurses (temporary nurses) are used to cover shortages. They proposed two MIPs to formulate the problem. One of the models is based on pattern-view formulation, while the other one is developed based on shift-view formulation. The objective in both models is to hire a fixed number of nurses and assign them to nurse schedules such that the maximum amount of uncovered shift in the planning horizon is minimized.

Beliën and Demeulemeester (2008) considered integrated nurse rostering and surgical block assignment to surgeons. In this problem, the surgical block assignment determines the demands in wards that must be covered by appropriate nurse rostering. They proposed column generation and branch-and-price solution algorithms. Maenhout and Vanhoucke (2010) developed a branch-and-price for a multi objective nurse rostering problem. They discussed various branching and pruning strategies.

He and Qu (2012) proposed a constraint-programming-based column generation algorithm for a nurse rostering problem. Because of the flexibility of constraint programming in modeling the column generation subproblems, they have successfully formulated all complex real-word constraints in several benchmarks. They applied Depth Bounded Discrepancy Search and Adaptive Cost Threshold Tightening approach to find high quality columns.

A recent research trend in nurse rostering is based on considering uncertainty in demands and formulating the problem through stochastic optimization approaches (Kim and Mehrotra 2015, Bagheri et al. 2016, Römer and Mellouli 2016). Kim and Mehrotra (2015) studied an integrated nurse staffing and scheduling problem with demand uncertainty. They formulated the problem as a two-stage stochastic programming model. The first-stage decision is to determine the staffing levels and nurse schedules. In the second stage, the model decides on adding and cancelling some shifts in order to accommodate with fluctuation in the demand. They showed that mixed-integer rounding inequalities for the second-stage model convexifies the recourse function and therefore a tight formulation can describe the convex hull of the solution space in the second-stage.

Bagheri et al. (2016) addressed a stochastic nurse rostering problem with uncertainty in patient arrivals and lengths of stays. The formulated the problem as a two-stage stochastic programming model where the first-stage model schedules nurses and the second-stage model adds additional nurses to shifts to cover understaffing. They used a sample average approximation method and applied the proposed method to a hospital in Iran.

Römer and Mellouli (2016) studied a multi-stage nurse rostering problem with demand uncertainty. They considered the fact that the nurse scheduling for the current planning horizon affects the scheduling problem in the next period, particularly in the case of demand uncertainty. They proposed lookahead policies and evaluated the proposed policies by simulation.

Cyclic rostering and rerostering are two other variants of the nurse rostering problem. In a cyclic rostering problem, we assume that nurse schedules cycle over a long period. Bard and Purnomo (2007) addressed a cyclic nurse rostering problem and developed two Lagrangian relaxation algorithms by either relaxing the preference and demand constraints. They theoretically showed that the first setting is not likely to provide good bounds. Therefore, they enhanced the second algorithm through combining it by subgradient optimization, the bundle method, heuristics, and variable fixing. Purnomo and Bard (2007) studied a cyclic nurse rostering problem and developed a branch-and-price algorithm that maximizes nurse's preferences and minimizes the personnel cost. They developed several branching strategies and a rounding heuristic. Maenhout and Vanhoucke (2009) studied the integration of nurse-specific characteristics in cyclic nurse rostering for hospitals in Belgium. They compared their approach with other cyclical and acyclical rostering approaches.

A nurse rerostering problem arises when the roster must be modified to accommodate unexpected staff absence. In this case, the new schedule must be as close as possible to the original one. Moz and Pato (2007) proposed a genetic algorithm for a rerostering problem and implemented it in a public hospital in Portuguese. Maenhout and Vanhoucke (2011) proposed an evolutionary metaheuristic algorithm for this problem. Bäumelt et al. (2016) also studied a nurse rerostering problem and proposed two parallel implementations of a local search algorithm. They showed that the best parallel implementation, called homogeneous algorithm, is between 12.6 to 17.7 times faster than the sequential local search algorithm.

CHAPTER 3 GENERAL ORGANIZATION OF THE DOCUMENT

Our literature review on operating room planning and scheduling shows that, compared to the rich literature in this context, only a few papers addressed integrated operating room planning and scheduling problems. In most of these papers, researchers proposed two-stage algorithms that do not guarantee finding optimal solutions. In the rest of the papers in this area, researchers provided mixed integer programming models that formulate operating room planning and scheduling problems as an integrated problem. However, the main deficiency of these works is that either the proposed models suffer from so-called big-M constraints or they include a large number of integer variables, which both results in non-scalability of these models in problem size. Considering these shortcomings, in Chapter 4, we study an integrated operating room planning and scheduling problem and formulate it as an integer programming model with pattern variables. We propose a constraint-programming-based branch-and-price-and-cut as the solution algorithm and enhance it in several ways. We also computationally compare our algorithm with an integer programming model from the literature and with a pure constraint programming that we formulated.

The literature review in Section 2.1.2 reveals that there is no work in the literature addressing the home-health care scheduling problem with synchronized visits and stochastic/time-dependent travel and service times. In Chapter 5, we study a general class of vehicle routing problems with synchronized visits (VRPS) where travel and service times are stochastic/time-dependent. In addition to considering a home-health care scheduling problem, we introduce an operating room scheduling problem with stochastic durations as a novel application of VRPS. We provide formulations and solution algorithms for the stochastic and the time-dependent home-health care and operating room scheduling problems.

Our literature review in Section 2.1.3 showed that no paper in the literature has addressed a nurse planning or a nurse rostering problem with uncertainty in patient demands by robust optimization approaches. Therefore, in Chapter 6, we consider a general class of two-stage robust optimization models with integer adversarial variables. We discuss the application of this class of problems to two-stage resource planning problems where some tasks have uncertain arrival and duration periods. We consider a nurse planning problem as an example of two-stage resource planning problems. We develop a reformulation approach and several solution algorithms for these models and provide computational experiments for the nurse planning problem.

In Chapter 7, we provide a general discussion on Chapters 4, 5, and 6 and discuss the presented articles as a whole. In Chapter 8, we first present a discussion on the contributions of this dissertation. Then, we explain the limitations of the proposed approaches and future research directions.

CHAPTER 4 ARTICLE 1: A CONSTRAINT-PROGRAMMING-BASED BRANCH-AND-PRICE-AND-CUT APPROACH FOR OPERATING ROOM PLANNING AND SCHEDULING

Seyed Hossein Hashemi Doulabi

Department of Mathematics and Industrial Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Louis-Martin Rousseau

Department of Mathematics and Industrial Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Gilles Pesant

Department of Computer and Software Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Abstract. This paper presents an efficient algorithm for an integrated operating room planning and scheduling problem. It combines the assignment of surgeries to operating rooms and scheduling over a short-term planning horizon. This integration results in more stable planning through consideration of the operational details at the scheduling level, and this increases the chance of successful implementation. We take into account the maximum daily working hours of surgeons, prevent the overlapping of surgeries performed by the same surgeon, allow time for the obligatory cleaning when switching from infectious to noninfectious cases, and respect the surgery deadlines. We formulate the problem using a mathematical programming model and develop a branch-and-price-and-cut algorithm based on a constraint programming model for the subproblem. We also develop dominance rules and a fast infeasibility-detection algorithm based on a multidimensional knapsack problem to improve the efficiency of the constraint programming model. The computational results show that our method has an average optimality gap of 2.81% and significantly outperforms a compact mathematical formulation in the literature.

Keywords. Integrated operating room planning and scheduling, operations research in healthcare, branch-and-price-and-cut, constraint programming.

History. This article is published in *INFORMS Journal on Computing* in 2016. Also a preliminary version of this work is published in *CPAIOR* conference (2014).

4.1 Introduction

Hospital managers must provide high-quality services by making efficient use of available medical resources. The operating theatre, which includes operating and recovery rooms, is particularly important : 60%–70% of patients are admitted for some forms of surgical intervention, accounting for more than 40% of the total cost (Guerriero and Guido 2011). Careful scheduling is necessary to decrease costs and improve resource utilization and patient flow.

Operating-theatre management has generally been divided into three stages (Guerriero and Guido 2011) :

1. In the strategic stage, known as *case mix planning*, the operating-room availability is divided among the surgical departments or surgeons. The objective function may minimize the total deviation from a target allocation (Blake and Carter 2002) or maximize the weighted benefit of the scheduled surgeries (Baligh and Laughhunn 1969).
2. In the tactical stage, a *master surgery schedule* (a cyclic schedule over a medium-term planning horizon) is determined. It specifies the number of operating rooms available daily, the hours of availability, and the relative priorities of the surgeons or surgical departments.
3. In the operational stage, known as *elective case scheduling*, the daily scheduling is carried out. For each case we determine the surgeon, the starting time, the required equipment, etc. This stage is usually divided itself into the following two steps :
   (a) *Operating-room planning* : Each surgery is assigned to a room and a day in the planning horizon (usually a week) ; this is also called *advance scheduling.*
   (b) *Operating-room scheduling* : The surgeries are assigned to specific time intervals or the sequence of surgeries for each room is determined ; this is also called *allocation scheduling.*

Two strategies can be used for the tactical and operational stages :

1. *Block-Scheduling Strategy* : A set of time blocks is assigned to surgeons or surgical departments. Each time block is an interval on a specific day in a specific operating room.
2. *Open-Scheduling Strategy* : Surgeons from different departments can perform surgeries in the same time block. The surgeons must select the surgeries to perform the next day, and the operating theatre manager will schedule those surgeries in one or more rooms. This is a relaxation of the block-scheduling strategy, and it is expected to give a more efficient schedule. However, the management of such schedules is more difficult.

We focus on integrated operating room planning and scheduling (IORPS) at the operational level with an open-scheduling strategy. Many side constraints that are treated separately in the planning and scheduling phases must be considered simultaneously in the integrated problem. These constraints enforce the maximum daily hours of surgeons, prevent the overlapping of surgeries performed by the same surgeon, ensure sufficient cleaning time when switching from infectious to noninfectious cases, and enforce the surgery deadlines. The integrated approach leads to a more detailed planning problem, and this increases the chance of obtaining a stable schedule that can be successfully implemented. Our algorithm can also be applied in the context of the block-scheduling strategy when a block is shared among surgeons from the same department (Day et al. 2012).

IORPS has received little attention in the literature because of its intrinsic complexity ; the planning and scheduling problems are usually solved sequentially that can lead to local optimal solutions. For example, Jebali et al. (2006) used a mixed integer programming (MIP) model to plan the surgeries for a given day and applied a second MIP model to assign surgeries to rooms. Guinet and Chaabane (2003) developed an assignment formulation for the planning problem and used a heuristic to assign surgeries to rooms. They then adjusted the daily assignments to satisfy other staff and material constraints. Fei et al. (2006) proposed a column generation (CG) model for a block-scheduling strategy assuming that just one surgeon works in a specific operating room on a specific day. This assumption makes the scheduling part of the studied problem less complex since surgeon overlapping does not happen. Fei et al. (2010) developed a two-stage approach for an open-scheduling strategy. The first stage maximizes the room utilization via CG, and the second stage uses a genetic algorithm to minimize the under- and overtime costs.

Only a limited number of papers in the literature have proposed unified approaches to do operating room planning and scheduling simultaneously. The exact approaches to IORPS use compact mathematical formulations based on four- or five-index binary variables. Roland et al. (2006, 2010) and Marques et al. (2012) defined binary variables x_{ikdt} that are 1 if surgery i starts at time t in room k on day d. To the best of our knowledge, Roland et al. (2006) were the first to investigate IORPS ; their work was inspired by resource-constrained project scheduling. Roland et al. (2010) extended the model by considering various human resource constraints for the surgeons and nurses ; they computationally showed that the proposed formulation is just capable of solving small-sized instances. Both studies applied genetic algorithms to solve large-sized instances. Marques et al. (2012) used the same type of four-index variable to formulate the integrated operating room planning and scheduling. Because of the large size of the MIP model, assignment and scheduling of emergency cases is first performed, then other patients are scheduled and finally a heuristic algorithm improves the solution. Vijayakumar et al. (2013) defined the binary variables x_{ikdts} that are 1 if surgery i is performed by surgeon s at time t in room k on day d. The model was able to solve small instances, and the authors also developed a heuristic approach.

In addition to the binary variables, the most recent model has continuous variables to model the start time of surgeries, and it includes the well-known scheduling constraints that require big-M values. Consequently, the linear programming relaxation of this model is expected to be weaker than the models previously discussed. Therefore, Roland et al. (2006, 2010) and Marques et al. (2012) can be considered the state-of-the-art exact algorithms for IORPS. Although these formulations address the IORPS problem as it is, they are inefficient in solving real-sized instances because they involve a huge number of binary variables. Our problem is not exactly the same as any of the problems investigated previously ; the differences will be highlighted in Section 3.2 where assumptions are made.

We present a constraint-programming-based branch-and-price-and-cut algorithm for IORPS. A preliminary version, presented as a conference paper (Hashemi Doulabi et al. 2014) developed a column generation (CG) algorithm. CG is a well-known approach for linear programs that considers a large number of variables without including all of them explicitly in the model (Dantzig and Wolfe 1960, Gilmore and Gomory 1961, Desaulniers et al. 2006). The basic columns (variables) to define the optimal solution of such linear programs are generated iteratively by solving a subproblem. In the CG algorithm of Hashemi Doulabi et al. (2014) each subproblem generates schedules for a single room on a single day.

This paper extends the algorithm of Hashemi Doulabi et al. (2014) in two ways. First, we improve the efficiency of the constraint programming model in the subproblem. We add dominance rules to decrease the size of the subproblems, and we also present an infeasibility-detection algorithm based on a relaxation of the original subproblem that discovers infeasible subproblems very quickly in many cases. Second, we extend the CG algorithm to a branch-and-price-and-cut approach by developing branching and cutting plane procedures. Our results show that the new approach significantly outperforms the original CG algorithm, decreasing the optimality gap from 9.90% to 2.81%. We also compare our approach with a state-of-the-art integer programming model for IORPS adapted from Marques et al. (2012) and also a pure constraint-programming model.

The remainder of this paper is organized as follows. Section 4.2 presents the problem definition and some notation, and Section 4.3 introduces the CG formulation. Section 4.4 discusses the dominance rules and a fast infeasibility-detection algorithm. Section 4.5 presents the general structure of the branch-and-price-and-cut algorithm and the branching and cutting-plane procedures. Section 4.6 presents extensive computational results, and Section 4.7 provides concluding remarks. We have provided supplements of this paper in Appendix A.

4.2 Problem Statement and Notation

In this section, we present the IORPS assumptions and some notation. An IORPS problem is defined over a planning horizon, usually a week. There are two types of surgeries : (1) mandatory surgeries whose deadlines fall within the current planning horizon and (2) optional surgeries that can be postponed to a later period. We consider an open scheduling environment, i.e., each room can be shared by different surgeons. The assumptions are as follows :

1. The durations of the surgeries are deterministic.
2. Each surgeon has determined his or her maximum surgery time for each day of the planning horizon.
3. Different types of infections can be present. An obligatory cleaning of the operating room is required when switching to a case with a different type of infection or to a noninfectious case. No cleaning is required between two cases with no infection or the same type of infection.
4. The surgeries have already been assigned to surgeons.
5. The recovery rooms are not a bottleneck : sufficient beds are available.
6. Eight hours are available in each operating room. No overtime is available.
7. The operating rooms are identical in terms of available time and equipment.

We maximize the total scheduled surgery time over the planning horizon subject to :

1. All mandatory surgeries are scheduled.
2. Surgeons do not exceed their maximum daily surgery times.
3. Each operating room and each surgeon can handle just one patient at a time.
4. Cleaning time is scheduled as previously explained.

One may criticize that Assumption 1 is not realistic as in the real world the durations of surgeries are stochastic. However it should be considered that the focus of this study is to have a planning problem with more details at the scheduling level to increase the chance of obtaining a stable schedule with more chance of successful implementation. Furthermore the problem is already very complicated so we leave the stochastic version to future research. Assumption 2 is realistic, but only Marques et al. (2012) have taken it into account. Assumption 3 is also very realistic and was first studied by Cardoen et al. (2009b), but it has not been considered in the IORPS context. Assumption 4 is appropriate for hospitals where surgeons operate on the patients they have previously seen in their clinics. Vijayakumar et al. (2013) consider the assignment of surgeons to surgeries ; this approach is suitable for teaching hospitals, where surgeries are assigned to assistant surgeons and residents as part of their training programs. Assumption 5 is common in the IORPS literature because the problem is already difficult without these additional resource constraints. Some researchers (Vijayakumar et al. 2013, Marques et al. 2012) apply Assumption 6 whereas others (Roland et al. 2006, 2010) allow overtime. Regarding Assumption 7, only Roland et al. (2010) consider rooms with different equipment. Our algorithm solves the problem without this assumption ; see Appendix A.1 after reading Section 4.3. Our problem has features in common with the problems in the literature, but it is not exactly the same as any of them.

We now introduce the basic notation used throughout the paper. Some other notations may also be presented separately in each section.

Sets:

$\mathcal{D}$:	Set of days in planning horizon.
$\mathcal{I}$:	Set of surgeries ; $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$.
$\mathcal{I}_1$:	Set of mandatory surgeries.
$\mathcal{I}_2$:	Set of optional surgeries.
$\mathcal{I}'_l$:	Set of surgeries for surgeon l.
$\mathcal{K}_d$:	Set of operating rooms for day d.
$\mathcal{L}$:	Set of surgeons.
$\mathcal{T}_d$:	Set of time slots for day d ; the available time is discretized into this set of time slots.

Parameters:

t_i :	Duration of surgery i ($i \in \mathcal{I}$).
s_i :	Surgeon for surgery i.
dd_i :	Deadline of surgery i.
A_{ld} :	Maximum hours for surgeon l on day d.
OCT :	Duration of an obligatory cleaning that was explained in Assumption 3.
$CL_{ii'}$:	Equal to OCT if cleaning is required between surgeries i and i' ; 0 otherwise.

Since the integer programming model developed in (Marques et al. 2012) is one of state-of-the-art exact algorithms for IORPS, an integer programming model adapted from (Marques et al. 2012) is presented in Appendix A.2. Moreover, a pure constraint-programming model formulating the previous problem is presented in Appendix A.3. We will compare our algorithm with these two approaches.

4.3 Formulation and Column Generation Algorithm

In this section we present an integer programming model and explain how we apply CG to this problem. The following two sections discuss the master problem and the subproblem.

4.3.1 Master Problem

For the master problem the sets, parameters, and variables are as follows.

Variable:

x_j : Equal to 1 if schedule j is accepted ; 0 otherwise. Each schedule is a set of surgeries and cleaning activities with fixed start times for a single operating room on a given day ; it must satisfy constraints 2–4 stated in Section 4.2.

Sets:

$\mathcal{J}$: Set of all feasible schedules for all operating rooms during the planning period.

$\mathcal{J}_i$: Set of feasible schedules that include surgery i.

$\mathcal{J}'_d$: Set of feasible schedules for day d.

Parameters:

B_j : Total duration of all the surgeries in schedule j.

a_{ijt} : 1 if surgery i starts at time t in schedule j ; 0 otherwise.

a'_{ij} : 1 if surgery i is included in schedule j ; 0 otherwise.

b_{jd} : 1 if schedule j is scheduled on day d ; 0 otherwise.

The master problem is as follows :

$$\max \sum_{j \in \mathcal{J}} B_j x_j \tag{4.1}$$

subject to :

$$\sum_{j \in \mathcal{J}_i} x_j = 1, \qquad i \in \mathcal{I}_1, \tag{4.2}$$

$$\sum_{j \in \mathcal{J}_i} x_j \leq 1, \qquad i \in \mathcal{I}_2, \tag{4.3}$$

$$\sum_{j \in \mathcal{J}'_d} x_j \leq |\mathcal{K}_d|, \qquad d \in \mathcal{D}, \tag{4.4}$$

$$\sum_{j \in \mathcal{J}'_d} \sum_{i \in \mathcal{I}'_l} t_i a'_{ij} x_j \leq A_{ld}, \qquad l \in \mathcal{L},\ d \in \mathcal{D}, \tag{4.5}$$

$$\sum_{j \in \mathcal{J}'_d} \sum_{i \in \mathcal{I}'_l} \sum_{t' \in \mathcal{T}_d :\, t' \le t < t' + t_i} a_{ijt'} x_j \le 1, \qquad d \in \mathcal{D},\, l \in \mathcal{L},\, t \in \mathcal{T}_d, \tag{4.6}$$

$$x_j \in \{0, 1\}, \qquad j \in \mathcal{J}. \tag{4.7}$$

Constraints (4.2)–(4.3) ensure that the mandatory surgeries are performed and allow the optional surgeries to be postponed. The deadlines of the mandatory surgeries will be respected in the subproblem. Constraint (4.4) ensures for each day that the number of schedules does not exceed the number of available operating rooms. Constraint (4.5) enforces the maximum working hours for each surgeon. Constraint (4.6) prevents overlapping surgeries for the same surgeon ; in the literature it is referred to as the coloring constraint.

The model defined by (4.1)–(4.7) is inspired by the model presented by Fei et al. (2009). We have added the coloring constraint, and the objective function is also different. Moreover, our binary variables determine the daily schedule of an operating room, whereas the in Fei et al. (2009) the binary variables determine the assignments of surgeries to rooms but not the detailed schedules. The Fei model is intended just for planning, and it may assign more than one surgery to a surgeon at a given time. Therefore it is possible that the generated plans do not result in feasible schedules. The structure of our subproblem is totally different from that of the Fei subproblem.

Moreover, the other advantage of the proposed formulation to Fei model is that we have broken the symmetry of the identical operating rooms (see Section 4.3.2), and our model has only $|\mathcal{D}|$ subproblems. In contrast, the Fei model must solve $\sum_{d \in \mathcal{D}} |\mathcal{K}_d|$ subproblems at each iteration of the CG, where $|\mathcal{D}|$ is the number of days and $|\mathcal{K}_d|$ is the number of operating rooms on day d. We could define other variables for the master problem ; e.g., the variables could correspond to daily schedules for surgeons. The main advantage of the variables as defined is that the symmetry of the operating rooms can be broken in this case.

We embed model (4.1)–(4.7) in a branch-and-price algorithm. At each node k of the branch and bound tree, we generate a restricted number of columns ($\mathcal{J}_k \subseteq \mathcal{J}$) ; this problem is called the restricted master problem denoted RMP_k. Its linear programming relaxation, denoted $LR\text{-}RMP_k$, is solved by a CG algorithm, thereby generating more columns and obtaining a valid upper bound from the active nodes at the bottom of the tree. At the end of the branch-and-price algorithm we solve model (4.1)–(4.7) with all the generated columns to find a feasible solution and the corresponding lower bound.

4.3.2 Subproblem

After solving *LR-RMP*$_k$ we must solve a number of subproblems to generate new columns. These subproblems are defined using the dual values obtained by solving the relaxed master problem. If no column can be generated we terminate the CG process. We use a constraint programming model to formulate the subproblem. For each day $d \in \mathcal{D}$ we solve a subproblem to generate a schedule for an operating room on that day. Let $\pi_i^{(2)}$, $\pi_i^{(3)}$, $\pi_d^{(4)}$, $\pi_{dl}^{(5)}$, and $\pi_{dlt}^{(6)}$ be the dual variables corresponding to constraints (4.2)–(4.6) in the LR-RMP. We also define $\pi_i^{(2,3)}$ as

$$\pi_i^{(2,3)} = \begin{cases} \pi_i^{(2)}, & i \in \mathcal{I}_1; \\ \pi_i^{(3)}, & i \in \mathcal{I}_2. \end{cases}$$

The reduced cost of variable x_j is

$$\sigma_j = B_j - \sum_{d\in\mathcal{D}} \sum_{i\in\mathcal{I}} b_{jd} a'_{ij} \pi_i^{(2,3)} - \sum_{d\in\mathcal{D}} b_{jd} \pi_d^{(4)} - \sum_{d\in\mathcal{D}} \sum_{l\in\mathcal{L}} \sum_{i\in\mathcal{I}'_l} t_i b_{jd} a'_{ij} \pi_{dl}^{(5)}$$
$$- \sum_{d\in\mathcal{D}} \sum_{l\in\mathcal{L}} \sum_{i\in\mathcal{I}'_l} \sum_{t\in\mathcal{T}} \sum_{t'\in\mathcal{T}_d :\, t' \le t < t'+t_i} a_{ijt'} b_{jd} \pi_{dlt}^{(6)}.$$

In the subproblem we seek the column with the largest positive reduced cost. Because B_j is constant and all the other terms are summed over $d \in \mathcal{D}$, decomposition of the subproblem over index $d \in \mathcal{D}$ can be presented as follows :

$$\max\left\{ B_j - \sum_{i\in\mathcal{I}} a'_{ij} \pi_i^{(2,3)} - \pi_d^{(4)} - \sum_{l\in\mathcal{L}} \sum_{i\in\mathcal{I}'_l} t_i a'_{ij} \pi_{dl}^{(5)} - \sum_{l\in\mathcal{L}} \sum_{i\in\mathcal{I}'_l} \sum_{t\in\mathcal{T}} \sum_{t'\in\mathcal{T}_d :\, t' \le t < t'+t_i} a_{ijt'} \pi_{dlt}^{(6)} \right\}.$$

Because only a small number of surgeries (n) can be performed in each operating room on a given day, we use a position-based model. The variables for the constraint programming model of the subproblem are as follows.

W_p : Index of surgery to be performed in position p on day d.

V_p : Start time of surgery assigned to position p on day d.

C_i : Number of W_p ($p \in \{1,\ldots,n\}$) variables that take the value i ($i \in \mathcal{I} \cup \{0\}$).

The subproblem constraints are :

$$W_p \in \{i \in \mathcal{I} \mid dd_i \ge d\} \cup \{0\}, \qquad p \in \{1,\ldots,n\}, \tag{4.8}$$
$$V_p \in \{0, 1, \ldots, |\mathcal{T}_d|\}, \qquad p \in \{1,\ldots,n\}, \tag{4.9}$$
$$\text{If } (W_p = 0) \text{ then } (W_{p+1} = 0), \qquad p \in \{2,\ldots,n-1\}, \tag{4.10}$$

$$V_{p+1} \geq V_p + t_{[W_p]} + CL_{[W_p][W_{p+1}]}, \qquad p \in \{1, \ldots, n-1\}, \qquad (4.11)$$

$$\sum_{p=1}^{n} (W_p == i) = C_i, \qquad i \in \mathcal{I}, \qquad (4.12)$$

$$0 \leq C_i \leq 1, \qquad i \in \mathcal{I}, \qquad (4.13)$$

$$\sum_{p=1}^{n} (s_{[W_p]} == l) t_{[W_p]} \leq A_{ld}, \qquad l \in \mathcal{L}, \qquad (4.14)$$

$$\text{If } W_p = 0 \text{ then } V_p = V_{p-1} + t_{[W_{p-1}]}, \qquad p \in \{2, \ldots, n\}. \qquad (4.15)$$

Constraint (4.8) ensures that the positions are either assigned to a surgery in set $\mathcal{I}$ or left empty with value 0. The value n is an upper bound on the number of surgeries assigned to an operating room. Constraint (4.9) defines the domain of the start-time variables. We have supposed the surgery durations are multiples of a 5 minute time unit and discritized the available time to 5-minute time slots. The values in domains are the indices of the time slots. Constraint (4.10) states that if a position is left empty then the next available position must also be empty, i.e., we do not allow empty positions between two surgeries. We observe that holes in the schedule are possible since the start times of the surgeries are determined by the V_p variables. The role of constraint (4.10) is to break the symmetry of the model by eliminating identical solutions with different assignments of the surgeries to the available positions. Constraint (4.11) ensures that the start time of a surgery is greater than the finish time of the previous surgery plus any cleaning time. In constraint (4.12), $(W_p == i)$ is a Boolean expression equal to 1 when W_p is equal to i and 0 otherwise. Constraint (4.12) together with constraint (4.13) ensures that each surgery appears at most once in a schedule. Constraint (4.14) enforces the maximum working hours of the surgeons. This restriction was present in the master problem (constraint (4.5)) ; we include it in the subproblem to prevent infeasible columns. Constraint (4.15) forces the start time of an empty position to be equal to the finish time of the previous position. This avoids similar columns with different start times for the empty positions.

We assume that $t_{[0]} = 0$ and $s_{[0]} = 0$. This ensures that when W_p is 0, $t_{[W_p]}$ and $s_{[W_p]}$ will be 0. This together with constraints (4.10) and (4.15) guarantees that when W_p is 0 the start time of this position and the next positions will be fixed to the start time of the last position with a surgery from $\mathcal{I}$. We also assume that $CL_{0i} = 0$, $CL_{i0} = 0$, i.e., there is no cleaning before and after empty positions.

The objective function is

$$\max\Bigg\{\sum_{p=1}^{n} t_{[W_p]} - \sum_{p=1}^{n} \pi_{[W_p]}^{(2,\,3)} - \sum_{p=1}^{n} (t_{[W_p]} \pi_{d[s_{[W_p]}]}^{(5)}) \\ - \sum_{p=1}^{n} \sum_{t \in \mathcal{T}_d} ((t \geq V_p) \wedge (t < V_p + t_{[W_p]})) \pi_{d[s_{[W_p]}]t}^{(6)} - \pi_d^{(4)} \Bigg\}. \tag{4.16}$$

Here, $\pi_{[W_p]}^{(2,\,3)}$, $\pi_{d[s_{[W_p]}]}^{(5)}$, and $\pi_{d[s_{[W_p]}]t}^{(6)}$ are element constraints that are 0 if W_p is 0. Otherwise, the corresponding dual values will be computed. The objective function can be simplified using precomputed matrices as follows :

$$\max\Bigg\{\sum_{p=1}^{n} G_{d[W_p]} - \sum_{p=1}^{n} \pi_{d[W_p][V_p]}^{*} - \pi_d^{(4)} \Bigg\}, \tag{4.17}$$

where

$$G_{di} = \begin{cases} t_i - \pi_i^{(2,3)} - t_i \pi_{ds_i}^{(5)}, & i \neq 0; \\ 0, & i = 0; \end{cases} \quad \text{and}$$

$$\pi_{dit}^{*} = \begin{cases} \sum\limits_{t' \in \mathcal{T}_d :\, t \leq t' < t + t_i} \pi_{ds_i t'}^{(6)}, & i \neq 0; \\ 0, & i = 0. \end{cases}$$

We observe that (4.17) is more efficient than (4.16) since the fourth term in (4.16) has $n \times |\mathcal{T}_d|$ element constraints whereas (4.17) has only n. As a result, it is much faster to create and delete (4.17).

In summary, the constraint programming model for the CG subproblem is

$\forall\, d \in \mathcal{D}$:

$$\max\Bigg\{\sum_{p=1}^{n} G_{d[W_p]} - \sum_{p=1}^{n} \pi_{d[W_p][V_p]}^{*} - \pi_d^{(4)} \Bigg\} \tag{4.18}$$

subject to :

constraints (4.8)–(4.15)

$$\sum_{p=1}^{n} G_{d[W_p]} - \sum_{p=1}^{n} \pi_{d[W_p][V_p]}^{*} - \pi_d^{(4)} \geq 0. \tag{4.19}$$

Constraint (4.19) ensures that only columns with nonnegative reduced costs will be generated. During the constraint programming search, solutions with positive reduced costs will become new columns in the master problem. After the CG convergence, the objective value obtained by solving the linear programming relaxation of the master problem (LR-RMP) is an upper bound for the problem. We obtain a lower bound by solving an integer programming model for the restricted master problem.

4.4 Column Generation Enhancements

In this section we enhance the CG with four dominance rules and a fast infeasibility-detection algorithm. In constraint programming, different methods such as symmetry breaking and dominance rules were developed to reduce the search effort. Focacci and Milano (2001), and Fahle et al. (2001) independently developed symmetry breaking via dominance detection (SBDD). This algorithm breaks symmetry in the search tree by pruning a node if it is symmetrically equivalent to another node that is already explored. The idea of dominance rules is to specify some characteristics such that solutions with these characteristics dominate other solutions with respect to the objective function. Using dominance rules many low-quality solutions can be pruned off quickly resulting in significant speedups. Chu and Stuckey (2012) developed a generic method for generating dominance rules that are proven to be correct and compatible with each other. Many researchers have also applied problem-specific dominance rules in different combinatorial problems such as template design problem (Proll and Smith 1998), steel mill design problem (Prestwich and Beck 2004), rectangle packing problem (Korf 2004), open stacks problem (Chu and Stuckey 2009) and talent scheduling problem (Garcia de la Banda et al. 2011). Following in their footsteps we design problem-specific dominance rules applicable to the subproblem of our column generation algorithm that is a single machine scheduling problem with irregular starting time costs.

The first dominance rule characterizes the start time of the surgery in the first position. The second removes surgeries that will reduce the objective value of the subproblem if added to the schedule. For the third rule, we observe that if because of certain constraints a pair of surgeries cannot exist simultaneously in the same schedule, and one dominates the other in terms of the change in the objective value, then the latter does not appear in the optimal solution. For this dominance rule, if the pair of surgeries can appear simultaneously in the schedule then instead of removing the dominated surgery, we condition its existence in the solution on the scheduling of the dominant surgery. The fourth dominance rule states that if because of the constraints in the subproblem a set of surgeries that dominate another surgery cannot appear in an operating room schedule simultaneously, then the latter surgery does not appear in the optimal solution.

The fast infeasibility-detection algorithm solves a relaxation of the subproblem using a fast algorithm instead of the constraint programming solver. The latter can be time-consuming for subproblems that are infeasible with respect to constraint (4.19). If the objective value obtained by solving the relaxation of the subproblem is negative then the original subproblem is infeasible. Otherwise, the positive objective value found by the fast infeasibility-detection algorithm is an upper bound on the objective value of the constraint programming model.

4.4.1 Dominance Rules

The first dominance rule characterizes the start time of the surgery in the first position.

Dominance Rule 1. *The following constraint is valid for subproblem d :*

$$(V_1 == 0) \quad or \quad (V_1 \geq SD_d),$$

where SD_d is the shortest surgery duration for all the surgeries with deadlines on or after day d (i.e., $SD_d = \min_{i \in \mathcal{I};\, dd_i \geq d}\{t_i\}$).

Proof. We prove that there always exists an optimal solution for the IORPS problem that satisfies the above condition in all the operating rooms. Assume that in an optimal solution, in one of the operating rooms on day d a surgery assigned to the first position starts at time V^* in the interval $[1, \mathrm{SD}_d - 1]$. The corresponding surgeon cannot have been in another operating room on this day in the interval $[0, \mathrm{SD}_d]$, because an overlap would then exist, which is a contradiction. Thus, it is possible to shift the start time of the surgery in the first position from V^* to 0. $\square$

The other rules can be implemented dynamically. Note that for the variable selection we select the W_p variables in lexicographic order and then fix the V_p variables. The following dominance rules are applied when there is at least one W_p variable that is not yet fixed. For the remainder of this paper it is supposed that p^* the smallest index such that W_{p^*} is not yet fixed.

We define $\Delta_{id}^{\text{best}}$ to be the best improvement in the objective value of subproblem $d \in \mathcal{D}$ when we add surgery i :

$$\Delta_{id}^{\text{best}} = t_i - \pi_i^{(2,\,3)} - t_i \pi_{d[s_i]}^{(5)} - \min_{t' \in \mathrm{Dom}(V_p^*)} \left\{ \sum_{t=t'}^{t'+t_i-1} \pi_{d[s_i]t}^{(6)} \right\}, \tag{4.20}$$

where $\mathrm{Dom}(V_{p^*})$ denotes the domain of variable V_{p^*}. Here t_i is the direct improvement from the increase in B_j when surgery i is added. The terms $-\pi_i^{(2,3)}$ and $-t_i\pi_{d[s_i]}^{(5)}$

are the changes in the objective function of the subproblem due to the dual values of constraints (4.2), (4.3), and (4.5). Because the dual values of constraint (4.6) are nonnegative, $-\min_{t' \in \text{Dom}(V^*_p)}\{\sum_{t=t'}^{t'+t_i-1} \pi^{(6)}_{[s_i]\,dt}\}$ is the best possible change in the objective function of the subproblem regarding the dual values of constraint (4.6). This term considers the best possible start time for surgery i. Note that the interval $[1, \text{SD}_d - 1]$ has been removed from $\text{Dom}(V_{p^*})$ by Dominance Rule 1. Equation (4.20) considers the possibility of performing surgery i in all available positions $p \geq p^*$ because in (4.20) we can replace $\text{Dom}(V_{p^*})$ with $\bigcup_{p \geq p^*} \text{Dom}(V_p)$ since $\text{Dom}(V_{p+1}) \subseteq \text{Dom}(V_p)$ is valid with respect to constraint (4.12).

In (4.16) and (4.18), $\pi^{(4)}_d$ is the nonnegative dual variable of constraint (4.4), which is fixed for each subproblem $d \in \mathcal{D}$; adding surgery i does not change this part of the objective function. We similarly define $\Delta^{\text{worst}}_{id}$ to be the largest deterioration in the objective function of subproblem $d \in \mathcal{D}$ when we add surgery i :

$$\Delta^{\text{worst}}_{id} = t_i - \pi^{(2,\,3)}_i - t_i \pi^{(5)}_{d[s_i]} - \max_{t' \in \text{Dom}(V^*_p)} \left\{ \sum_{t=t'}^{t'+t_i-1} \pi^{(6)}_{d[s_i]t} \right\}. \tag{4.21}$$

The next dominance rule eliminates surgeries that deteriorate the objective value of the subproblem.

Dominance Rule 2. *If $\Delta^{best}_{id} < 0$ then surgery i does not appear in the optimal solution of subproblem d.*

Proof. Assume that surgery i with $\Delta^{\text{best}}_{id} < 0$ appears in an optimal solution. When we remove surgery i, the objective function is improved by at least $|\Delta^{\text{best}}_{id}|$, which is a contradiction. □

Dominance Rules 3(a)–3(c) remove or condition a surgery by finding a dominant surgery. We introduce the following notation.

RAS_l :	The maximum remaining time for surgeon l given the surgeries scheduled in positions 1 to $p^* - 1$. This is $A_{ld} - \sum_{p=1}^{p^*-1} (s_{[W_p]} == l) t_{[W_p]}$ for $p^* > 1$ and A_{ld} for $p^* = 1$.
RAO :	The remaining time in the operating room given the surgeries scheduled in positions 1 to $p^* - 1$. This is $|\mathcal{T}_d| - \min\{\text{Dom}(V_{p^*})\}$ where $\min\{\text{Dom}(V_{p^*})\}$ is the minimum value in the domain of V_{p^*}.

Dominance Rule 3(a). *In subproblem d, for a given surgery i with $dd_i \geq d$, if there is a surgery i' with $dd_{i'} \geq d$ and $s_i = s_{i'}$ such that*

(1) $\Delta^{worst}_{i'd} > \Delta^{best}_{id}$;

(2) $t_i \geq t_{i'} + \lambda$ *; and*

(3) $t_i + t_{i'} > RAS_{s_i}$ *;*

then surgery i is dominated by surgery i' and can be removed from the domain of W_p variables without losing the optimal solution. Here, λ is a constant upper bound on the additional pre-/post-cleaning time needed when surgery i is replaced by surgery i'.

PROOF. The right-hand side of condition (3) is the maximum remaining time for surgeon $s_i = s_{i'}$. Given this condition, surgeries i and i' cannot both appear in a schedule. Thus, assume that surgery i is included in the schedule. Then, by condition (2), removing surgery i gives sufficient time to perform surgery i'. Condition (1) ensures that replacing surgery i by surgery i' improves the objective value. Therefore, surgery i does not appear in the optimal solution of subproblem d. $\square$

Table 4.1 – Values of λ for Different Infectious and Noninfectious Combinations for Surgeries i, i', and the Surgery in Position $p^* - 1$

Case	Surgery i	Surgery i'	Surgery in position $p^* - 1$	λ
1	Noninfectious	Noninfectious	No condition	0
2	Noninfectious	Infectious	Noninfectious or infectious with $f'_{p^*-1} \neq f_{i'}$	OCT
3	Noninfectious	Infectious	Infectious with $f'_{p^*-1} = f_{i'}$	0
4	Infectious	Noninfectious	Noninfectious or infectious with $f'_{p^*-1} = f_i$	0
5	Infectious	Noninfectious	Infectious with $f'_{p^*-1} \neq f_i$	OCT
6	Infectious	Infectious with $f_{i'} = f_i$	No condition	0
7	Infectious	Infectious with $f_{i'} \neq f_i$	Noninfectious or infectious with $f'_{p^*-1} \neq f_i$ and $f'_{p^*-1} \neq f_{i'}$	OCT
8	Infectious	Infectious with $f_{i'} \neq f_i$	Infectious with $f'_{p^*-1} = f_i$	2OCT
9	Infectious	Infectious with $f_{i'} \neq f_i$	Infectious with $f'_{p^*-1} = f_{i'}$	0

Table 4.1 presents the values of λ for different combinations of infectious and noninfectious situations : $f_i, f_{i'}$, and f'_{p^*-1} denote the infection status of surgeries i, i', and the surgery in position $p^* - 1$, respectively. Appendix A.4 derives the values of λ for the nine cases.

DOMINANCE RULE 3(b). *If condition (3) of Dominance Rule 3(a) does not hold, then the following constraint is valid for the subproblem :*

$$C_i \leq C_{i'}. \tag{4.22}$$

This constraint states that if surgery i appears in the schedule then surgery i' must also appear ; otherwise the objective value can be improved by replacing surgery i by surgery i'.

DOMINANCE RULE 3(c). *In Dominance Rule 3(a) if $s_i \neq s_{i'}$ but conditions (1) and (2) hold and we have*

$$RAS_{s_{i'}} \geq RAO, \tag{4.23}$$

then constraint (4.22) can be added to the subproblem without losing the optimal solution because the objective value can be improved by replacing surgery i by surgery i′. Condition (4.23) guarantees that the constraint on the maximum remaining time for surgeon $s_{i'}$ is not violated by the replacement.

The next dominance rule considers a set of surgeries that conditions another surgery through constraint (4.22). If it is not possible to schedule all the entries in the set, then the latter surgery does not appear in the optimal solution.

DOMINANCE RULE 4. *Let $\mathcal{P}_i$ be a set of surgeries that, with respect to constraint (4.22), must be scheduled if surgery i is scheduled. Then surgery i can be removed from the domain of W_p if $\sum_{r \in \mathcal{P}_i \cup \{i\}} t_r > |\mathcal{T}_d|$ or $\sum_{r \in (\mathcal{P}_i \cap \mathcal{I}'_{s_i}) \cup \{i\}} t_r > A_{s_i d}$.*

PROOF. If we include surgery i in the schedule, we must also include all the surgeries in $\mathcal{P}_i$. However, $\sum_{r \in \mathcal{P}_i \cup \{i\}} t_r > |\mathcal{T}_d|$ indicates that the sum of the durations of surgery i and the preceding surgeries exceeds the available time in the operating room in subproblem d. Moreover, $\sum_{r \in (\mathcal{P}_i \cap \mathcal{I}'_{s_i}) \cup \{i\}} t_r > A_{s_i d}$ indicates that the sum of the duration of surgery i and the durations of surgeon s_i's previous surgeries exceeds the available time for s_i on day d. Therefore, surgery i can be removed from the domain of W_{p^*}. □

4.4.2 Fast Infeasibility-Detection Algorithm

Sometimes the dual value of $\pi_d^{(4)}$ is so large that the best possible schedule in the subproblem has a negative objective value. No column with a positive reduced cost can be generated, and the subproblem is infeasible with respect to constraint (4.19). The rapid detection of such infeasibilities is very critical, especially in the last CG iterations. We use a knapsack relaxation to find an upper bound on the optimal value of subproblem d. We consider a multidimensional knapsack problem in which each surgery i from the subproblem corresponds to an item with the value $\Delta^{\text{best}}_{id}$. The integer programming formulation of this multidimensional knapsack problem reads as follows :

Sets :

$\mathcal{I}''_d$: Set of surgeries with deadlines on or after day d that remain in the domain of W_p after that dominance rules have been applied.

Variables :

y_i : Equals to 1 if item i (corresponding to surgery i) is included in the knapsack.

$$\max \sum_{i \in \mathcal{I}''_d} \Delta^{\text{best}}_{id} y_i \tag{4.24}$$

subject to :

$$\sum_{i \in \mathcal{I}''_d} t_i y_i \leq |\mathcal{T}_d|, \tag{4.25}$$

$$\sum_{i \in \mathcal{I}''_d \cap \mathcal{I}'_l} t_i y_i \leq A_{ld}, \qquad l \in \mathcal{L}, \tag{4.26}$$

$$y_i \in \{0, 1\}, \qquad i \in \mathcal{I}''_d. \tag{4.27}$$

Constraint (4.25) ensures that the sum of the durations of scheduled surgeries does not exceed the total available time in the operating room. Constraint (4.26) states that the total scheduled time for each surgeon does not exceed the maximum time for that surgeon. Obviously this multidimensional knapsack problem is a relaxation of the scheduling problem in the subproblem and an upper bound for this problem is an upper bound for the subproblem if we neglect the fixed value $\pi^{(4)}_d$. We use a well-known greedy algorithm with a value/weight selection criterion to find an upper bound for the problem defined by (4.24)–(4.27). As before, p^* is the smallest index such that W_{p^*} is not yet fixed. This fast infeasibility-detection algorithm is implemented dynamically, i.e., it is called once at the beginning when no variable W_p is fixed ($p^* = 1$) and also whenever a variable W_p is fixed ($p^* > 1$). Before presenting the algorithm we define $\Delta^{\text{best}}_{idp}$ as follows :

$$\Delta^{\text{best}}_{idp} = t_i - \pi^{(2,\,3)}_i - t_i \pi^{(5)}_{d[s_i]} - \min_{t' \in \text{Dom}(V_p)} \left\{ \sum_{t=t'}^{t'+t_i-1} \pi^{(6)}_{d[s_i]t} \right\}. \tag{4.28}$$

This value is the best improvement in the objective value of subproblem d obtained by setting the start time of surgery i that is already scheduled at position p. This start time must take a value from Dom(V_p). The difference between $\Delta^{\text{best}}_{idp}$ and $\Delta^{\text{best}}_{id}$ is that V_{p^*} is replaced by V_p.

The fast infeasibility-detection algorithm is as follows :

1. Set $\mathit{UpperBound}_d = \sum_{p=1}^{p^*-1} \Delta^{\text{best}}_{[W_p]dp}$. For $p^* = 1$ this value is defined as zero.
2. Sort the unscheduled surgeries by $(\Delta^{\text{best}}_{id}/t_i)$ and set $i^* = \text{argmax}_{i\in\mathcal{I};\, dd_i\geq d}\{\Delta^{\text{best}}_{id}/t_i\}$.
3. Assign surgery i^* to the operating room if the remaining available time in the operating room is sufficient and the constraint on the maximum time of surgeon s_{i^*} is not violated. In this case, set $\mathit{UpperBound}_d := \mathit{UpperBound}_d + \Delta^{\text{best}}_{i^*d}$, $\mathit{RAS}_{s_{i^*}} := \mathit{RAS}_{s_{i^*}} - t_{i^*}$, and $\mathit{RAO} := \mathit{RAO} - t_{i^*}$ and return to step 1. Otherwise go to step 4 or 5.
4. If $t_{i^*} > \mathit{RAS}_{s_{i^*}}$ and $\mathit{RAS}_{s_{i^*}} < \mathit{RAO}$ then divide i^* into two surgeries i^*_1 and i^*_2 with durations $\mathit{RAS}_{s_{i^*}}$ and $t_{i^*} - \mathit{RAS}_{s_{i^*}}$ where s_{i^*} denotes the surgeon of surgery i^*. Assign surgery i^*_1 to the schedule and disregard surgery i^*_2. Set $\mathit{UpperBound}_d := \mathit{UpperBound}_d + \mathit{RAS}_{s_{i^*}}(\Delta^{\text{best}}_{i^*d}/t_{i^*})$, $\mathit{RAS}_{s_{i^*}} := 0$, and $\mathit{RAO} := \mathit{RAO} - (t_{i^*} - \mathit{RAS}_{s_{i^*}})$ and return to step 2. Here $\mathit{RAO} > \mathit{RAS}_{s_{i^*}}$ guarantees that there is sufficient time in the operating room for the assignment of surgery i^*_1. Surgery i^*_2 is eliminated for the rest of the algorithm as there is not any available time for surgeon s_{i^*} to perform it.
5. If $t_{i^*} > \mathit{RAO}$ and $\mathit{RAO} \leq \mathit{RAS}_{s_{i^*}}$ then divide surgery i^* into two surgeries i^*_1 and i^*_2 with durations RAO and $t_{i^*} - \mathit{RAO}$. Assign surgery i^*_1 to the schedule and set $\mathit{UpperBound}_d := \mathit{UpperBound}_d + \mathit{RAO}(\Delta^{\text{best}}_{i^*d}/t_{i^*})$. Then stop because no operating-room time remains after the assignment of surgery i^*_1. Here condition $\mathit{RAO} \leq \mathit{RAS}_{s_{i^*}}$ ensures that surgeon s_{i^*} has sufficient time to perform surgery i^*_1.

This fast infeasibility-detection algorithm is evaluated dynamically as the W_p variables are fixed and p^* increases. When $p^* = 1$, if $\mathit{UpperBound}_d - \pi^{(4)}_d < 0$ at the end of the algorithm then the subproblem on day d is infeasible and can be skipped. Otherwise, the following constraint can be added to the subproblem :

$$\begin{aligned} &\mathit{Objective\ Function\ of\ subproblem\ d} \\ &\leq \mathit{UpperBound}_d - \pi^{(4)}_d. \end{aligned} \tag{4.29}$$

Constraint (4.29) ensures that the constraint programming solver stops when it finds a feasible solution with an objective value of $\mathit{UpperBound}_d - \pi^{(4)}_d$.

When $p^* > 1$, if $\mathit{UpperBound}_d - \pi^{(4)}_d < 0$, then the current partial solution will not lead to a complete solution with a positive objective value, and the constraint programming solver backtracks to another partial solution in the search tree. However, if $\mathit{UpperBound}_d - \pi^{(4)}_d \geq 0$, constraint (4.29) can be added locally to the search tree. In this case, constraint (4.29) reflects that the objective value of the current partial solution cannot be better than $\mathit{UpperBound}_d -$

$\pi_d^{(4)}$ and the constraint programming solver then backtracks whenever it finds a local complete solution with an objective value of $\mathit{UpperBound}_d - \pi_d^{(4)}$.

It should be noted that the effect of cleaning times are not considered in the previous upper-bound calculation since this simplification results in a relaxed knapsack problem and is necessary to get a valid upper bound. Note that for the value selection of W_p variables the surgeries are sorted by descending order of $\Delta_{id}^{\text{best}}/t_i$ and for the V_p variables a lexicographic ordering is applied.

4.5 Branch-and-Price-and-Cut Algorithm

It is well-known that to improve the quality of solutions obtained from a CG algorithm, it can be extended to a branch-and-price algorithm where in each node of a branch-and-bound tree, variables with positive reduced cost (in a maximization problem) are generated by CG. We now describe the general structure of the branch-and-price-and-cut algorithm, and then we develop branching and cutting-plane procedures.

4.5.1 Structure of the Branch-and-Price-and-Cut Algorithm

Algorithms 4.1 and 4.2 describe the price-and-cut and branch-and-price-and-cut procedures. In Algorithm 4.1, two search phases are possible. If *CheckOptimalityPhase* is 0, it means that in solving the subproblems, the constraint programming solver will stop after *CPtimelimit* seconds. When *CheckOptimalityPhase* is set to 1, the constraint programming solver is allowed to solve the subproblems without any time limit. The algorithm switches from the first phase to the second only if no column is generated in an iteration (Line 16). It is also possible that the algorithm switches to the first phase if some columns are generated in the second phase (Line 21). If no column is generated when *CheckOptimalityPhase* is 1, *StoppingCriterion* will be set to 1 in Line 18, terminating the while loop. In this case, the objective value of the relaxed master problem will be a valid upper bound. This careful use of a time limit is vital for the efficiency of the algorithm. The cuts added in Line 6 will be explained in Section 4.5.3. In Line 10 of Algorithm 4.1, when the constraint programming solver finds a solution with a positive objective value a new column is generated and the constraint programming algorithm continues its search to generate other columns with larger positive reduced costs. In Line 12, we check the columns generated in this iteration to see if they also have positive reduced costs on other days. If they do, similar schedule patterns will be generated for the other days if they respect the surgery deadlines. Finally, we solve the master problem restricted to generated columns with binary variables to find a lower bound (Line 24). The initial columns for the restricted master problem are generated by a constructive heuristic ; see Section 4.6.1.

Algorithm 4.1. (Price-and-cut algorithm)

```
1:  $StoppingCriterion = 0$ ; $CheckOptimalityPhase = 0$ ; $CPtimelimit = 10s$
2:  while ($StoppingCriterion == 0$) do
3:    Solve the relaxed master problem.
4:    Check for violated minimal cover cuts.
5:    while (there are violated minimal cover cuts) do
6:      Add a lifted minimal cover inequality to the master problem.
7:      Solve the relaxed master problem.
8:    end while
9:    for (all subproblems $d \in \mathcal{D}$) do
10:     Solve the subproblem using constraint programming.
11:   end for
12:   Check if the generated columns can be copied to other days.
13:   Add all columns generated in this iteration to the master problem.
14:   if (no columns are generated) then
15:     if ($CheckOptimalityPhase == 0$) then
16:       $CheckOptimalityPhase = 1$ ; $CPtimelimit = inf$ ;
17:     else
18:       $StoppingCriterion = 1$;
19:     end if
20:   else if ($CheckOptimalityPhase == 1$)
21:     $CheckOptimalityPhase = 0$ ; $CPtimelimit = 10s$ ;
22:   end if
23: end while
24: Solve the restricted master problem with integer variables.
```

In the branch-and-price-and-cut algorithm presented by Algorithm 4.2, lines 3, 6, and 9 implement branches (1)–(3) ; see Section 4.5.2. The algorithm stops when a predetermined time limit is reached or the lower and upper bounds are the same, confirming the optimality of the current integer solution. The node selection (Line 5) is based on a best-first strategy. If no branching is possible, the restricted master problem with integer variables is solved (Line 11) to find a lower bound. If a predetermined time limit is reached (Line 13), the algorithm computes a valid upper bound based on the upper bounds for those nodes for which the LR-RMP has been solved to optimality. The algorithm subsequently switches to fast branching, i.e., *CheckOptimalityPhase* is set to 1 in Algorithm 4.1. This allows the algorithm to rapidly generate many columns by visiting many nodes in the tree instead of spending time proving the optimality of the generated nodes. In fact, generating many columns using the prior strategy increases the chance of getting a good integer solution in Line 16.

Algorithm 4.2. (Branch-and-price-and-cut algorithm)

```
1:  UpperBound = $+\infty$ ; LowerBound = $-\infty$ ;
2:  Apply *Price-and-Cut algorithm* at the root node.
3:  Check and save possible branching at the root node ;
4:  **while** (stopping criteria are not activated) **do**
5:    Select a node from the list of feasible nodes ;
6:    **if** (branching is possible) **then**
7:      Branch and generate child nodes ;
8:      Apply *Price-and-Cut algorithm* to the generated nodes.
9:      Check and save possible branching at generated nodes ;
10:   **else**
11:     Solve the restricted master problem in the current node with integer variables.
12:   **end if**
13:   Check for switch to fast branching ;
14:   Check stopping criteria ;
15: **end while**
16: Solve the restricted master problem with all generated columns and integrality constraints.
```

4.5.2 Branching procedure

We use three types of branching :

(1) Branching on the number of total scheduled surgeries in the planning horizon.

(2) Branching on the number of scheduled surgeries for a particular surgeon in the planning horizon.

(3) Branching on whether two particular surgeries are scheduled in the same operating room (Ryan and Foster 1981).

Branches (1) and (2) require a new constraint in the master problem. For branch (3) some columns must be removed from the master problem, and some constraints must be added to the subproblem. Branches (1) and (2) are implemented as traditional dichotomic branches generating two new nodes. For branch (1), we add one of the following constraints to the master problem :

$$\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}} a'_{ij}x_j \geq \lceil m \rceil, \tag{4.30}$$

$$\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}} a'_{ij}x_j \leq \lfloor m \rfloor, \tag{4.31}$$

where m is the fractional number of scheduled surgeries in LR-RMP. For branch (2) for surgeon l, we add one of the following constraints to the master problem :

$$\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}'_l} a'_{ij}x_j \geq \lceil m'_l \rceil, \tag{4.32}$$

$$\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}'_l} a'_{ij}x_j \leq \lfloor m'_l, \rfloor, \tag{4.33}$$

where m'_l is the fractional number of scheduled surgeries for surgeon l in LR-RMP. In the computation of matrix G_{di} in (4.18), $\Delta_{id}^{\text{best}}$, $\Delta_{id}^{\text{worst}}$, and $\Delta_{idp}^{\text{best}}$ must be replaced by $g_i(G_{di})$, $g_i(\Delta_{id}^{\text{best}})$, $g_i(\Delta_{id}^{\text{worst}})$, and $g_i(\Delta_{idp}^{\text{best}})$, where

$$g_i(h) := \begin{cases} h - \pi^{(30)} - \pi^{(31)}, & i \neq 0,\ s_i \neq l, \\ h - \pi^{(30)} - \pi^{(31)} - \pi_{s_i}^{(32)} - \pi_{s_i}^{(33)}, & i \neq 0,\ s_i = l, \\ h, & i = 0, \end{cases}$$

and $\pi^{(30)}$, $\pi^{(31)}$, $\pi^{(32)}$, and $\pi^{(33)}$ are the dual variables of constraints (4.30)–(4.33). Branch (3), introduced by Ryan and Foster (1981), gives better integer solutions than the other branches give. In one of the child nodes, two surgeries i and i' must be scheduled in the same operating room, and we remove from the master problem all the columns in which this is not the case. In the other child node, surgeries i and i' cannot be scheduled in the same operating room, and we remove some columns accordingly. We also add constraint (4.34) to the first child node and (4.35) to the second :

$$C_i = C_{i'}, \tag{4.34}$$

$$C_i + C_{i'} \leq 1. \tag{4.35}$$

As computationally shown in Appendix A.5, branches (1) and (2) usually improve the upper bound, whereas branch (3) is usually more effective in generating good columns, thus increasing the likelihood of finding good integer solutions. Therefore, to improve lower and upper bounds with the proposed branching rules, they are applied in the following order : We apply branch (1) first. For branch (2), we select the surgeon for which the fractional part of the number of scheduled surgeries is closest to 0.5. Finally, in branch (3), we select the pair of surgeries for which the fractional number of times that they appear together is closest to 0.5.

4.5.3 Cutting planes

To improve the quality of the upper bound obtained by CG algorithm, a class of cutting planes is added to the master problem iteratively. We use the lifted minimal cover inequality (Gu et al. 1998, Atamtürk 2005) based on a reformulation of the knapsack capacity constraints (4.5), which are the constraints on the maximum daily hours of the surgeons. Barnhart et al. (2000) used the same idea to improve the quality of a branch-and-price algorithm for a multicommodity flow problem. We introduce an auxiliary binary variable z_{id} that is 1 if surgery i is scheduled on day d and 0 otherwise. Constraint (4.5) can now be reformulated as follows :

$$\sum_{i\in\mathcal{I}'_l} t_i z_{id} \le A_{ld}, \qquad d\in\mathcal{D},\ l\in\mathcal{L}. \tag{4.36}$$

The corresponding lifted minimal cover inequalities are

$$\sum_{i\in\mathcal{I}'_l\setminus\mathcal{C}} \alpha_i z_{id} + \sum_{i\in\mathcal{C}} z_{id} \le |\mathcal{C}|-1, \qquad d\in\mathcal{D},\ \ l\in\mathcal{L},\ \ \mathcal{C}\in\Psi_{ld}, \tag{4.37}$$

where $\mathcal{C}$ is a minimal cover set for surgeon l on day d. This is the minimal subset of surgeries for surgeon l that cannot be scheduled together on day d because of the limited knapsack capacity (A_{ld}). The set Ψ_{ld} contains all minimal cover sets on day d for surgeon l. To impose these cutting planes we express z_{id} in terms of x_j via $z_{id} = \sum_{j\in\mathcal{J}'_d\cap\mathcal{J}_i} x_j$. The cuts to be added to the master problem are as follows :

$$\sum_{i\in\mathcal{I}'_l\setminus\mathcal{C}}\ \sum_{j\in\mathcal{J}'_d\cap\mathcal{J}_i} \alpha_i x_j + \sum_{i\in\mathcal{C}}\ \sum_{j\in\mathcal{J}'_d\cap\mathcal{J}_i} x_j \le |\mathcal{C}|-1, \quad d\in\mathcal{D},\ l\in\mathcal{L},\ \mathcal{C}\in\Psi_{ld}. \tag{4.38}$$

During the CG procedure, whenever we detect a violated cut, it is added to the master problem. To detect violated cuts, we use the default lifted cover inequality (LCI) algorithm of Gu et al. (1998). However, because of time efficiency instead of the exact lifting algorithm, we use a very fast lifting procedure from Atamtürk (2005) to generate LCI. This procedure is as follows :

$$\alpha_i \;=\; \begin{cases} r, & \theta_r \le t_i \le \theta_{r+1}-1 \quad (r\in\{1,2,\ldots,|\mathcal{C}|-1\}), \\ |\mathcal{C}|, & t_i \ge \theta_{|\mathcal{C}|}, \end{cases} \qquad \forall i\in\mathcal{I}'_l\setminus\mathcal{C}, \tag{4.39}$$

where t_i is the duration of surgery i. Here $\theta_r = \sum_{k=1}^{r} t_{c_k}$, $\forall\, r\in\{1,2,\ldots,|\mathcal{C}|\}$, and c_k is the kth surgery in the minimal cover $\mathcal{C}$ when the surgeries in $\mathcal{C}$ are sorted in nondecreasing order of durations.

The main concern when adding cuts to a CG algorithm is how to consider the effect of the dual values of the added cuts in the objective function of the subproblems. To address this, we define Γ_{di} to be the set of all added cuts that include surgery i on day d with a nonzero coefficient for z_{id}. In each subproblem $d \in \mathcal{D}$, whenever a new column that includes surgery i is examined, the effect of the dual values of all the cuts in Γ_{di} must be taken into account in the reduced-cost computation. We define λ_{di} to be the contribution of the dual values of the added cuts in the objective function of subproblem d due to the inclusion of surgery i on day d :

$$\lambda_{di} = \sum_{v \in \Gamma_{di}} \pi_v \alpha'_{iv}, \qquad i \in \mathcal{I},\ d \in \mathcal{D}, \tag{4.40}$$

where π_v is the dual value of cut $v \in \Gamma_{di}$ and α'_{iv} is the coefficient of z_{id} in this cut. After we solve the master problem, we compute the matrix $\lambda = [\lambda_{di}]$ and replace (4.18) by the following objective function :

$$\max\left\{\sum_{p=1}^{n} G_{d[W_p]} - \sum_{p=1}^{n} \pi^*_{d[W_p][V_p]} - \sum_{p=1}^{n} \lambda_{d[W_p]} - \pi_d^{(4)}\right\}.$$

This is equivalent to the aggregation of matrix λ in matrix A in (4.18) as follows : $G_{di} = t_i - \pi_i^{(2,3)} - t_i \pi_{ds_i}^{(5)} - \lambda_{di}$ (for $i \neq 0$). Further, $\Delta_{id}^{\text{best}}$, $\Delta_{id}^{\text{worst}}$, and $\Delta_{idp}^{\text{best}}$ must be updated via $\Delta_{id}^{\text{best}} := \Delta_{id}^{\text{best}} - \lambda_{di}$, $\Delta_{id}^{\text{worst}} := \Delta_{id}^{\text{worst}} - \lambda_{di}$, and $\Delta_{idp}^{\text{best}} := \Delta_{idp}^{\text{best}} - \lambda_{di}$.

We define the lifted minimal cover inequalities on constraint (4.36) and not constraint (4.5) because in the former the effect of the dual values of cuts can be considered in the objective function of the subproblems, whereas in the latter there is no effective way to do this. We also tested clique cuts and odd-cycle inequalities, but because of the difficulty in efficiently considering the dual values of the variables in these cuts, they were not very effective and we do not discuss them further.

4.6 Computational Experiments

We implemented the algorithm in IBM ILOG CPLEX Optimization Studio V12.4, which includes CPLEX and CP Optimizer to solve linear, integer, and constraint programming models. Experiments were run on a computer with two Intel Xeon X5675 processors, 3.07 GHz, and a total of 12 cores. We ran different instances using different cores.

4.6.1 Instances

We generated three sets of instances with different specifications. In all the instances, the planning horizon is five days of a week and the maximum hours for the surgeons (A_{ld}) are taken from Table 1 of (Fei et al. 2009) with some modifications. Fei et al. (2009) allowed some surgeons to work overtime beyond eight hours but we do not, and the corresponding entries are decreased to eight hours. In our instances, the average maximum time for surgeons is about 320 minutes with a variance of roughly 20 minutes (i.e., $\text{Var}[(\sum_{d \in \mathcal{D}} A_{ld}/|\mathcal{D}|)_{l \in \mathcal{L}}] =$ 20 min). The surgery deadlines are uniformly generated in the interval $[1, 14]$ as suggested by Fei et al. (2009). Surgeries with deadlines after the fifth day are optional. The surgeries are randomly assigned to eight surgeons, as suggested by Fei et al. (2009). We randomly assigned half of the surgeries in each instance to the infectious category, and we set the cleaning time (OCT) to 30 minutes.

In the first set of instances, denoted A, the surgery durations are uniformly generated in the interval [2 hours, 4 hours] and time is discretized into five-minute units. Six operating rooms are available for eight hours each day. Usually the number of rooms is large enough that this resource does not happen to be the bottleneck of the system and the availability of the surgeons is more restrictive.

Instance set B is the same as A except that the number of operating rooms is set using a heuristic (see Appendix A.6). The heuristic computes an approximate average of the daily working hours of each surgeon, assuming that the surgeon is going to perform all the surgeries and the workload is distributed evenly over the planning horizon. It then uses these values to compute the operating-room hours required daily. From this we derive the number of operating rooms required. The main characteristic of instance set B is that the number of operating rooms can be restrictive.

In instance set C, the surgery durations are generated according to the Pearson III distribution, as explained by Fei et al. (2009). The daily numbers of operating rooms are again set using the heuristic. The average durations of the surgeries in this set is considerably lower than those of the two previous sets.

The three instance sets cover a wide range of operating room planning and scheduling problems in different surgical departments. Sets A and B are good representatives of cases that surgeries take more than two hours, such as cardiac surgery, bile duct and liver surgery, and various types of vascular surgery (Leong et al. 2006). Set C is representative of cases with shorter surgery durations such as hemiarthroplasty, limb amputation, and abdominal hysterectomy which commonly take less than two hours (Leong et al. 2006). We set the number of surgeries in A and B to $\{40, 60, 80, 100, 120\}$ and the number in set C to $\{60, 80, 100\}$. We generate five instances for each problem setting for a total of 65 instances. We do not consider 40 or 120 surgeries in set C because in the former case, due to short surgery durations, only one operating room is required based on the heuristic presented in Appendix A.6 and in the latter neither the proposed algorithm nor the integer programming and constraint programming models presented in Appendices A.1 and A.2 are very efficient.

To ensure feasibility of generated instances considering surgery deadlines, we apply a constructive heuristic to generate an initial solution for each instance. In this heuristic whenever a surgery deadline cannot be respected its deadline is postponed to the next day. The heuristic fills the operating rooms consecutively starting from the first operating room on the first day to the last one on the last day. At each step it assigns a single surgery to an operating room. For scheduling surgeries on each day the surgeons are considered in a random order, and the surgeries are sorted by deadline. Two criteria break ties : (1) Surgeries performed by more important surgeons according to a random importance assignment are considered before other surgeries for the assignment. (2) If ties remain, the tied cases are sorted by decreasing order of processing time. In each operating room earlier time slots are favored as the start time of surgeries. The selected start time must lead to a feasible solution for the other surgeries of the same surgeon, i.e., all coloring constraints and maximum working hours must be respected. The resulting schedules define the initial columns of the master problem.

4.6.2 Parameters

The computational results are presented in Tables 4.2–4.8. We set the CG time limit to two hours, and we set *CPtimelimit*, one of the parameters of the Algorithm 4.1 explained in Section 4.5.1, to 10 seconds. If the CG does not converge within the time limit we report a trivial upper bound, $\min(\sum_{d\in\mathcal{D}} |\mathcal{K}_d||\mathcal{T}_d|, \sum_{d\in\mathcal{D}}\sum_{l\in\mathcal{L}} A_{ld}, \sum_{i\in\mathcal{I}} t_i)$. We then solve the integer programming model of the restricted master problem with a time limit of one hour to find a lower bound (Line 24 of Algorithm 4.1). In the case of a branch-and-price or a branch-and-price-and-cut, we allow an additional two hours for the branching procedure. In the first 15 minutes, we try to prove the optimality of the nodes. We then compute an upper bound of the tree and the optimality of generated nodes will not be proven anymore and child nodes of a given node are generated as the CG algorithm switches to $\mathit{CheckOptimalityPhase} = 1$. After the branching procedure, we apply the integer programming model solver for one hour with all the generated columns (Line 16 of Algorithm 4.2).

4.6.3 Results

In Tables 4.2–4.8, we give the computational times in seconds ; each row presents the average over five instances. In Table 4.2, we evaluate the dominance rules and the fast infeasibility-detection algorithm. Under columns "*Original CG*" and "*Enhanced CG*" the computational results of the CG algorithm without applying any of enhancements and with all of them are presented, respectively. In the enhanced CG, dominance rules and the fast infeasibility-detection algorithm are implemented dynamically and are called whenever a W_p variable is fixed. We give the number of CG iterations (*No. of iterations*) and the number of instances for which the CG has not converged within the time limit (*No. of unconverged*). The results in columns 5–12 are obtained by averaging over all the subproblems and all the CG iterations at the root node of the constraint programming search tree, (i.e., when no W_p variable is fixed yet and $p^* = 1$) although the dominance rules and the fast infeasibility-detection algorithm are implemented dynamically. This provides a good sense of the effectiveness of these features.

"*DR* 1" gives the percentage of values in the domains of V_1 removed by Dominance rule 1. "*DR* 2," "*DR* 3," and "*DR* 4" give the percentages of surgeries in the subproblems removed by dominance rules 2–4. When a rule removes a surgery from the domain of W_{p^*}, that surgery is not considered by the subsequent rules. The dominance rules are applied sequentially as previously mentioned. This is reasonable because Dominance Rule 2 is the least time-consuming : it is executed in $O(|\mathcal{I}|)$. Dominance Rule 2 removes some of the surgeries from the domain of W_{p^*} before Dominance Rule 1 (which runs in $O(|\mathcal{I}|^2)$) examines all pairs of available surgeries. Dominance Rules 1(b) and 1(c) add some constraints in the form of constraint (4.22) that increases the number of surgeries in the set $\mathcal{P}_i$, thus making Dominance Rule 4 more effective. "*Conditioned values*" gives the percentage of surgeries in the subproblems that are conditioned by constraint (4.22) but not removed by any of the dominance rules. "*FID* 1" gives the percentage of infeasible subproblems detected by the fast infeasibility-detection algorithm. "*FID* 2" gives the percentage of infeasible subproblems over all the CG iterations that are detected because all the surgeries are removed from the domains of W_p by the application of Dominance Rules 2–4 at the root node of the constraint programming search tree (i.e., when no W_p variable is fixed yet and $p^* = 1$). Column "*Stopped by knapsack bound*" gives the percentage of subproblems for which the constraint programming solver stopped because it found a solution with an objective value equal to the upper bound imposed by constraint (4.29) for the case where $p^* = 1$. Table 4.2 shows that

(1) 31% of the surgeries are removed by the dominance rules ;

(2) 16.22% of the surgeries are conditioned by constraint (4.22) ;

(3) about 14% of subproblems are detected to be infeasible before letting the constraint programming solver do any propagation ;

(4) 8.58% of the subproblems also stopped as they reach the knapsack bound of constraint (4.29) ; and

(5) the dominance rules and fast infeasibility-detection algorithm reduce the number of unconverged instances from 36 to 7.

We observe that the effects of the dominance rules and fast infeasibility-detection algorithm vary ; this is especially noticeable for "*FID* 2."

Tables 4.3–4.6 present results for CG algorithms, price-and-cut, branch-and-price, and branch-and-price-and-cut. We apply the dominance rules and the fast infeasibility detection algorithm unless otherwise stated. The columns are : (1) "*No. of columns*" : the total number of columns generated. (2) "*Alg. time*" : the computational time, excluding the time to find a lower bound at the end of the CG and the end of the branching. (3) "*LB time*" : the time to solve the restricted master problem to obtain a lower bound. (4) "*Total time*" : the sum of "*Alg. Time"* and "*LB time*", (5) "*UB*" : the upper bound. (6) "*LB"* : the lower bound. (7) "*Gap*" : the optimality gap, i.e., $100(UB - LB)/UB$. (8) "*No. of cuts*" : the number of cuts added. (9) "*No. of nodes* (*optimality proved*)" : the number of nodes for which optimality is proved. (10) "*No. of nodes* (*all nodes*)" : the total number of nodes generated.

Table 4.3 shows that the dominance rules and the fast infeasibility-detection algorithm decrease the computational time from 4,477 seconds to 1,715 seconds, but for most instances in sets A and B the quality of integer solutions obtained from the original CG is better than that of integer solutions resulted from the enhanced CG. This is because of the larger number of columns generated in the earlier algorithm which let the integer programming model called at the end of the algorithm have more choices to get good integer solutions, whereas the second algorithm converges faster with fewer number of columns as a result of applying dominance rules. However, in 21 instances of sets A and B the original CG has not converged and trivial upper bounds are reported. For set C, the original CG struggles to solve the subproblems ; the enhanced CG generates more columns and finds a better lower bound.

As can be seen in Tables 4.3–4.6, the average optimality gaps for original CG, enhanced CG, price-and-cut, branch-and-price, and branch-and-price-and-cut are 9.90%, 7.82%, 4.76%, 5.36%, and 2.81% respectively. In Table 4.6, the "*initial objective value"* is obtained from the constructive heuristic previously explained in this section ; our algorithm significantly improves these values.

Table 4.2 – Evaluation of Dominance Rules and the Fast Infeasibility-Detection Algorithm.

		Original CG		Enhanced CG									
Instance set	No. of surgeries	No. of iterations	No. of unconverged	DR 1 (%)	DR 2 (%)	DR 3 (%)	DR 4 (%)	*Conditioned values(%)*	FID 1 (%)	FID 2 (%)	Stopped by knapsack bound(%)	No. of iterations	No. of unconverged
A	40	13	0	25.21	67.41	3.95	0.88	3.44	9.61	28.32	11.72	16	0
	60	26	0	25.00	61.64	4.25	0.84	3.77	29.96	13.61	7.83	30	0
	80	57	0	25.00	43.14	5.18	1.16	4.77	20.96	3.42	6.77	41	0
	100	122	5	25.21	38.86	5.15	0.82	4.64	27.72	0.15	3.58	76	0
	120	125	5	25.00	34.97	6.19	1.45	6.56	24.28	0.00	4.65	62	0
B	40	13	0	25.17	4.96	5.88	1.99	12.11	0.00	0.00	11.08	17	0
	60	22	0	25.21	3.22	8.61	2.79	10.37	0.00	0.00	8.15	25	0
	80	57	1	25.00	2.72	7.49	1.84	9.05	0.00	0.00	5.55	39	0
	100	97	5	25.00	4.59	8.39	1.13	6.20	7.79	0.00	2.70	68	0
	120	95	5	25.00	3.65	9.45	2.12	6.45	3.74	0.00	2.55	89	3
C	60	61	5	10.5	4.65	5.50	7.71	48.93	2.27	0.00	23.80	25	1
	80	43	5	9.44	4.18	6.76	9.02	47.10	7.93	0.00	12.50	55	1
	100	69	5	9.17	4.08	6.00	11.99	47.51	4.35	0.00	10.65	50	2
Average :	-	61	-	21.53	21.39	6.37	3.36	16.22	10.66	3.50	8.58	46	-
Total :	-	-	36	-	-	-	-	-	-	-	-	-	7

Table 4.3 – Evaluation of the CG algorithms.

		Original CG							Enhanced CG						
Instance set	No. of surgeries	No. of columns	Alg. time	LB time	Total time	UB	LB	Gap (%)	No. of columns	Alg. time	LB time	Total time	UB	LB	Gap (%)
A	40	627	48	0	48	1431	1395	2.44	357	25	0	25	1431	1384	3.20
	60	1396	532	46	578	2020	1941	3.92	877	108	2	110	2020	1865	7.72
	80	2204	2966	11	2977	2307	2123	7.94	1386	407	7	414	2307	2077	9.98
	100	3004	7200	12	7212	2880	2195	23.79	1619	1960	14	1974	2401	2159	10.04
	120	2606	7200	5	7205	2880	2183	24.19	1532	1337	8	1345	2423	2183	9.91
B	40	692	79	1	80	1371	1352	1.35	448	35	1	36	1371	1355	1.12
	60	1243	605	9	614	1866	1807	3.17	812	137	5	142	1866	1799	3.59
	80	2041	3569	176	3745	2207	2083	5.58	1220	736	23	759	2205	2067	6.22
	100	2894	7200	523	7723	2438	2276	6.67	1921	2349	373	2722	2416	2271	6.01
	120	2861	7200	54	7254	2496	2206	11.61	1878	4807	155	4962	2488	2221	10.71
C	60	571	7200	0	7200	576	507	11.98	741	1964	4	1968	576	532	7.71
	80	412	7200	0	7200	792	688	13.11	1307	3508	4	3512	792	692	12.58
	100	480	7200	0	7200	979	853	12.91	1878	4916	7	4923	979	853	12.89
Average :			4477					9.90		1715					7.82

Table 4.4 – Evaluation of the price-and-cut algorithm.

Instance set	No. of surgeries	Price-and-cut							
		No. of columns	No. of cuts	Alg. time	LB time	Total time	UB	LB	Gap (%)
A	40	474	31	35	0	35	1411	1402	0.61
	60	1093	72	129	10	139	1975	1897	3.90
	80	1833	118	424	10	434	2205	2141	2.88
	100	2498	134	2158	47	2205	2297	2218	3.44
	120	2591	199	1055	10	1065	2301	2231	3.01
B	40	546	29	42	1	43	1367	1355	0.88
	60	1117	73	187	18	205	1860	1819	2.13
	80	2206	105	1662	97	1759	2173	2111	2.85
	100	3036	144	4633	1139	5772	2402	2320	3.38
	120	3173	178	5002	315	5317	2457	2306	6.14
C	60	754	0	2329	9	2338	576	532	7.57
	80	1230	0	6034	3	6037	792	693	12.53
	100	1787	0	4729	11	4740	978	855	12.56
Average :									4.76

Table 4.5 – Evaluation of the branch-and-price algorithm.

Instance set	No. of surgeries	Branch-and-Price								
		No. of columns	No. of nodes (optimality proved)	No. of nodes (all nodes)	Alg. time	LB time	Total time	UB	LB	Gap (%)
A	40	631	397	821	182	1	183	1421	1400	1.49
	60	6399	20	507	7323	1441	8764	2017	1957	2.99
	80	4660	21	966	6590	288	6878	2308	2151	6.73
	100	4211	8	135	9170	79	9249	2401	2217	7.64
	120	3557	12	152	8552	30	8582	2471	2234	9.58
B	40	2776	64	781	7238	6	7244	1366	1361	0.39
	60	3036	32	388	7344	323	7667	1862	1841	1.17
	80	2523	13	416	7945	88	8033	2202	2114	3.97
	100	2778	4	455	9532	1068	10600	2416	2303	4.68
	120	2853	1	65	12033	306	12339	2488	2296	7.72
C	60	3964	10	356	9175	1101	10276	576	570	1.08
	80	4030	5	102	10739	2568	13307	792	712	10.08
	100	5348	2	84	12141	2957	15098	977	858	12.20
Average :										5.36

Table 4.6 – Evaluation of the branch-and-price-and-cut algorithm.

Instance set	No. of surgeries	Initial objective value (heuristic)	Branch-and-price-and-cut									
			No. of columns	No. of cuts	No. of nodes (optimality proved)	No. of nodes (all nodes)	Alg. time	LB time	Total time	UB	LB	Gap (%)
A	40	1321	785	34	3	39	83	1	84	1403	1403	0.00
	60	1814	9379	132	19	300	7353	1306	8659	1971	1960	0.55
	80	1894	7015	152	15	137	7639	864	8503	2203	2174	1.30
	100	1968	5413	178	6	80	9381	222	9603	2297	2255	1.81
	120	2014	5487	230	6	66	8273	22	8295	2298	2266	1.40
B	40	1193	1781	63	41	225	2262	207	2469	1361	1361	0.00
	60	1601	3475	131	21	191	7398	779	8177	1853	1843	0.55
	80	1824	4796	141	11	89	8878	906	9784	2161	2132	1.32
	100	1993	4819	168	2	46	11862	2814	14676	2401	2337	2.65
	120	2004	5120	215	2	43	12233	1157	13390	2457	2355	4.13
C	60	507	4216	0	5	141	9547	1786	11333	576	571	0.94
	80	688	3745	2	2	85	13261	2319	15580	792	713	9.92
	100	853	4929	0	2	105	11956	3589	15545	978	860	12.01
Average :												2.81

Table 4.7 – Evaluation of the integer programming model, constraint programming model, and mixed algorithm.

Instance set	No. of surgeries	IP				CP						CP-IP			
		Time	UB	LB	Gap (%)	No. of branches (millions)	No. of fails (millions)	Time	UB	LB	Gap (%)	Time	UB	LB	Gap (%)
A	40	300	1415	1112	20.97	2.71	1.12	300	1446	1366	5.51	257	1411	1399	0.86
	60	11000	1966	1941	1.20	70.29	28.24	11000	2150	1693	21.23	10401	1974	1879	4.64
	80	11100	2220	1670	24.76	58.74	23.25	11100	2436	1947	20.07	11100	2225	1995	10.31
	100	11600	2302	851	63.15	44.74	16.79	11600	2436	2058	15.50	11600	2316	2101	9.20
	120	8800	2338	421	81.10	29.13	10.58	8800	2436	2041	16.22	8800	2351	2056	12.52
B	40	4400	1361	1304	4.18	46.17	20.56	4400	1402	1285	8.27	4117	1365	1323	3.09
	60	11000	1851	1700	8.11	92.89	40.00	11000	1920	1673	12.85	11000	1856	1739	6.21
	80	12800	2180	1439	33.90	70.61	28.58	12800	2224	1841	17.12	12800	2189	1898	13.26
	100	18200	2400	0	100.00	77.18	28.95	18200	2433	2007	17.50	18200	2408	2035	15.51
	120	17800	2430	0	100.00	67.26	23.87	17800	2484	2140	13.86	17800	2434	2133	12.39
C	60	14800	576	538	6.63	125.40	54.27	14800	576	534	7.22	14800	576	547	4.97
	80	18200	792	267	66.24	161.85	68.96	18200	792	611	22.80	18200	792	718	9.29
	100	18200	978	0	100.00	112.69	60.74	18200	979	487	51.65	18200	978	487	51.62
Average :					46.94						17.68				11.84

Table 4.8 – Improvement in optimality gap from cuts and branching.

Instance set	No. of surgeries	Improvement from cuts (%)	Improvement from branching (%)	Improvement from branching and cuts (%)
A	40	2.59	1.71	3.20
	60	3.82	4.74	7.18
	80	7.10	3.24	8.67
	100	6.61	2.41	8.23
	120	6.90	0.33	8.51
B	40	0.24	0.73	1.12
	60	1.45	2.42	3.03
	80	3.37	2.24	4.89
	100	2.63	1.33	3.36
	120	4.57	2.98	6.58
C	60	0.14	6.63	6.77
	80	0.05	2.50	2.65
	100	0.33	0.69	0.88
Average :		3.06	2.46	5.01

Depending on the time limits in practical cases, one may decide to use the branch-and-price-and-cut or the price-and-cut algorithms that require two to four hours for large instances. If less time is available, the practitioner could develop a heuristic and validate it by comparing it with our algorithms.

Table 4.7 gives the results for the integer programming model (*IP*) and the constraint programming model (*CP*) developed in Appendices 1 and 2. For each row we set the time limit to the maximum time for the same instances in Table 4.6. As the constraint programming model does not provide upper bounds, the trivial upper bounds of the instances are reported for this algorithm. Column *CP-IP* gives the results for a mixed algorithm that uses CP for half of the available time and then switches to IP. The IP solver can prune more quickly by starting with the CP solution, and the heuristics in the CPLEX solver use it to find better solutions. The average optimality gaps are large : 46.94%, 17.68%, and 11.84% for IP, CP, and CP-IP respectively which are weaker than the average optimality gap 2.81% obtained by the proposed branch-and-price-and-cut algorithm. Before running IP, we tuned its search parameters using CPLEX's automatic tuning tool. We randomly chose three small instances from each of sets A, B, and C, and CPLEX examined different parameter settings to find the most reliable setting over the nine instances ; the time limit was three hours for each instance. Table 4.8 presents the improvement in the optimality gap achieved by adding cuts, branching, or both to the enhanced CG. Adding both cuts and branching improves the gap by 5.01% on average. In Appendix A.5 we evaluate the different types of branches.

4.7 Conclusion and Future Research

Integrated operating room planning and scheduling (IORPS) can improve the efficient use of operating theatres by synchronizing the assignment and scheduling of surgeries. Operational details can be considered at the scheduling level, increasing the chance of obtaining a stable schedule that can be successfully implemented. Since patients, surgeons, nurses, anesthesiologists, and operating-room managers all play a role, there are many constraints to take into account. We have developed a branch-and-price-and-cut algorithm for the IORPS which is capable of addressing various kinds of problem settings. To improve the efficiency of the algorithm, a constraint programming model proposed to solve subproblems is enhanced by some dominance rules and a fast infeasibility-detection algorithm. Our model reduces the number of subproblems at each CG iteration, so we can solve large instances. Extensive computational results demonstrate the superiority of the developed method to a compact mathematical programming model adapted from the literature and show that our algorithm can obtain solutions with an average optimality gap of 2.81% for instances with up to 120 surgeries. Future research could explore the following two issues :

1. Two sources of uncertainty could be considered : the durations of the surgeries and the arrival times of emergency cases. Similar problems have been studied under a block-scheduling strategy. However, to the best of our knowledge, only Batun et al. (2011) have studied the problem under an open-scheduling strategy, and they solve instances with at most 11 surgeries. More powerful algorithms are needed for stochastic IORPS under an open-scheduling strategy.
2. Integration of the tactical and the operational levels can be another interesting opportunity where the number of blocks assigned to each surgical department and the start and finish times of these blocks could be determined concurrently with the IORPS.

CHAPTER 5 ARTICLE 2: VEHICLE ROUTING PROBLEMS WITH SYNCHRONIZED VISITS AND STOCHASTIC/TIME-DEPENDENT TRAVEL AND SERVICE TIMES : APPLICATIONS IN HEALTHCARE

Seyed Hossein Hashemi Doulabi

Department of Mathematics and Industrial Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Gilles Pesant

Department of Computer and Software Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Louis-Martin Rousseau

Department of Mathematics and Industrial Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Abstract. This paper, for the first time, studies vehicle routing problems with synchronized visits (VRPS) where travel and service times are stochastic/time-dependent. In addition to considering a home-health care scheduling problem, we introduce an operating room scheduling problem with stochastic durations as a novel application of VRPS. We formulate VRPS with stochastic times as a two-stage stochastic programming model with integer variables in both stages. An advantage of the proposed model is that, in contrast to the deterministic models in the VRPS literature, it does not have any big-M constraints. This advantage comes at the cost of a large number of second-stage integer variables. We prove that the integrality constraints on second-stage variables are trivial, and therefore we can apply the L-shaped algorithm and its branch-and-cut implementation to solve the problem. We enhance the model by developing valid inequalities and a lower bounding functional. We analyze the subproblems of the L-shaped algorithm and devise a solution method for them that is significantly faster than standard linear programming algorithms. Moreover, we extend our model to formulate VRPS with time-dependent travel and service times. Computational results show that, in the stochastic home-health care scheduling problem, the branch-and-cut algorithm optimally solves instances with 15 patients and 10% to 30% of synchronized visits. It also finds solutions with an average optimality gap of 3.57% for instances with 20 patients. Furthermore, the branch-and-cut algorithm optimally solves stochastic operating room scheduling problems with 20 surgeries, a considerable improvement over the literature that reports on instances with 11 surgeries. In addition, the proposed formulation for the time-dependent problem solves a large portion of home-health care scheduling instances with 30 to 60 patients and different rates of synchronized visits to optimality.

Keywords. Vehicle routing with synchronized visits, stochastic/time-dependent travel and service times, home-health care scheduling, operating room scheduling.

5.1 Introduction

In the literature of Vehicle Routing Problem (VRP), a large number of studies have taken into account the scheduling of services to customers in addition to the routing of vehicles. Such routing problems are generally referred to as routing and scheduling problems. The most well-known problem in this area is VRP with Time Windows (VRPTW) where a number of vehicles must serve customers with minimum cost while satisfying some time window constraints. Recently there has been an emergent interest in VRP with Synchronized visits (VRPS) in which two or more vehicles of different types must be simultaneously available at the customer's location for serving. VRPS is significantly more complicated than the classic VRPTW because the scheduling of vehicles is interdependent, and therefore finding a quality solution satisfying all time window constraints is more challenging. This problem has a wide range of applications in home-health care scheduling (Bredström and Rönnqvist 2008, Di Mascolo et al. 2014), raw milk collection (Drexl and Sebastian 2007), staff scheduling (Lim et al. 2004, Li et al. 2005), garbage collection (De Rosa et al. 2002), forest management (Paraskevopoulos et al. 2016), and telecommunication (Jaumard et al. 2016). We refer readers to Drexl (2012) for VRP with other types of synchronizations.

In routing and scheduling problems, two important factors significantly increase problem complexity. The first factor is uncertainty in travel and service times that has been dealt with in the literature using stochastic and robust optimization approaches. Some researchers studied Travelling Salesman Problems (TSPs) with independent and normally distributed travel times and developed dynamic programming algorithms to maximize the probability of completing the tour by a deadline (Kao 1978, Sniedovich 1981, Carraway et al. 1989). In some papers, authors proposed two-stage stochastic programming methods to formulate the problems and applied branch-and-cut algorithms (Laporte et al. 1992, Kenyon and Morton 2003, Adulyasak and Jaillet 2015). Column generation is another prevalent approach to formulate routing and scheduling problems with stochastic travel and service times. In this approach, the uncertainty of travel and service times are encapsulated in the column definition and are handled in the subproblem (Taş et al. 2014b, Yuan et al. 2015, Errico et al. 2016). In another category of papers, researchers developed chance-constrained programming models for routing and scheduling problems and solved them either optimally or heuristically (Laporte et al. 1992, Li et al. 2010, Zhang et al. 2012, Chen et al. 2014, Miranda and Conceição 2016). In these models, chance constraints ensure that time window constraints or constraints restricting the maximum durations of tours are satisfied probabilistically. Moreover, a large number of works in the literature applies heuristic and metaheuristic algorithms on routing and scheduling problems with stochastic times (Taş et al. 2013, Yan et al. 2014, Gómez et al. 2015, Ehmke et al. 2015, Binart et al. 2016). In addition, some researchers assumed that uncertain travel and service times belong to an uncertainty set and then applied robust optimization methods to find reliable routes. Such robust solutions either satisfy the time windows constraints for all possible realizations of uncertain parameters (Agra et al. 2013, Lee et al. 2012), or minimize a measure index representing the amount of constraint violations for the worst-case scenario (Souyris et al. 2013, Han et al. 2013, Adulyasak and Jaillet 2015, Jaillet et al. 2016, Zhang et al. 2016). We refer interested readers to a recent survey by Oyola et al. (2016) that covers routing with stochastic travel and service times comprehensively.

Time dependency in travel and service times is another factor that significantly influences the difficulty of routing and scheduling problems. In the literature, researchers have considered time dependency mostly for travel times, due to traffic congestions, rather than service times. As suggested by Ichoua et al. (2003), from a modeling perspective, we can categorize papers in this area as follows. In some articles, authors addressed the time dependency by simply considering weighted averages of travel times over different periods and then solved a deterministic VRP (Fisher et al. 1982, Hill et al. 1988, Shen et al. 1995). In another category, authors used piecewise linear functions to represent the travel times (Malandraki and Dial 1996, Ichoua et al. 2003, Hashimoto et al. 2008, Donati et al. 2008, Kuo 2010, Balseiro et al. 2011, Dabia et al. 2013, Jabali et al. 2012, Franceschetti et al. 2013, Taş et al. 2014a). Most articles in this area proposed metaheuristic algorithms as the solution method. Another way to model time dependent problems is based on discretizing the scheduling horizon to smaller intervals with equal lengths that can have different travel and service times (Malandraki and Daskin 1992, Jung and Haghani 2001, Haghani and Jung 2005, Soler et al. 2009, Huang et al. 2017). In these papers, the idea of discretization is essential to formulating problems as mixed integer programming models.

Although there is a rich literature on VRP with synchronized visits and on VRP with stochastic/time dependent travel and service times, to the best of our knowledge, there is no paper addressing these aspects simultaneously. The main contributions of this research are as follows :

- For the first time, we study VRP with synchronized visits and stochastic travel and service times. In addition to considering a home-health care scheduling problem, we introduce an operating room scheduling problem with stochastic durations as a novel application of VRPS. We then formulate the problem as a two-stage stochastic programming model with integer variables in both stages. An advantage of the proposed model is that, unlike the deterministic models in the VRPS literature, our model does not have any big-M constraints for the scheduling part of the problem. This advantage comes at the cost of a large number of second-stage integer variables. We prove that the integrality constraints on second-stage variables are redundant, and therefore we can apply the well-known L-shaped algorithm and its branch-and-cut implementation.

- We considerably enhance the quality of the proposed L-shaped algorithm in the following ways. We develop some valid inequalities for both first- and second-stage models. We also develop a lower bounding functional for the second-stage cost and add it to the master problem of the L-shaped algorithm. Moreover, we propose a solution method for the subproblems of the L-shaped algorithm. This solution method that we specifically design for our problem is much faster than standard linear programming algorithms and is vital for the computational efficiency of the proposed branch-and-cut algorithm.
- We report extensive computational results on the VRP with synchronized visits and stochastic travel and service times. For the home-health care scheduling problem with stochastic travel and service times, we show that our algorithm optimally solves instances with 15 patients and 10% to 30% synchronized visits. It also finds solutions with an average optimality gap of 3.57% for instances with 20 patients. Moreover, we show that the developed algorithm optimally solves stochastic operating room scheduling instances with up to 20 surgeries, a considerable improvement over Batun et al. (2011) that reports on instances with 11 surgeries. In addition, our solution times are significantly lower for instances with 11 surgeries (2231 seconds versus 9992 seconds).
- We also extend the proposed model to solve time-dependent VRP with synchronized visits and show that our proposed algorithm finds optimal solutions for a large portion of time-dependent home-health care scheduling instances with 30 to 60 patients and different rates of synchronized visits.

We organize the remainder of this paper as follows. In Section 5.2, we introduce a VRP with synchronized visits and stochastic travel and service times. We also discuss the applications of this problem in home-health care and operating room scheduling problems. In Sections 5.3 and 5.4, we respectively propose a two-stage stochastic programming model and some valid inequalities to improve it. In Section 5.5, we develop an L-shaped algorithm and present the master problem and subproblems. We develop a lower bounding functional in Section 5.6. In Section 5.7, we propose a solution method for the subproblems of the L-shaped algorithm.

In Section 5.8, we extend the proposed formulation to model time-dependent VRPs with synchronized visits. We provide some implementation details in Section 5.9. We also present extensive computational results on home-health care and operating room scheduling problems in Section 5.10. Finally, we give some concluding remarks in Section 5.11. Proofs of all lemmas and theorems (except Theorem 5.2) are provided in Appendix B.

5.2 Problem definition and applications

In this paper, we consider the following VRP with synchronized visits. Two or more fleets of homogeneous vehicles are available at a depot to serve a number of customers within a day. For serving each customer, a specific set of vehicles of different types must be available at the customer's location. If some vehicles arrive to the customer's location earlier than other required vehicles, they must wait until others arrive before service starts. Vehicles may require consuming some amounts of limited resources while serving customers. The service times of homogeneous vehicles are the same, but they can differ for each vehicle type. When a vehicle's service to a customer finishes, the vehicle either travels to the next customer or finishes its tour by returning to the depot regardless of whether other vehicles are still serving the customer. For each vehicle type, we assume that there is a set of arcs on which traversing is allowed. We suppose that travel and service times are stochastic and a number of scenarios representing the uncertainty is available. Moreover, for each customer there is a time window with a hard earliest start time constraint and a soft latest start time constraint that if violated, some penalties are incurred to the objective function. Furthermore, we suppose that the time available to complete the tours is limited (e.g., 11 hours). There is also a time threshold (e.g., 9 hours) for the start of the overtime period in which drivers are paid in addition to their fixed daily payments. The decision maker must decide on the number of vehicles of different types to hire, routing of vehicles, and departure times of vehicles from the depot. The objective function includes the fixed costs of vehicles, travel costs, waiting costs, overtime costs, and the penalty of delays in serving customers with respect to given latest start times.

In this paper, we consider two applications of the above problem in health care. The first application is a home-health care scheduling problem. In this problem, two types of nurses that we refer to as Registered Nurses (RNs) and Home Health Aides (HHA), must serve patients at their homes. An RN is allowed to provide a wide range of nursing services such as wound dressing, ostomy care, intravenous therapy, administering medication, monitoring the general health of the patient, pain control, and other health support. However, HHAs can help patients with only their basic personal needs such as walking, feeding, and dressing

(Types of Home Health Care Services n.d.). In this problem, we divide patients into three categories ; patients to be cared by an RN, patients to be visited by an HHA and patients that needs to be served simultaneously by an RN and an HHA. Di Mascolo et al. (2014) studied this problem in the absence of uncertainty for travel and service times and proposed a mixed integer programming model to minimize the total waiting costs.

The second application that we study in this work is an operating room scheduling problem with stochastic surgery, anesthesia and cleaning times. In this problem, each surgery is equivalent to a customer in the VRP with synchronized visits explained at the beginning of this section. To perform each surgery, we require two servers (vehicles) to be simultaneously available. These servers are surgeons and operating rooms. We suppose that the number of surgeons is given, but the number of operating rooms, which are identical, is to be determined. We also assume that a unique surgeon is assigned to each surgery and each surgeon has already fixed the sequence of surgeries to operate. We can divide the total time to complete a surgery in an operating room into three parts : 1) preparation and anesthesia part during which the surgeon is not necessarily present in the operating room, 2) the main part of the surgery in which the surgeon operates, and 3) the cleaning part during which the operating room is busy, but the surgeon is not and may have left the room in order to rest or reach her/his next surgery in another operating room. In this problem, for surgeons and operating rooms, the service time to perform a surgery is equal to the duration of the main part of the surgery. Also, for operating rooms as one of the available servers (vehicles), we suppose that the travel time from surgery i to surgery j is equal to the sum of cleaning time after surgery i and the duration of the preparation and anesthesia part of surgery j. The durations of all three parts of surgeries explained above are stochastic.

In order to match the operating room scheduling problem to the VRP with synchronized visits, we assume that servers (operating rooms and surgeons) leave a dummy depot to perform surgeries and return to it at the end of their routes. The operating room scheduling problem previously explained is still slightly different from the VRP with synchronization. For example, surgeons are not homogeneous and surgeries are already assigned to surgeons. To match the operating room scheduling problem with our VRPS, we consider surgeons as homogeneous vehicles and then for this vehicle type, we consider the set of allowed arcs for travelling based on the given sequences of surgeries for surgeons. The decision maker must decide on the number of operating rooms and their routings in order to minimize the total cost that includes fixed costs of operating rooms, waiting costs of surgeons, and overtime costs of operating rooms and surgeons.

Batun et al. (2011) viewed a similar operating room scheduling problem as a scheduling problem rather than a VRP with synchronized visits. They formulated the problem as a two-stage stochastic program and proposed an L-shaped solution algorithm. Although their model is one of the most interesting papers in the field of stochastic operating room scheduling, we observed the following shortcomings of their work :

1. The scheduling part of their model suffers from big-M constraints. It is well-known that models with such constraints are weak due to poor linear programming relaxation. This issue is even worse in this model, because big-M constraints are in the second-stage model, and therefore big-M values appear as coefficients of variables in L-shaped cuts. As a result, the cuts are weak and cannot approximate the second-stage cost effectively.
2. Authors proposed two sets of symmetry-breaking constraints to deal with symmetries in their model. Although these constraints prevent the model from obtaining the same optimal solution with different presentations, they are not very effective in improving the quality of linear programming relaxation and therefore, branch-and-bound algorithms may explore a large number of nodes before these constraints become binding.

We have overcome these issues by developing a model in the next section that is free of big-M constraints and any symmetries.

5.3 Two-stage stochastic programming model

In this section, we propose a two-stage stochastic programming model for the VRPS defined in Section 5.2. In the first-stage model, the decision maker decides about the number of vehicles of different types to hire, routing of vehicles, and their departure times. Then, after the realization of uncertain travel and service times, the second-stage model computes the start times of services to customers. We present the first- and second-stage formulations in the two following sections separately. For some parameters in our model, we have used superscripts that are initials of some keywords.

5.3.1 First-stage model

We use the following notation for sets, parameters and variables in the first-stage model.

Sets:

$\mathcal{R}$:	The set of all types of available vehicles.
$\mathcal{R}_i$:	The set of vehicles required for serving customer i.
$\mathcal{I}$:	The set of all customers.

$\mathcal{I}_r$: The set of customers requiring a type r vehicle. Customers in this set may also need to be visited by vehicles of other types.

$\mathcal{A}_r$: The set of allowed arcs for type r vehicles.

$\mathcal{T}$: The set of available time slots within the scheduling horizon (We suppose that the available time is split into smaller time slots with equal lengths. In the remainder of this paper, wherever we state that an event happens at time slot t, we mean that it occurs at the beginning of the time slot).

$\mathcal{T}_{ri}^{dep}$: The set of time slots at which a type r vehicle may depart the depot to serve customer i with the condition that it can return to the depot before the end of the scheduling horizon. The vehicle may visit some other customers before finishing its tour. This set is inclusive but not necessarily exclusive, i.e., all time slots satisfying the explained condition are members of this set, but there might be some time slots in this set not respecting the condition. This set is not a part of given data. We will discuss at the end of this section how to compute it using the available data.

Parameters:

c_{rij}^t : travel cost for a type r vehicle for travelling from customer i to customer j. Superscript t stands for "*travel*".

c_r^v : The fixed cost of hiring a type r vehicle. Superscript v stands for "*vehicle*".

d_{ri} : The required amount of the resource (if there is any) that is consumed by a type r vehicle while serving customer i.

C_r : The capacity of type r vehicles for the resource (if there is any) that they consume while serving customers. We have $C_r = \infty$ if type r vehicles do not consume any resource for serving customers.

q_{rs} : An lower bound on the minimum number of type r vehicles required for serving customers in set $S \subseteq \mathcal{I}_r$. We compute it by $q_{rs} = max\left\{\left\lceil \sum_{i\in\mathcal{S}} d_{ir}/C_r \right\rceil, 1\right\}$.

Variables:

m_r : The number of type r vehicles to hire.

x_{rij} : 1 if a type r vehicle visits customer j immediately after customer i ; 0 otherwise.

y_{rjt} : 1 if a type r vehicle departs the depot at time slot t to visit customer j ; 0 otherwise.

Based on the given notation, we formulate the first-stage model as follows. In Model (S1), index 0 stands for the depot.

$$\text{(S1)} \qquad \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{m}} \left(\sum_{r\in\mathcal{R}} c_r^v m_r + \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in\mathcal{I}_r\cup\{0\}:\\(i,j)\in\mathcal{A}_r}} c_{rij}^t x_{rij} + Q(\boldsymbol{x},\boldsymbol{y}) \right) \tag{5.1}$$

Subject to :

$$\sum_{i\in\mathcal{I}_r:(0,i)\in\mathcal{A}_r} x_{r0i} = m_r \qquad r\in\mathcal{R} \tag{5.2}$$

$$\sum_{i\in\mathcal{I}_r\cup\{0\}:(i,j)\in\mathcal{A}_r} x_{rij} = 1 \qquad r\in\mathcal{R}, j\in\mathcal{I}_r \tag{5.3}$$

$$\sum_{i\in\mathcal{I}_r\cup\{0\}:(i,j)\in\mathcal{A}_r} x_{rij} = \sum_{i\in\mathcal{I}_r\cup\{0\}:(j,i)\in\mathcal{A}_r} x_{rji} \qquad r\in\mathcal{R}, j\in\mathcal{I}_r\cup\{0\} \tag{5.4}$$

$$\sum_{i,j\in\mathcal{S}:(i,j)\in\mathcal{A}_r} x_{rij} \le |\mathcal{S}| - q_{rs} \qquad r\in\mathcal{R}, \mathcal{S}\subseteq\mathcal{I}_r : |\mathcal{S}|\ge 2 \tag{5.5}$$

$$\sum_{t\in\mathcal{T}_{ri}^{dep}} y_{rjt} = x_{r0j} \qquad r\in\mathcal{R}, j\in\mathcal{I}_r \tag{5.6}$$

$$x_{rij}\in\{0,1\} \qquad r\in\mathcal{R}, (i,j)\in(\mathcal{I}_r\cup\{0\}) : (i,j)\in\mathcal{A}_r \tag{5.7}$$

$$y_{rjt}\in\{0,1\} \qquad r\in\mathcal{R}, j\in\mathcal{I}_r, t\in\mathcal{T}_{rj}^{dep} \tag{5.8}$$

$$m_r\ge 0, \text{integer} \qquad r\in\mathcal{R} \tag{5.9}$$

Objective function (5.1) consists of the fixed costs of vehicles, the travel costs, and the second-stage cost $Q(\boldsymbol{x},\boldsymbol{y})$ that is a function of first-stage decision variables x_{rij} and y_{rjt}. We define the second-stage cost in Section 5.3.2. Constraints (5.2) and (5.3) are degree constraints for the depot and customers respectively. Constraint (5.4) is the flow conservation constraint. Constraint (5.5) guaranties that for each type of vehicles, the capacity constraints are respected and no subtour is allowed. Constraint (5.6) determines the departure time of vehicles from the depot. Constraints (5.7)-(5.9) represent the integrality constraints for first-stage decision variables. In Model (S1), we point out that constraint (5.5) eliminates subtours composed of arcs traversed by vehicles of the same type for different types of vehicles separately, but not together. We illustrate it by an example depicted in Figure 5.1. In this figure, there is no subtour for type 1 and type 2 vehicles separately, but path A-B-C-D formed by both types of vehicles is a subtour. In our problem, this type of subtour is also illegal and results in infeasibility in the scheduling of customers visits. The infeasibility happens because the start times of services to any pair of customers (i_1, i_2) in the subtour must satisfy $start_{i_1} < start_{i_2}$ and $start_{i_1} > start_{i_2}$ that is a contradiction. Therefore, although infeasibility cuts from the subproblems of L-shaped algorithm will remove such subtours, we should avoid them beforehand in the first-stage model. As discussed later, we address this issue by adding valid inequalities and also a lower bounding functional developed in Sections 5.4.1 and 5.6 respectively.

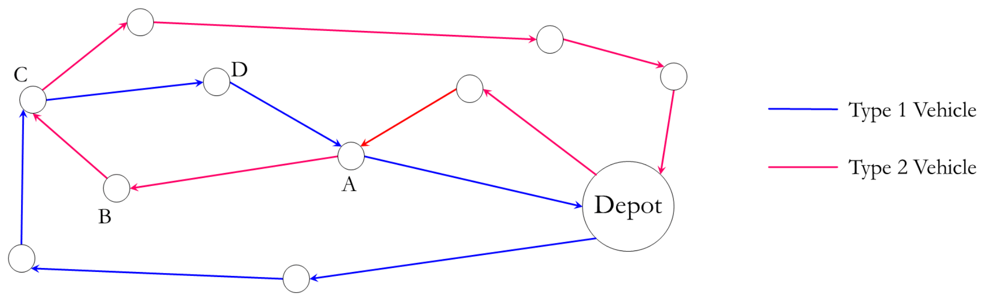

Figure 5.1 A subtour formed by two different types of vehicles.

5.3.2 Second-stage model

To formulate the second-stage model, we use the following variables, sets, and parameters.

Variables:

$u_{rijt\omega}$: 1 if a type r vehicle starts serving customer i at time slot t immediately before serving customer j in scenario ω ; 0 otherwise.

$v_{it\omega}$: 1 if the service to customer i starts at time slot t in scenario ω ; 0 otherwise.

$w_{r\omega}$: The total waiting time of all type r vehicles in scenario ω.

Parameters:

L : The length of the normal working hours for which no overtime cost is considered. In our model, we consider this parameter in terms of time slots length. We also note that $L \leq |\mathcal{T}|$ holds as set $\mathcal{T}$ defined in Section 3.1 includes some additional time slots for overtime periods.

e_i : The earliest start time in the time window of customer i.

l_i : The latest start time in the time window of customer i.

$s_{ri\omega}$: The duration of the service provided by a type r vehicle to customer i in scenario ω.

$t_{rij\omega}$: The travel time of a type r vehicle from customer i to a customer j in scenario ω.

$c^o_{rit\omega}$: The overtime cost of a type r vehicle if it starts serving customer i at time slot t in scenario ω and then immediately returns to the depot. Superscript o stands for "*overtime*". We compute it by $c^o_{rit\omega} = c'_{overtime} \times max\{t + s_{ri\omega} + t_{ri0\omega} - L, 0\}$ where $c'_{overtime}$ is the overtime cost for a single time slot beyond the session length L.

c^d_{it} : The delay cost of serving customer i when the service to the customer starts at time slot t. Superscript d stands for "*delay*". We compute it by $c^d_{it} = c'_{delay} \times max\{t - l_i, 0\}$ where c'_{delay} is the delay cost for a single time slot beyond the latest start time l_i.

c_r^w : The waiting cost of a type r vehicle for a single time slot. Superscript w stands for "*waiting*".

$f_{rit\omega}$: The time slot at which a type r vehicle arrives in the depot immediately after starting to serve customer i at time slot t in scenario ω. We have $f_{rit\omega} = t + s_{ri\omega} + t_{ri0\omega}$.

$g_{rijt\omega}$: The amount of time in scenario ω that a type r vehicle is involved in serving customer i and also in travelling to the next customer j. We have by $g_{rijt\omega} = s_{ri\omega} + t_{rij\omega}$.

$\xi(\omega)$: The vector of uncertain parameters including travel and service times in scenario ω.

Sets:

$\mathcal{T}_{i\omega}$: The set of time slots at which the service to customer i may start in scenario ω. This set is inclusive but not necessarily exclusive (Later in this section, we discuss how to compute this set).

$T_{rij\omega}$: The set of times slots at which a type r vehicle may start serving customer i immediately before customer j in scenario ω. This set is inclusive, but not necessarily exclusive (Later in this section, we discuss how to compute this set).

Ω : The set of random scenarios.

An assumption in modeling the second stage is that travel and service times are multiples of time slot length. This assumption may require small modifications in real data. However, we believe that the effect of this modification is negligible since the length of time slots that we consider in our applications is very small (5 minutes) compared to the length of the scheduling horizon (11 hours). In the previous notation, we suppose that $s_{ri\omega}$, $t_{rij\omega}$, and also all elements of $\xi(\omega)$ are given in terms of time slot length. Using the given notation, we formulate the second-stage model for scenario $\omega \in \Omega$ as follows.

$$\text{(S2)} \qquad Q(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega)) = \min_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}} \left(\sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ (i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0t\omega}} c^o_{rit\omega} u_{ri0t\omega} + \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}_{i\omega}} c^d_{it} v_{it\omega} + \sum_{r\in\mathcal{R}} c^w_r w_{r\omega} \right) \tag{5.10}$$

Subject to :

$$\sum_{t\in\mathcal{T}_{rij\omega}} u_{rijt\omega} = x_{rij} \qquad r\in\mathcal{R}, i,j \in (\mathcal{I}_r \cup \{0\}) \colon (i,j)\in\mathcal{A}_r \tag{5.11}$$

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\ t'+s_{ri\omega}+t_{rij\omega}\le t}} u_{rijt'\omega} \ge \sum_{\substack{k\in(\mathcal{I}_r\cup\{0\}):\\ (j,k)\in\mathcal{A}_r \,\&\, t\in\mathcal{T}_{rjk\omega}}} u_{rjkt\omega} \quad r\in\mathcal{R}, j\in\mathcal{I}_r, t\in\mathcal{T}_{j\omega} \tag{5.12}$$

$$\sum_{\substack{j\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r \,\&\, t'\in\mathcal{T}_{rij\omega}}} u_{rijt\omega} = v_{it\omega} \qquad r\in\mathcal{R}, i\in\mathcal{I}_r, t\in\mathcal{T}_{i\omega} \tag{5.13}$$

$$w_{r\omega} = \sum_{\substack{i\in\mathcal{I}_r:\\(i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0\omega}} f_{rij\omega}u_{ri0t\omega} - \sum_{\substack{i,j\in\mathcal{I}_r\cup\{0\}:\\(i,j)\in\mathcal{A}_r}} g_{rij\omega}x_{rij} - \sum_{i\in\mathcal{I}_r}\sum_{t\in\mathcal{T}^{dep}_{ri}} ty_{rit} \quad r\in\mathcal{R} \tag{5.14}$$

$$u_{r0it\omega} = y_{rit} \qquad r\in\mathcal{R}, i\in\mathcal{I}_r : (0,i)\in\mathcal{A}_r, t\in\mathcal{T}^{dep}_{ri} \tag{5.15}$$

$$u_{rijt\omega} \in \{0,1\} \qquad r\in\mathcal{R}, i,j\in(\mathcal{I}_r\cup\{0\}) : (i,j)\in\mathcal{A}_r, t\in\mathcal{T}_{rij\omega} \tag{5.16}$$

$$v_{it\omega} \in \{0,1\} \qquad i\in\mathcal{I}, t\in\mathcal{T}_{i\omega} \tag{5.17}$$

Objective function (5.10) represents the second-stage cost in scenario ω and includes overtime, delay, and waiting costs. The second-stage cost $Q(\boldsymbol{x},\boldsymbol{y})$ in objective function (5.1) is calculated by $Q(\boldsymbol{x},\boldsymbol{y}) = E_{\omega\in\Omega}[Q(\boldsymbol{x},\boldsymbol{y},\xi(\omega))]$ where $E_{\omega\in\Omega}[.]$ computes the expected value over scenarios $\omega\in\Omega$. Constraint (5.11) links first- and second-stage decision variables x_{rij} and $u_{rijt\omega}$. Constraint (5.12) is the no-overlap constraint and indicates that if the service to customer j starts at time slot t, then the service to customer i visited immediately before customer j by the same vehicle must have started by time slot $t - s_{ri\omega} - t_{rij\omega}$. Constraint (5.13) is the synchronization constraint and ensures that required vehicles start serving the customer at the same time. Constraint (5.14) computes total waiting times for different types of vehicles. In this constraint, $\sum_{i\in\mathcal{I}_r:(i,0)\in\mathcal{A}_r}\sum_{t\in\mathcal{T}_{ri0\omega}} f_{rij\omega}u_{ri0t\omega}$ computes the total tour completion times, $\sum_{i,j\in\mathcal{I}_r\cup\{0\}:(i,j)\in\mathcal{A}_r} g_{rij\omega}x_{rij}$ computes the sum of total service and travel times, and $\sum_{i\in\mathcal{I}_r}\sum_{t\in\mathcal{T}^{dep}_{ri}} ty_{rit}$ calculates the total departure times. We used auxiliary variables $w_{r\omega}$ for the ease of presenting objective function (5.10). We can simply substitute $w_{r\omega}$ in objective function (5.10). In this case, it would be better to transfer the expected value of the second and the third terms on the right-hand side of constraint (5.14) to objective function (5.1) in the first-stage model. Constraint (5.15) introduces the departure times from the depot to the second-stage model. Constraints (5.16) and (5.17) represent integrality constraints for second-stage variables.

In the following we aim to explain how to compute sets $\mathcal{T}_{i\omega}$ and $\mathcal{T}_{rij\omega}$, but before that we need introduce two new parameter $SRT_{rijt\omega}$ and $n\text{-}SRT_{rijt\omega}$ by which we define these sets.

$SRT_{rijt\omega}$: The shortest amount of time from the beginning of time slot t that a type r vehicle starts serving customer i until it reaches customer j in scenario ω. By reaching customer j we mean that the vehicle arrives the customer's location and if needed it waits until the customer's time window opens. As we suppose that the triangle inequality does not necessarily hold, the vehicle may visit some other customers between customers i and j.

$n\text{-}SRT_{rijt\omega}$: It is defined the same as $SRT_{rijt\omega}$ with a additional condition that the path from customer i to customer j includes at most n arcs.

Algorithm 5.1 provides the pseudo code of the algorithm that we propose to compute $SRT_{rijt\omega}$ for a fixed customer i, a fixed time slot t and all customers $j \in \mathcal{I}\backslash\{i\}$. This algorithm is the extension of Bellman-Ford algorithm where earliest start time constraints and service times are also considered. The following lemmas and Theorem 5.1 proves the correctness of Algorithm 5.1.

LEMMA 5.1 *In Algorithm 5.1, if $d_j < \infty$, it is equal to the amount of time that it takes for a type r vehicle departing customer i at time slot t, until it reaches customer j and completes its service. The vehicle may visit and serve some other customers between customers i and j.*

PROOF. Appendix B.1. This lemma is used in the proof of Theorem 5.1. □

Algorithm 5.1. Shortest path algorithm with the earliest start time constraints and service times

```
1: Set $d_i = 0$ and $d_j = \infty$ for $j \in \mathcal{I}\backslash\{i\}$
2: for ($n$=1 to $|\mathcal{I}| - 1$) do
3:     for (each edge $(j,k)$ in the set of arcs) do
4:         if ($d_k > max\{d_j + t_{rjk\omega}, e_k - t\} + s_{rk\omega}$) then
5:             $d_k = max\{d_j + t_{rjk\omega}, e_k - t\} + s_{rk\omega})$
6:         end if
7:     end for
8: end for
9: Set $SRT_{rijt\omega} = d_j - s_{rj\omega}$ for $j \in \mathcal{I}\backslash\{i\}$
```

LEMMA 5.2 *After performing the n-th iteration in the main loop of Algorithm 5.1, d_j is equal to $n\text{-}SRT_{rijt\omega}$.*

PROOF. Appendix B.2. This lemma is used in the proof of Theorem 5.1. □

THEOREM 5.1 *Algorithm 5.1 computes correct values of $SRT_{rijt\omega}$.*

PROOF. Appendix B.3. □

In order to calculate sets $\mathcal{T}_{rij\omega}$, $\mathcal{T}_{i\omega}$ and $\mathcal{T}_{ri}^{dep}$, we also need to define a new time set $\mathcal{T}'_{rij\omega}$ as follows :

$\mathcal{T}'_{rij\omega}$: This time set is defined the same as $\mathcal{T}_{rij\omega}$. The only difference is that $\mathcal{T}_{rij\omega}$ is a refined version of $\mathcal{T}'_{rij\omega}$ and we have $\mathcal{T}_{rij\omega} \subseteq \mathcal{T}'_{rij\omega}$. We compute $\mathcal{T}'_{rij\omega}$ by $\mathcal{T}'_{rij\omega} = \{t | t \geq max\{e_i, SRT_{r0i0\omega}\}$ & $max\{e_j, t + s_{ri\omega} + t_{rij\omega}\} + s_{rj\omega} + SRT_{rj0t^*\omega} \leq |\mathcal{T}|\}$ where $t^* = max\{e_j, t + s_{ri\omega} + t_{rij\omega}\} + s_{rj\omega}$.

Based on $\mathcal{T}'_{rij\omega}$ and $SRT_{rijt\omega}$ we compute sets $\mathcal{T}_{rij\omega}$, $\mathcal{T}_{i\omega}$ and $\mathcal{T}^{dep}_{ri}$ as follows.

$$\mathcal{T}_{i\omega} = \bigcap_{r \in \mathcal{R}_i} \left[\bigcup_{j \in (\mathcal{I}_r \cup \{0\} \setminus i} \mathcal{T}'_{rij\omega} \right] \tag{5.18}$$

$$\mathcal{T}_{rij\omega} = \mathcal{T}_{i\omega} \cap \mathcal{T}'_{rij\omega} \tag{5.19}$$

$$\mathcal{T}^{dep}_{ri} = \{t \in \mathcal{T} | t + \max_{\omega \in \Omega} \{t_{r0i\omega} + s_{ri\omega} + SRT_{ri0(t + t_{r0it\omega} + s_{ri\omega})\omega}\} \leq |\mathcal{T}|\} \tag{5.20}$$

5.4 Valid inequalities

In this section, we develop some valid inequalities for the first- and the second-stage models. We add the first two valid inequalities to the first-stage model, while the third class of valid inequalities are for the second-stage model.

5.4.1 Subtour-elimination constraints

As illustrated by Figure 5.1, constraint (5) does not eliminate subtours formed by arcs traversed by vehicles of different types. While lower bounding functional introduced in Section 6 and also infeasibility cuts generated from the subproblem of the L-shaped algorithm remove these subtours, we prefer to avoid them more efficiently through adding valid inequalities (5.21)-(5.24) to the first-stage model.

New variables:

z_{ij} : 1 if any vehicle serves customer j immediately after customer i ; 0 otherwise.

$$z_{ij} \geq x_{rij} \qquad r \in \mathcal{R}, i, j \in (\mathcal{I}_r \cup \{0\}) : (i, j) \in \mathcal{A}_r \tag{5.21}$$

$$z_{ij} \leq \sum_{r \in \mathcal{R} : (i,j) \in \mathcal{A}_r} x_{rij} \qquad i, j \in (\mathcal{I}_r \cup \{0\}) : (i, j) \in \bigcap_{r \in R_i} \mathcal{A}_r \tag{5.22}$$

$$\sum_{i,j \in \mathcal{S} : (i,j) \in \bigcap_{r \in R_i} \mathcal{A}_r} z_{ij} \leq |\mathcal{S}| - 1 \qquad \mathcal{S} \subseteq \mathcal{I} : |\mathcal{S}| \geq 2 \tag{5.23}$$

$$0 \leq z_{ij} \leq 1 \qquad i, j \in (\mathcal{I}_r \cup \{0\}) : (i, j) \in \bigcap_{r \in R_i} \mathcal{A}_r \tag{5.24}$$

Constraints (5.21)-(5.22) indicate that for a fixed arc (i, j), z_{ij} takes 1 if at least one of the variables x_{rij} is equal to 1 and it takes 0 if all variables x_{rij} are equal to 0. Constraint (5.23) is the subtour-elimination constraint defined on z_{ij} variables. As x_{rij} variables are binary, we do not need to consider integrality constraints for z_{ij} variables.

5.4.2 Capacity constraints for service times

As discussed for the first-stage model, if vehicles consume a limited resource while serving customers, constraint (5.5) is essential to ensure that the capacity constraints are satisfied. In healthcare applications explained in Section 5.2, there is not any physical resource required while serving customers. However, in routing and scheduling problems, we can consider "time" as a resource and impose constraint (5.5) to guarantee that the total service time in each tour does not exceed the maximum available time in the scheduling horizon. Therefore, we can add the following valid inequality to the first-stage model.

$$\sum_{i,j\in\mathcal{S}:(i,j)\in\mathcal{A}_r} x_{rij} \leq |\mathcal{S}| - \left\lceil \sum_{i\in\mathcal{S}} s_{ri\omega}/|\mathcal{T}| \right\rceil \qquad r\in\mathcal{R},\omega\in\Omega,\mathcal{S}\subseteq\mathcal{I}:|\mathcal{S}|\geq 2 \tag{5.25}$$

The above cut is known as the rounded capacity inequality in the literature (Lysgaard et al. 2004). We also add some other valid inequalities by considering "time" as a consumable resource while serving customers. These constraints are Framed capacity, Strengthened comb, Homogeneous multistar and Hypotour inequalities. We refer readers for more information about these valid inequalities to Lysgaard et al. (2004).

5.4.3 Improved no-overlap constraints

The following theorem shows how we can improve constraint (5.12).

THEOREM 5.2 *Constraints (5.26)-(5.27) are valid inequalities for the second-stage model.*

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\t'+s_{ri\omega}+t_{rij\omega}\leq t}} u_{rijt'\omega} \geq \sum_{\substack{k\in(\mathcal{I}_r\cup\{0\}):\\(j,k)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rjk\omega}:\\t'\leq t}} u_{rjkt'\omega} \qquad r\in\mathcal{R},j\in\mathcal{I}_r,t\in\mathcal{T}_{j\omega} \tag{5.26}$$

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):(i,j)\in\mathcal{A}_r\\ \&\,(t-s_{ri\omega}-t_{rij\omega})\in\mathcal{T}_{rij\omega}}} u_{rij(t-s_{ri\omega}-t_{rij\omega})\omega} = \sum_{\substack{k\in(\mathcal{I}_r\cup\{0\}):\\(j,k)\in\mathcal{A}_r\,\&\,t\in\mathcal{T}_{rjk\omega}}} u_{rjkt\omega} \qquad \begin{array}{l} r\in\mathcal{R},j\in\mathcal{I}_r:|\mathcal{R}_j|=1\\ t\in\mathcal{T}_{j\omega}:t>e_j\end{array} \tag{5.27}$$

PROOF. The validity of constraint (5.26) originates from the definition of variables $u_{rijt\omega}$. We have obtained constraints (5.26) by lifting the right-hand side of constraint (5.12).

Constraint (5.26) indicates that if service to customer j starts at time t or earlier, then the vehicle must have arrived to this customer at time t or earlier. Constraint (5.27) implies that for any customer j requiring a single vehicle (condition $\mathcal{R}_j = \{r\}$ in (5.27)), the service starts as soon as the vehicle arrives to the customer if the arrival time is after the earliest start time of the corresponding time window (condition $t \in \mathcal{T}_{j\omega} : t > e_j$ in (5.27)). Constraint (5.27) is valid because, as stated later by Lemma 5.3, there is no advantage to postpone the service to a customer when all required vehicles are available at the customer's location. □

THEOREM 5.3 *Clique inequalities (5.28)-(5.29) are equivalent to constraints (5.26)-(5.27).*

$$\sum_{\substack{k\in(\mathcal{I}_r\cup\{0\}):\\ (j,k)\in\mathcal{A}_r \,\&\, t'\in\mathcal{T}_{rjk\omega}}} \sum_{\substack{t'\in\mathcal{T}_{rjk\omega}:\\ t'\leq t}} u_{rjkt\omega} + \sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\ t'+s_{ri\omega}+t_{rij\omega}>t}} u_{rijt'\omega} \leq 1 \quad r\in\mathcal{R}, j\in\mathcal{I}_r, t\in\mathcal{T}_{j\omega} \tag{5.28}$$

$$\sum_{\substack{k\in(\mathcal{I}_r\cup\{0\}):\\ (j,k)\in\mathcal{A}_r \,\&\, t\in\mathcal{T}_{rjk\omega}}} u_{rjkt\omega} + \sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\ t'+s_{ri\omega}+t_{rij\omega}\neq t}} u_{rijt'\omega} = 1 \quad \begin{array}{l} r\in\mathcal{R}, j\in\mathcal{I}_r : \\ |\mathcal{R}_j| = 1, t\in\mathcal{T}_{j\omega} : t > e_j \end{array} \tag{5.29}$$

PROOF. Appendix B.4. □

Although constraints (5.28)-(5.29) and constraints (5.26)-(5.27) are equivalent, there is a significant benefit in considering the former constraints. Mixed integer programming (MIP) solvers usually create a clique graph where each node represents a binary variable and an edge between two nodes shows that the two corresponding variables do not take the value of one simultaneously in any feasible solution. While solving a model, MIP solvers dynamically updates this graph and detects more clique inequalities by reasoning on this graph. In the branch-and-bound tree, whenever the solver finds a fractional solution violating any clique inequality, it adds the violated cut to the model. As discussed later in Section 6, as the lower bounding functional, we add a copy of second-stage constraints for the average scenario to the master problems of the proposed L-shaped algorithm. For the lower bounding functional, by introducing constraints (5.28)-(5.29) rather than constraints (5.26)-(5.27), we enrich the clique graph and let the MIP solver detect more clique cuts and add them to the model when they are violated.

5.5 L-shaped algorithm

The L-shaped algorithm is applicable to stochastic programming models with continuous recourse decision variables. However, the second-stage variables of the model that we developed in Section 5.3 are integer. The following theorem addresses this issue.

THEOREM 5.4 *Integrality constraints (5.16)-(5.17) on second-stage variables are trivial assuming that subtours are prevented using constraints (5.21)-(5.24) or the lower bounding functional later presented in Section 5.6.*

PROOF. Appendix B.5. □

As a result of Theorem 5.4, we can relax the integrality constraints of all second-stage variables and apply the L-shaped algorithm rather than the integer L-shaped algorithm. The master problem (MP) of the L-shaped algorithm reads as follows.

$$\text{(MP)} \quad \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{\theta}} \left[\sum_{r\in\mathcal{R}} c^v_r m_r + \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in\mathcal{I}_r\cup\{0\}:\\(i,j)\in\mathcal{A}_r}} c^t_{rij} x_{rij} - \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} c^w_r \left(\sum_{\omega\in\Omega} p_\omega g_{rijt\omega} \right) x_{rij} + \right.$$

$$\left. - \sum_{r\in\mathcal{R}} \sum_{i\in\mathcal{I}_r} \sum_{t\in\mathcal{T}^{dep}_{ri}} c^w_r t y_{rit} + \theta \right] \tag{5.30}$$

Subject to :

$$(5.2) - (5.9) \tag{5.31}$$

$$\theta \geq \sum_{\omega\in\Omega} p_\omega \theta_\omega \tag{5.32}$$

$$\theta_\omega \geq \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in\mathcal{I}_r\cup\{0\}:\\(i,j)\in\mathcal{A}_r}} \pi^{(1)}_{crij\omega} x_{rij} - \sum_{r\in\mathcal{R}} \sum_{i\in\mathcal{I}_r:(i,0)\in\mathcal{A}_r} \sum_{t\in\mathcal{T}^{dep}_{ri}} \pi^{(2)}_{crit\omega} y_{rit} \qquad \omega\in\Omega, c\in\mathcal{C}^o_\omega \tag{5.33}$$

$$\sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in\mathcal{I}_r\cup\{0\}:\\(i,j)\in\mathcal{A}_r}} \sigma^{(1)}_{crij\omega} x_{rij} - \sum_{r\in\mathcal{R}} \sum_{i\in\mathcal{I}_r:(i,0)\in\mathcal{A}_r} \sum_{t\in\mathcal{T}^{dep}_{ri}} \sigma^{(2)}_{crit\omega} y_{rit} \leq 0 \qquad \omega\in\Omega, c\in\mathcal{C}^f_\omega \tag{5.34}$$

We also formulate subproblem (SP$_\omega$) for scenarios $\omega\in\Omega$ in the L-shaped algorithm as follows.

$$\text{(SP}_\omega\text{)} \qquad Q'(\boldsymbol{x},\boldsymbol{y},\xi(\omega)) = \min_{u,v} \left(\sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\(i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0t\omega}} c^o_{rit\omega} u_{ri0t\omega} + \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}_{i\omega}} c^d_{it} v_{it\omega} + \right.$$

$$\left. \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\(i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0t\omega}} c^w_r f_{rit\omega} u_{ri0t\omega} \right) \tag{5.35}$$

Subject to :

$$(5.11), (5.13), (5.15), (5.26) \tag{5.36}$$

$$0 \leq u_{rijt\omega} \leq 1 \qquad r\in\mathcal{R}, i,j\in(\mathcal{I}_r\cup\{0\}):(i,j)\in\mathcal{A}_r, t\in\mathcal{T}_{rij\omega} \tag{5.37}$$

$$0 \leq v_{it\omega} \leq 1 \qquad i \in \mathcal{I}, t \in \mathcal{T}_{i\omega} \tag{5.38}$$

In objective function (5.30), the third and the fourth terms are the negative parts of the waiting cost in the second-stage model (S2). We have assumed that in the second-stage model (S2), $w_{r\omega}$ in objective function (5.10) is substituted using constraint (5.14) and then the negative parts of $w_{r\omega}$ are transferred to the objective function of first-stage model (S1). In the third term of (5.30), p_ω represents the probability of scenario ω. Also, θ is the approximation of the second-stage cost without the negative parts of the waiting cost. In constraint (5.32), θ_ω stands for the approximation of the second-stage cost in scenario ω without the negative parts of the waiting cost. Constraints (5.33), and (5.34) are the optimality and feasibility cuts that are iteratively generated after solving subproblems (SP$_\omega$). In these constraints, $\mathcal{C}^o_\omega$ and $\mathcal{C}^f_\omega$ are the index set of generated optimality and feasibility cuts for scenario ω. In constraint (5.33), $\pi^{(1)}_{crij\omega}$ and $\pi^{(2)}_{crit\omega}$ are the dual variables of constraints (5.11) and (5.15) when generating the c-th optimality cut, respectively. Likewise, in constraint (5.34), $\sigma^{(1)}_{crij\omega}$ and $\sigma^{(2)}_{crit\omega}$ are the extreme rays of constraints (5.11) and (5.15) when generating the c-th feasibility cut. In the objective function (5.35), the third term is obtained after substituting $w_{r\omega}$ in objective function (5.10) using constraint (5.14).

In the L-shaped algorithm, after solving the master problem (MP), we fix the first-stage solution in subproblems and solve them for all scenarios $\omega \in \Omega$. We then generate optimality and feasibility cuts for subproblems solved optimally and those ones detected to be infeasible respectively. The algorithm iterates until it reaches the maximum acceptable optimality gap or the time limit. In addition to the standard L-shaped algorithm, we also implement the branch-and-cut version of the L-shaped algorithm, where we solve the master problem once by a branch-and-cut algorithm. In the branch-and-bound tree, whenever we find a first-stage feasible solution we solve subproblems and add optimality and feasibility cuts (5.33) and (5.34) to the branch-and-bound tree.

A question that may arise about the subproblem (SP$_\omega$) is that why we considered constraint (5.26) rather than its clique version (5.28). The reason is that, as stated by Theorem 5.3, these two constraints are mathematically equivalent. In addition, as discussed in Section 5.4.3, constraint (5.28) is more effective than constraint (5.26) only when we add it as a part of the lower bounding functional to the master problem. In the subproblems of the L-shaped algorithm, although these constraints are equivalent, we prefer constraint (5.26). This is because there is a non-zero constant on the right-hand side of constraint (5.28) and therefore if we include constraint (5.28) in subproblems, we must add an extra dual value of this constraint to the optimality and feasibility cuts. This additional dual value makes the analysis of the subproblem discussed in Section 5.7 more complicated without providing any benefit. On the contrary, we do not need to add any dual value corresponding to constraint (5.26) to optimality and feasibility cuts since constant value of this constraint is zero.

One may wonder why we ignored constraints (5.27) and (5.29) in subproblems. This is because we solve the subproblems only for first-stage integer solutions. As constraints (5.27) and (5.29) are trivial in this case, we simply neglect them in subproblems. However, we note that these constraints are beneficial when added to the master problem as the lower bounding functional.

5.6 Lower bounding functional

In the following, we aim to develop a lower bounding functional for the proposed L-shaped algorithm.

Lemma 5.3 *For a fixed first-stage solution, in any scenario $\omega \in \Omega$, services to customers start as soon as all required vehicles are available at the customers' locations.*

Proof. Appendix B.6. This lemma is used in the proof of Lemma 5.5. □

Lemma 5.4 *For a fixed first-stage solution, if customers are served as soon as the required vehicles are available at customers' locations, the finish time of service to each customer is convex in terms of $\xi(\omega)$.*

Proof. Appendix B.7. This lemma is used in the proof of Lemma 5.5. □

As discussed in Section 5.3.2, $Q(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ that we formulated by relations (5.10)-(5.17) computes the second-stage cost for a fixed first-stage solution $(\boldsymbol{x}, \boldsymbol{y})$ in scenario ω assuming that service and travel times in this scenario are multiples of the time slot length. Similarly, $Q'(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$, defined by (5.35)-(5.38), calculates the second-stage cost without negative parts of the waiting cost for scenario ω if all travel and service times are multiples of the time slot length. Let's define $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ as a function that computes the second-stage cost without negative parts of the waiting cost for a fixed first-stage solution $(\boldsymbol{x}, \boldsymbol{y})$ in scenario ω without any condition on service and travel times, i.e., these times are not necessarily multiples of the time slot length.

Lemma 5.5 *For a fixed first-stage solution $(\boldsymbol{x}, \boldsymbol{y})$, $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ is convex in $\xi(\omega)$.*

PROOF. Appendix B.8. This lemma is used in the proof of Theorem 5.5. $\square$

LEMMA 5.6 *$Q'(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\omega)\rfloor) \leq Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ holds where $\lfloor\xi(\omega)\rfloor$ denotes a vector of rounded-down travel and service times for scenario ω.*

PROOF. Appendix B.9. This lemma is used in the proof of Theorem 5.5. $\square$

THEOREM 5.5 *$\theta \geq Q'(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\bar{\omega})\rfloor)$ is a valid inequality for master problem (MP) where $\xi(\bar{\omega})$ is the vector of travel and service times for the average scenario (i.e., $\xi(\bar{\omega}) = \sum_{\omega\in\Omega} p_\omega \xi(\omega)$).*

PROOF. Appendix B.10. The proof of Theorem 5.5 is based on Lemmas 5.5 and 5.6, and also Jensen's Inequality (Jensen 1906). $\square$

Theorem 5.5 shows that we can add $\theta \geq Q'(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\bar{\omega})\rfloor)$ as a lower bounding functional to the master problem (MP). To consider this lower bounding functional in our model, we should impose the following constraints to the master problem.

$$\theta \geq \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ (i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0t\bar{\omega}}} c^o_{rit\bar{\omega}} u_{ri0t\bar{\omega}} + \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}_{i\bar{\omega}}} c^d_{it} v_{it\bar{\omega}} + \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ (i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0t\bar{\omega}}} c^w_r f_{rit\bar{\omega}} u_{ri0t\bar{\omega}} \tag{5.39}$$

$$(5.11), (5.13), (5.15), (5.26), (5.37)\text{-}(5.38) \text{ for scenario } \bar{\omega} \tag{5.40}$$

We note that in constraints (5.39)-(5.40), $\bar{\omega}$ denotes the scenario corresponding to the realization $\lfloor\xi(\bar{\omega})\rfloor$. As shown above, we make a copy of the second-stage variables for scenario $\bar{\omega}$ and add the corresponding second-stage constraints and the objective function by (5.39)-(5.40). The above lower bounding functional is very effective and is a vital part of the L-shaped algorithm developed in this paper. This is because it provides a strong lower bound for approximating the second-stage cost. There are some points that can improve the proposed lower bounding functional. First, as stated by Theorem 5.4, integrality constraints on the second-stage variables are trivial and we can relax them. However, we noticed that CPLEX finds feasible solutions more easily when we declare these variables as integer variables. This is perhaps because CPLEX applies some heuristics to find feasible solutions that are more effective in the case of integer variables. Therefore, in the lower bounding functional (5.39)-(5.40), we replace constraint (5.37)-(5.38) by constraint (5.16)-(5.17) and consider variables $v_{it\bar{\omega}}$ and $u_{rijt\bar{\omega}}$ as binary variables. Second, as discussed in Section 5.4.3, constraint (5.28) is more effective than constraint (5.26) when we have integrality constraint on $v_{it\bar{\omega}}$ and $u_{rijt\bar{\omega}}$ variables.

Therefore, in the lower bounding functional, we replace constraint (5.26) by constraint (5.28) in (5.39). Moreover, we improve the lower bounding functional by adding constraint (5.29) for scenario $\bar{\omega}$.

The proposed lower bounding functional is inspired by Batun et al. (2011) where authors applied Jensen's Inequality to devise a lower bounding functional for the second-stage cost of a stochastic operating room scheduling problem. In this work, authors proved the convexity of the second-stage cost function based on the fact that their second-stage model is a linear programming model with linear uncertain parameters. However, we could not follow the same way as uncertain parameters appear in the definition of time sets in our second-stage model. Thus, we proved the convexity of the second-stage model by analyzing the structure of the objective function. Another difference between our lower bounding method with that of Batun et al. (2011) is that we use the realization $\lfloor \xi(\bar{\omega}) \rfloor$ rather than $\xi(\bar{\omega})$ because our second-stage formulation works in the case that travel and service times are multiples of time slots length. An advantage of our lower bounding functional to the one presented by Batun et al. (2011), is that ours does not have any big-M constraints and thus we expect it to provide a tighter approximation of the second-stage cost.

THEOREM 5.6 *The proposed lower bounding functional eliminates subtours.*

PROOF. Appendix B.11. $\square$

Although Theorem 5.6 guaranties that the lower bounding functional removes subtours from the first-stage model, we include the subtour-elimination constraints (5.21)-(5.24) in the first-stage model, because they are stronger. Grötschel and Padberg (1979) proved that similar subtour-elimination constraints in the symmetric travelling salesman problem define facets of the solution space polytope.

5.7 Analysis of subproblems

As discussed in the previous section, to generate the optimality and feasibility cuts, we need to solve subproblems for all scenarios $\omega \in \Omega$ and extract dual values or infeasibility extreme rays. This step of the algorithm is computationally demanding especially for our subproblems that include a large number of variables and constraints. This issue is even worse in the case of the branch-and-cut implementation of the L-shaped algorithm where we must solve subproblems whenever we find a first-stage feasible solution in any node of the branch-and-bound tree. In this section, we develop a solution method for subproblems that is much faster than standard linear programming algorithms. We have designed this solution method specifically for the subproblems of our problem. In the following, we provide Algorithm 5.2 that, for a fixed first-stage solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$, computes the start times of services to customers in scenario ω. Using this algorithm, if we find that the completion times of all tours are within the scheduling horizon $|\mathcal{T}|$, the subproblem is feasible and we use the formula presented in Section 5.7.1 to compute the dual values $\pi^{(1)}_{crij\omega}$ and $\pi^{(2)}_{crit\omega}$. We then generate an optimality cut (5.33) based on the calculated dual values. However, if we realize that any vehicle completes its tour after the end of the scheduling horizon, we generate an infeasibility cut as explained in Section 5.7.2.

In Algorithm 5.2, L_2 is the set of customers with a fixed start time for the service and L_1 maintains the set of customers that do not have any preceding customer or if they have, all of them are in L_2. In Line 3 of Algorithm 5.2, $\mathcal{P}_j$ is the set of all customers that are served immediately before customer j by the same vehicle(s). In Algorithm 5.2, whenever we determine the start time of the service for a customer j, we remove that customer from all sets $\mathcal{P}_k$ (see Line 6). In Line 5, we compute the start times of services to customers in L_1. In this line, dep_{rj} denotes the time that a type r vehicle departs the depot to visit customer j. We note that dep_{rj} is available as a part of the first-stage solution and we keep this notation for the remainder of this paper. We update L_1 and L_2 in lines 7 and 8. Although we can compute the optimal values of second-stage variables using Algorithm 5.2, it does not help us generate feasibility and optimality cuts (5.33) and (5.34) directly because we need dual values or infeasibility extreme rays corresponding to the constraints of subproblems.

Algorithm 5.2. Computation of start times for a fixed first-stage solution in scenario ω

```
1: Initialize $L_1 = \{j \in \mathcal{I} | \hat{x}_{r0j} = 1 \text{ for } r \in R_j\}$ and $L_2 = \varnothing$
2: for $r \in \mathcal{R}_j, i, j \in \mathcal{I}$ define $p_{rj} = i$ if $\hat{x}_{rij} = 1$
3: for $j \in \mathcal{I}$ initialize $\mathcal{P}_j = \{i \in \mathcal{I} | \exists r \in \mathcal{R}_j : \hat{x}_{rij} = 1\}$
4: while $(L_1 \neq \varnothing)$ do
5:     for $j \in L_1$ set $start_j = \max_{r \in \mathcal{R}_j} \{g(r,j) + s_{r(p_{rj})\omega} + t_{r(p_{rj})j\omega}\}$ where $g(r,j) = dep_{rj}$ if $p_{rj} = 0$
       and $g(r,j) = start_{p_{rj}}$ otherwise.
6:     for $j \in L_1, k \in \mathcal{I} : j \in \mathcal{P}_k$ set $\mathcal{P}_k := \mathcal{P}_k \setminus \{j\}$
7:     Set $L_2 := L_2 \cup L_1$ and $L_1 := \varnothing$
8:     for $k \in \mathcal{I}$ if $(\mathcal{P}_k = \varnothing$ and $k \notin L_2)$ set $L_1 := L_1 \cup \{k\}$
9: end while
```

5.7.1 Optimality cuts

In this section, knowing that the subproblem is feasible, we aim to analyze its dual formulation in order to compute the optimal values of dual variables. As stated in Lemma 5.3, subproblem

(SP$_\omega$) has a special structure and for a fixed first-stage solution, we can find the optimal second-stage solution by serving customers as soon as all required vehicles are available. The special structure of subproblem (SP$_\omega$) motivated us to analyze the dual formulation of the subproblem in order to see if there is any shortcut to find the optimal dual solution.

Before writing the dual formulation of subproblem (SP$_\omega$) given by (5.35)-(5.38), we note that upper bounds constraints in (5.37)-(5.38) are trivial and we can remove them. The redundancy of $u_{rijt\omega} \leq 1$ in constraint (5.37) is clear with respect to constraint (5.3) in the master problem, constraint (5.11) in the subproblem and $u_{rijt\omega} \geq 0$. The redundancy of $v_{it\omega} \leq 1$ is also obvious with respect to constraints (5.3), (5.11), and (5.13) in the master problem and the subproblem respectively. Another point in the subproblem is that, with respect to constraint (5.11), constraint (5.15) is redundant in the case that $\hat{x}_{r0i} = 0$ holds. Considering the above points, by ignoring the upper bounds constraints in (5.37)-(5.38) and constraint (5.15) for cases that $\hat{x}_{r0i} = 0$ holds, we write the dual formulation of the subproblem (SP$_\omega$) as follows.

$$(\mathrm{D}_\omega) \qquad \max_{\boldsymbol{\pi}} \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in\mathcal{I}_r\cup\{0\}:\\(i,j)\in\mathcal{A}_r}} \hat{x}_{rij}\pi^{(1)}_{rij\omega} + \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \hat{x}_{r0i}=1}} \sum_{t\in\mathcal{T}^{dep}_{ri}} \hat{y}_{rit}\pi^{(2)}_{rit\omega} \tag{5.41}$$

Subject to :

$$\pi^{(1)}_{r0j\omega} + \sum_{\substack{t'\in\mathcal{T}_{j\omega}:\\ t+t_{r0j\omega}\leq t'}} \pi^{(3)}_{rjt'\omega} + F_{(\hat{x}_{r0j}=1)}\pi^{(2)}_{rjt\omega} \leq 0 \quad r\in\mathcal{R}, j\in\mathcal{I}_r : (0,j)\in\mathcal{A}_r, t\in T_{r0j\omega} \tag{5.42}$$

$$\pi^{(1)}_{rij\omega} + \sum_{\substack{t'\in\mathcal{T}_{j\omega}:\\ t+s_{ri\omega}+t_{rij\omega}\leq t'}} \pi^{(3)}_{rjt'\omega} - \sum_{\substack{t'\in\mathcal{T}_{i\omega}:\\ t\leq t'}} \pi^{(3)}_{rit'\omega} + \pi^{(4)}_{rit\omega} \leq 0 \quad r\in\mathcal{R}, i,j\in\mathcal{I}_r : (i,j)\in\mathcal{A}_r, t\in T_{rij\omega} \tag{5.43}$$

$$\pi^{(1)}_{ri0\omega} - \sum_{\substack{t'\in\mathcal{T}_{i\omega}:\\ t\leq t'}} \pi^{(3)}_{rit'\omega} + \pi^{(4)}_{rit\omega} \leq \lambda_{rit\omega} \quad r\in\mathcal{R}, i\in\mathcal{I}_r : (i,0)\in\mathcal{A}_r, t\in T_{ri0\omega} \tag{5.44}$$

$$-\sum_{r\in\mathcal{R}} \pi^{(4)}_{rit\omega} \leq c^d_{it} \quad i\in\mathcal{I}, t\in\mathcal{T}_{i\omega} \tag{5.45}$$

$$\pi^{(3)}_{rjt\omega} \geq 0 \quad r\in\mathcal{R}, j\in\mathcal{I}_r, t\in T_{j\omega} \tag{5.46}$$

In Model (D$_\omega$), $\pi^{(1)}_{rij\omega}$, $\pi^{(2)}_{rjt\omega}$, $\pi^{(3)}_{rjt\omega}$, and $\pi^{(4)}_{rit\omega}$ denote the dual variables corresponding to constraints (5.11), (5.15), (5.26) and (5.13) respectively. To simplify the model and also the analysis that follows, we have defined a new parameter $\lambda_{rit\omega}$ by $\lambda_{rit\omega} = c^o_{rit\omega} + c^w_r f_{rit\omega}$. In constraint (5.42) and also in the remainder of the paper, we note that $F_{(.)}$ is a constant value that is equal to 1 if condition (.) is satisfied, and 0 otherwise.

In the following, we define some sets based on which we propose some formulas to compute the optimal values of dual variables $\pi^{(1)}_{rij\omega}$, $\pi^{(2)}_{rjt\omega}$, $\pi^{(3)}_{rjt\omega}$, and $\pi^{(4)}_{rit\omega}$. Before that, we need to define the concepts of "critical and non-critical vehicles". For a given first-stage solution, based on the start times that we obtain by Algorithm 5.2, we refer to a vehicle as a "critical vehicle" of customer j in scenario ω if the service starts as soon as the vehicle arrives to the customer's location. In other words, if the vehicle arrives one time unit later than the current arrival time, the service to the customer postpones for the same amount of time. Conversely, we refer to a vehicle as a "non-critical vehicle" of customer j in scenario ω if the start time of the service to the customer in this scenario is later than the vehicle arrival time.
Sets:

$\mathcal{NR}_{j\omega}$:	The set of non-critical vehicles in scenario ω for customer j.
$\mathcal{CR}_{j\omega}$:	The set of critical vehicles in scenario ω for customer j.
$\mathcal{NT}_{\omega}$:	The set of customers without any critical vehicle in scenario ω. For these customers, the services start as soon as their time windows open.
$\mathcal{CT}_{\omega}$:	The set of customers with at least one critical vehicle in scenario ω.

In addition to sets defined above, we define some notation as follows. k_{rj} and i_{rj} respectively denote the customers that are visited by a type r vehicle immediately after and before customer j (i.e., $\hat{x}_{rjk_{rj}} = 1$ and $\hat{x}_{ri_{rj}j} = 1$). Also, $t^*_{j\omega}$ denotes the start time of the service to customer j in scenario ω that we computed by Algorithm 5.2. Moreover, we suppose that $\alpha_{rj\omega}$ $j \in \mathcal{CT}_{\omega}, r \in \mathcal{CR}_{j\omega}$ are arbitrary parameters satisfying relations (5.47)-(5.48). We can consider $\alpha_{rj\omega}$ as the degree of criticality for vehicle r in serving customer j in scenario ω. In the case that vehicle r is the only critical vehicle for customer j, $\alpha_{rj\omega}$ is equal to 1 ; otherwise it could be less than 1.

$$\sum_{r \in \mathcal{CR}_{j\omega}} \alpha_{rj\omega} = 1 \qquad j \in \mathcal{CT}_{\omega} \tag{5.47}$$

$$\alpha_{rj\omega} \geq 0 \qquad j \in \mathcal{CT}_{\omega}, r \in \mathcal{CR}_{j\omega} \tag{5.48}$$

Based on the sets and notation defined above, we propose the following formulas to compute the optimal values of dual variables in Model (D$_\omega$).

$$\pi^{(4)}_{rjt\omega} = -\alpha_{rj\omega} \left(c^d_{j\left(min\left(t,t^*_{j\omega}\right)\right)} + \sum_{r' \in \mathcal{NR}_{j\omega} : k_{r'j}=0} \lambda_{r'j\left(min\left(t,t^*_{j\omega}\right)\right)\omega} \right) + \alpha_{rj\omega} \sum_{\substack{r' \in \mathcal{NR}_{j\omega}: \\ k_{r'j} \neq 0}} \sum_{\substack{t' \in \mathcal{T}_{k_{r'j}\omega}: \\ t + s_{r'j\omega} + t_{r'jk_{r'j}\omega} \leq t'}} \pi^{(3)}_{r'k_{r'j}t'\omega} \qquad j \in \mathcal{CT}_{\omega}, r \in \mathcal{CR}_{j\omega} : t \in \mathcal{T}_{ri_{rj}j\omega} \tag{5.49}$$

$$\pi^{(4)}_{rjt\omega}=-\sum_{\substack{t'\in\mathcal{T}_{k_{rj}\omega}:\\ t+s_{rj\omega}+t_{rjk_{rj}\omega}\leq t'}}\pi^{(3)}_{rjk_{rj}t'\omega}F_{(\hat{x}_{rj0}=0)}+\lambda_{rj\left(min\left(t,t^{*}_{j\omega}\right)\right)\omega}F_{(\hat{x}_{rj0}=1)}\qquad j\in\mathcal{CT}_{\omega},r\in\mathcal{NR}_{j\omega},t\in\mathcal{T}_{j\omega}\tag{5.50}$$

$$\pi^{(2)}_{rjt\omega}=-\sum_{\substack{t'\in\mathcal{T}_{j\omega}:t+t_{r0j\omega}\leq t'\\ t'\leq dep_{rj}+t_{r0j\omega}}}\pi^{(3)}_{rjt'\omega}\qquad j\in\mathcal{CT}_{\omega},r\in\mathcal{CR}_{j\omega}:x_{r0j}=1,t\in\mathcal{T}_{r0j\omega}:t<dep_{rj}\tag{5.51}$$

$$\pi^{(1)}_{rij\omega}=\alpha_{rj\omega}\left.\quad c_{i\left(t^{*}_{i\omega}\right)}+\sum_{r'\in\mathcal{NR}_{i\omega}:k_{r'i}=0}\lambda_{r'i\left(t^{*}_{i\omega}\right)\omega}\right)+\lambda_{ri\left(t^{*}_{i\omega}\right)\omega}F_{(\hat{x}_{ri0}=1)}\qquad i\in\mathcal{CT}_{\omega},r\in\mathcal{CR}_{i\omega},j\in\mathcal{I}_{r}\cup\{0\}:\hat{x}_{rij}=1\tag{5.52}$$

$$\pi^{(3)}_{rjt\omega}=\left(\pi^{(4)}_{rjt\omega}-\pi^{(4)}_{rj(t+1)\omega}\right)+F_{(\hat{x}_{rj0}=1)}\left(\lambda_{rj(t+1)\omega}-\lambda_{rjt\omega}\right)+F_{\left[(\hat{x}_{rj0}\neq 0)\ \&\ (t+s_{rj\omega}+t_{rjk_{rj}\omega})\in\mathcal{T}_{k_{rj}\omega}\right]}\pi^{(4)}_{rk_{rj}(t+s_{rj\omega}+t_{rjk_{rj}\omega})\omega}\qquad j\in\mathcal{CT}_{\omega},r\in\mathcal{CR}_{j\omega},t\in\mathcal{T}_{j\omega}:t<t^{*}_{j\omega}\tag{5.53}$$

$$\pi^{(1)}_{ri0\omega}=\lambda_{ri(t^{*}_{i\omega})\omega}\qquad i\in\mathcal{NT}_{\omega},r\in\mathcal{R}:\hat{x}_{ri0}=1\tag{5.54}$$

$$\pi^{(1)}_{r0j\omega}=\min_{t\in\mathcal{T}_{r0j\omega}}\left(-\sum_{\substack{t'\in\mathcal{T}_{j\omega}:\\ t+t_{r0j\omega}\leq t'}}\pi^{(3)}_{rjt'\omega}-F_{(\hat{x}_{r0j}=1)}\pi^{(2)}_{rjt\omega}\right)\qquad r\in\mathcal{R},j\in\mathcal{I}_{r}:(0,j)\in\mathcal{A}_{r}\ \&\ \hat{x}_{r0j}=0\tag{5.55}$$

$$\pi^{(1)}_{rij\omega}=\min_{t\in\mathcal{T}_{rij\omega}}\left(-\sum_{\substack{t'\in\mathcal{T}_{j\omega}:\\ t+s_{ri\omega}+t_{rij\omega}\leq t'}}\pi^{(3)}_{rjt'\omega}+\sum_{\substack{t'\in\mathcal{T}_{i\omega}:\\ t\leq t'}}\pi^{(3)}_{rit'\omega}-\pi^{(4)}_{rit\omega}\right)\qquad r\in\mathcal{R},i,j\in\mathcal{I}_{r}:(i,j)\in\mathcal{A}_{r}\ \&\ \hat{x}_{rij}=0\tag{5.56}$$

$$\pi^{(1)}_{ri0\omega}=\min_{t\in\mathcal{T}_{ri0\omega}}\left(\lambda_{rit\omega}+\sum_{\substack{t'\in\mathcal{T}_{i\omega}:\\ t\leq t'}}\pi^{(3)}_{rit'\omega}-\pi^{(4)}_{rit\omega}\right)\qquad r\in\mathcal{R},i\in\mathcal{I}_{r}:(i,0)\in\mathcal{A}_{r}\ \&\ \hat{x}_{ri0}=0\tag{5.57}$$

$$\textit{All other variables = 0}\tag{5.58}$$

LEMMA 5.7 *For any realization of parameters $\alpha_{rj\omega}$ $j\in\mathcal{CT}_{\omega},r\in\mathcal{CR}_{j\omega}$ satisfying relations (5.47)-(5.48), the solution obtained by formulas (5.49)-(5.58) is a feasible dual solution for Model (D_{ω}).*

LEMMA 5.8 *For any realization of parameters $\alpha_{rj\omega}$ $j\in\mathcal{CT}_{\omega},r\in\mathcal{CR}_{j\omega}$ satisfying relations (5.47)-(5.48), the objective value of the dual solution obtained by formulas (5.49)-(5.58) is equal to the optimal objective value of subproblem (SP_{ω}).*

THEOREM 5.7 *For any realization of parameters $\alpha_{rj\omega}$ $j\in\mathcal{CT}_{\omega},r\in\mathcal{CR}_{j\omega}$ satisfying relations (5.47)-(5.48), the solution obtained by formulas (5.49)-(5.58) is an optimal solution of Model (D_{ω}).*

The validity of Theorem 5.7 is based on the strong duality theorem in linear programming. Lemmas 5.7 and 5.8 demonstrate that the dual solution obtained by (5.49)-(5.58) is feasible and has an objective value that is equal to the optimal objective value of the primal subproblem. Theorem 5.7 indicates that, instead of using simplex or interior point algorithms, we can simply use relations (5.49)-(5.58) in order to find the values of dual variables $\pi^{(1)}_{crij\omega}$ and $\pi^{(2)}_{crit\omega}$ in optimality cut (5.33).

5.7.2 Feasibility cuts

The only way that, for a first-stage solution, subproblem (SP$_\omega$) may turn out infeasible is that for at least one vehicle, the tour does not complete within the scheduling horizon. The first idea to prevent from revisiting first-stage solutions with infeasible subproblems is to use the following simple no-good cut.

$$\sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \exists t\in\mathcal{T}^{dep}_{ri}:\hat{y}_{rit}=1}} y_{ri(dep_{ri})} + \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r \,\&\, \hat{x}_{rij}=1}} x_{rij} \leq n-1 \tag{5.59}$$

In (5.59), n is the sum of the number of $\hat{x}_{rij}$ and $\hat{y}_{rit}$ variables that are to equal to 1. It is clear that if the subproblem is infeasible for a given first-stage solution, modified solutions obtained by postponing the departure times are also infeasible. Therefore, we can enhance (5.59) as follows.

$$\sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \exists t\in\mathcal{T}^{dep}_{ri}:\hat{y}_{rit}=1}} \sum_{\substack{t'\in\mathcal{T}^{dep}_{ri}:\\ t'\geq dep_{ri}}} y_{rit'} + \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r \,\&\, \hat{x}_{rij}=1}} x_{rij} \leq n-1 \tag{5.60}$$

No-good cuts are generally known as weak cuts. Therefore, in the following, we explain how to develop stronger feasibility cuts. In order to propose the new feasibility cuts, we first need to define the notions of "path" and "critical path". We define a path P by $P = (P_{nodes}, P_{resources})$ where $P_{nodes} = \langle a_v \rangle_{v=1\,to\,|P_{nodes}|}$ is a sequence of $|P_{nodes}|$ nodes in $\mathcal{I}\cup\{0\}$ visited on the path and $P_{resources} = \langle r_v \rangle_{v=1 \text{ to } |P_{nodes}|-1}$ is series of vehicles types corresponding to arcs (a_v, a_{v+1}) for $v = 1$ to $|P_{nodes}| - 1$. The vehicles types are not necessarily the same for different arcs. We also suppose that the destination of the last arc is the depot (i.e. $a_{|P_{nodes}|} = 0$), while the origin may or may not be the depot. For a given first-stage solution, we refer to a path as a "critical path" in scenario ω if the two following conditions are satisfied :

1. The vehicle corresponding to each arc on the path is a critical vehicle for the customer at the arc tail (we defined the notion of "critical vehicle" in Section 5.7.1)

2. The sum of travel and service times on the path plus the start time of the service to the first customer on the path violates the scheduling horizon's time limit.

The reason for infeasibility of subproblem (SP$_\omega$) for a given first-stage solution, is the existence of at least one critical path. In the following, we present two types of no-good cuts in order to prevent the part of the first-stage solution resulting in critical paths. We devise the first cut for critical paths originating from the depot. For this type of critical paths, we propose the following cut using the notation $P = (P_{nodes}, P_{resources})$ explained above.

$$\sum_{t \in \mathcal{T}^{dep}_{r_1 a_2} : t \geq dep_{r_1 a_2}} y_{r_1 a_2 t} + \sum_{v=1}^{|P_{nodes}|-1} x_{r_v a_v a_{v+1}} \leq |P_{nodes}| - 1 \tag{5.61}$$

We propose the second type of cuts for critical paths not originating from the depot. In this case, all vehicles visiting the first customer are non-critical and the service to the customer starts when his time window opens. We can write the no-good cut as follows without any knowledge about customers visited before the first customer on the path and any departure time.

$$\sum_{v=1}^{|P_{nodes}|-1} x_{r_v a_v a_{v+1}} \leq |P_{nodes}| - 2 \tag{5.62}$$

We apply Algorithm 5.3 to extract critical paths for a given first-stage solution in scenario ω. To explain Algorithm 5.3, we first need to describe the two following networks. Network N is induced by non-zero variables $\hat{x}_{rij}$. In this network, $\mathcal{N}_{nodes} = \mathcal{I} \cup \{0\}$ is the set of nodes and the set of arcs includes arc (i, j) if there is at least one vehicle $r \in \mathcal{R}$ for which we have $\hat{x}_{rij} = 1$. We modify Network N to obtain Network N'. We make a copy of the depot (node 0) and denote it by $0'$. Then, we replace all active arcs $(i, 0)$ with arcs $(i, 0')$. We also denote the set of arcs in Network N' by $\mathcal{N}'_{arcs}$.

In Line 1 of Algorithm 5.3, *Critical_Paths* is a set including critical paths that the algorithm generates. The algorithm updates this set within Procedure 5.1. Moreover, $Start_{0'}$, that is set to $|\mathcal{T}|+1$, is the smallest tour makespan violating the available time in the scheduling horizon. $Start_i$ stands for the start time of the service to customer i that we compute by Algorithm 5.2. In Line 2 of Algorithm 5.3, P denotes the partial critical path that the algorithm extends by moving backward from node $0'$ in Network N'. Using Procedure 5.1, we extend the incomplete critical path P. In this procedure, a_1 denotes the first node at the beginning of partial path P. In lines 1 to 9, for all arcs $(i, a_1) \in \mathcal{N}'_{arcs}$ and resources $r \in \mathcal{R}_{a_1}$ satisfying $x_{ria_1} = 1$, we check the *if statement* in Line 3. If the given condition in Line 3 is true, it means that node

i is on a critical path followed by partial critical path P, and therefore we can add node i and resource r to the beginning of P_{nodes} and $P_{resources}$ respectively. However, since we can also probably extend the partial critical path P through other nodes and resources, we first make a copy of path P in Line 4 and then, in Line 5, we add node i and resource r to the beginning of the node set and the resource set of the new path respectively. In Line 6, we call Procedure 1 for extending the new path P'. In Line 10, we check if path P is extended in the *for loop* started in Line 1. If it is not extended, it means that the path is a complete critical path and we add it to *Critical_Paths*.

Algorithm 5.3. Generation of critical paths for a fixed first-stage solution in scenario ω

```
1: Set $Critical\_Paths = \varnothing$, $Start_{0'} = |\mathcal{T}| + 1$ and $Start_i$ for $i \in I$ by Algorithm 5.2.
2: Set $P = (P_{nodes}, P_{resources})$ with $P_{nodes} = \langle 0' \rangle$ and $P_{resources} = \varnothing$.
3: Call Procedure 5.1 to extend path $P$.
```

Procedure 5.1. Backward extension of the partial critical path P

```
1: for $(i, a_1) \in \mathcal{N}'_{arcs}$ do
2:   for $r \in \mathcal{R}_{a_1} : \hat{x}_{ria_1} = 1$ do
3:     if $Start_{a_1} - t_{ri(a_1)\omega} - s_{ri\omega} \leq Start_i$ then
4:       Make a copy of path $P$ and name it $P'$
5:       Add node $i$ and resource $r$ to the beginning of $P'_{nodes}$ and $P'_{resources}$ respectively.
6:       Call Procedure 5.1 to extend path $P'$.
7:     end if
8:   end for
9: end for
10: if path P did not extend in the recent for loop then
11:   Add path P to $Critical\_Paths$
12: end if
```

5.8 Extension to time-dependent problems

As in the second-stage model, we discretized the available time, we can simply consider time dependency for travel and service times in our problem by replacing parameters $t_{rij\omega}$ and $s_{ri\omega}$ with time-dependent travel and service times. We introduce $s_{rit\omega}$ as the duration of the service to customer i provided by a type r vehicle in scenario ω when the service starts at time t. Also, we define $t_{rijt\omega}$ as the travel time of a type r vehicle from customer i to customer j in scenario ω assuming that the vehicle leaves customer i at time t. We suppose that the First-in-First-Out (FIFO) rule holds, i.e., if a vehicle starts its travel between two customers later, it does not arrive to the destination earlier. Mathematically, FIFO rule means that $t_1 + t_{rijt_1\omega} \leq t_2 + t_{rijt_2\omega}$ holds for $t_2 \geq t_1$. Ichoua et al. (2003) discussed FIFO vastly and proposed an algorithm to modify travel times in order to satisfy it. We also suppose that time-dependent service times are non-decreasing over time, i.e., $s_{rit_1\omega} \leq s_{rit_2\omega}$ holds for $t_2 \geq t_1$. To capture time dependency in our problem, we need to slightly modify constraint (5.12) and also time sets T_{ri}^{dep} and $T'_{rij\omega}$ in the second-stage model as follows.

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\ t'+s_{rit'\omega}+t_{rij(t'+s_{rit'\omega})\omega}\leq t}} u_{rijt'\omega} \geq \sum_{\substack{k\in(\mathcal{I}_r\cup\{0\}):\\ (j,k)\in\mathcal{A}_r \,\&\, t\in\mathcal{T}_{rjk\omega}}} u_{rjkt\omega} \quad r\in\mathcal{R}, j\in\mathcal{I}_r, t\in\mathcal{T}_{j\omega} \tag{5.63}$$

$$\mathcal{T}'_{rij\omega} = \Big\{t|t \geq max\{e_i, SRT_{r0ie_i\omega}\} \,\&\, max\{e_j, t+s_{rit\omega}+t_{rij(t+s_{rit\omega})\omega}\} + s_{rjt_1^*\omega} + SRT_{rj0t_2^*\omega} \leq |\mathcal{T}|\Big\}$$

where $t_1^* = max\{e_j, t+s_{rit\omega}+t_{rij(t+s_{rit\omega})\omega}\}$ and $t_2^* = t_1^* + s_{rjt_1^*\omega}$.

$$\mathcal{T}_{ri}^{dep} = \Big\{t\in\mathcal{T}|t + \max_{\omega\in\Omega}\{t_{r0it\omega} + s_{ri(t+t_{r0it\omega})\omega} + SRT_{ri0(t+t_{r0it\omega}+s_{ri(t+t_{r0it\omega})\omega})\omega}\} \leq |\mathcal{T}|\Big\}$$

Moreover, we slightly modify Algorithm 5.1 to consider time-dependent travel and service times in the computation of $SRT_{rijt\omega}$. In lines 4, 5 and 9 of Algorithm 5.1, we replace $t_{rjk\omega}$, $s_{rk\omega}$ and $s_{rj\omega}$ by $t_{rjk(d_j)\omega}$, $s_{rkt_1^*\omega}$, and $s_{rjt_2^*\omega}$ where $t_1^* = t + max\{d_j + t_{rjk(d_j)\omega}, e_k - t\}$ and $t_2^* = t + d_j - s_{rjt_2^*\omega}$.

By considering time-dependency in our model, Theorem 5.1 remains valid. In addition, we can simply modify (5.26)-(5.29) to benefit from Theorems 5.2 and 5.3. Theorem 5.4 does not hold in the current problem setting. However, it holds if we either ignore the waiting cost in the objective function of the second-stage model or replace $g_{rijt\omega}$ with $g_{rijt} = \bar{s}_{ri} + \bar{t}_{rij}$ where $\bar{s}_{ri}$ denotes an average value of service time for customer i by a type r vehicle and $\bar{t}_{rij}$ stands for the travel time from customer i to customer j for the same vehicle type. The statement of Lemma 5.5 also does not necessarily hold and therefore lower bounding functional proposed by Theorem 5.5 based on Lemma 5.5 is not valid anymore. It is clear that L-shaped algorithm and the enhancements that we proposed in Sections 5.5, 5.6 and 5.7 are not applicable when we have a problem with travel and service times that are simultaneously time-dependent and stochastic. However, we can use the deterministic version of the proposed model in Section 5.3 to solve problems with time-dependent travel and service times.

5.9 Implementation details

In this section, we provide the implementation details of the L-shaped algorithm and of the L-shaped-based-branch-and-cut algorithm that we simply refer to as branch-and-cut algorithm. In the branch-and-cut algorithm, we solve subproblems only when we find a first-stage feasible solution, i.e., do not solve them for fractional first-stage solutions in any node in the branch-and-bound tree. For solving subproblems, we use the fast solution method proposed in Section 5.7. To generate the optimality cuts using formulas (5.47)-(5.58), we set $\alpha_{r^*j\omega} = 1$ and $\alpha_{rj\omega} = 0$, $r \in \mathcal{CR}_{j\omega} \backslash \{r^*\}$ where r^* is randomly selected from $\mathcal{CR}_{j\omega}$ for $\omega \in \Omega$, $j \in \mathcal{CT}_{\omega}$. In all nodes with fractional first-stage solutions, we call a maximum flow algorithm to separate the subtour-elimination constraints presented in Section 5.4.1. This algorithm is a combination of the maximum flow algorithm developed by Goldberg and Tarjan (1988) and the shrinking procedure proposed in Padberg and Rinaldi (1990). This algorithm separates the most violated subtour-elimination constraint. Jünger et al. (2000) tested several maximum flow algorithms existing in the literature and concluded that the above algorithm is the most efficient one. There are some freedoms for the implementation of Padberg-Rinaldi algorithm. We considered the details provided on pages 175 and 176 in Jünger et al. (2000) to address these freedoms.

Furthermore, we used the code provided by Lysgaard et al. (2004) for the separation of the Capacity cuts presented in Section 5.4.2 and other cuts including Framed capacity, Strengthened comb, Homogeneous multistar and Hypotour inequalities. We allow at most three rounds for the separation of these cuts at the root node of the branch-and-cut algorithm and of the master problem in the L-shaped algorithm. We consider all random scenarios separately for the generation of these cuts in the root node. In addition to the root node, for the first 100 nodes, we separate these cuts for only one scenario that is randomly selected in each node. For each round of the separation of above cuts, we used the strategy suggested in Section 5.3.1 of Lysgaard et al. (2004).

Additionally, since the master problem of the L-shaped algorithm is computationally demanding, we stop proving the optimality of the branch-and-bound tree for the master problem when it reaches a local optimality gap 1%. By local optimality gap, we mean the optimality gap of the branch-and-bound tree when solving the master problem not the optimality gap of the L-shaped algorithm. In order to find the optimal solution, we relax this stopping condition in the master problem whenever the L-shaped algorithm reaches an optimality gap 2%. Furthermore, in each iteration of the L-shaped algorithm, when solving the master problem, we save the best 50 obtained solutions and then generate the optimality and feasibility cuts by solving the subproblems for all these solutions. This is because each iteration of the master problem is computationally unwieldy and it is not efficient to generate feasibility and optimality cuts only for the best-obtained solution.

5.10 Computational results

We implemented the proposed algorithms in C++ and used IBM ILOG CPLEX Optimization Studio V12.6 to solve mixed integer programming models. We ran experiments on a computer with two Intel Xeon X5650 Westmere processors, 2,67 Ghz, and a total of 12 cores. We used a single core for running each test instance.

5.10.1 Home-health care scheduling instances

In this section, we explain how we generated a set of home-health care scheduling instances with stochastic travel and service times and another set of instances with time-dependent travel times. We refer to the first set of instances as stochastic home-health care scheduling instances and the other set as time-dependent instances.

To generate most of data in these instance sets, we used the data generation approach proposed by Di Mascolo et al. (2014) that is inspired from a real case in France. We slightly modified the proposed approach in order to consider stochasticity/time-dependency for travel and service times. In these instances, two groups of nurses including Registered Nurses (RNs) and Home Health Aides (HHAs) must serve patients that are uniformly dispersed in a square area with a side length of 40 kilometers. The Home-Healthcare Center (HHC) is located at the center of this area. For stochastic instances, we set the number of patients to {10, 15, 20}. Also for the time-dependent problem, we generated instances with {30, 40, 50, 60} patients. Moreover, we define synchronization rate as the percentage of patients requiring a simultaneous service by an RN and an HHS and set it to {10, 20, 30, 40}. To determine the type of required nurses for patients without any synchronization, we randomly divided them to two groups of the same size and supposed that these groups must by served by RNs and HHAs separately. For each patient, we generated the earliest start time and also the length of the time window from [0 min, 120 min] and [60 min, 180 min] randomly and then fixed the latest start time to the sum of time window's earliest start time and its length.

For stochastic instances, we generated 100 random scenarios for travel and service times. For each scenario, we randomly generated the service times from [20 min, 180 min]. Also, we generated the travel time between every pair of customers i and j from a normal distribution with a mean $\mu_{ij} = d_{ij}$ and a standard deviation $\sigma_{ij} = d_{ij}/6$ where d_{ij} is the Euclidean distance between customers i and j. For time-dependent instances, we generated travel times using the approach proposed by Ichoua et al. (2003). We first randomly assigned each arc to one of the three available traffic zones. For each traffic zone, the travel speed changes over the scheduling horizon. We used travel speeds provided in Table 5.1 of Ichoua et al. (2003). In this table, travel speeds for three traffic scenarios are given. In each traffic scenario, for each traffic zone, there are three time intervals with different travel speeds. In a fixed traffic scenario, assuming that a vehicle starts travelling from customer i and customer j at time slot t, we set the travel time to $\mu_{ij}/Speed_{\tau_t \eta_{ij}}$ where $\mu_{ij} = d_{ij}$ is the nominal value of the travel time on arc (i, j), τ_t denotes the time interval that time slot t belongs to, and $Speed_{\tau_t \eta_{ij}}$ denotes the travel speed in zone η_{ij} in time interval τ_t. To respect the FIFO rule, we modified travel times using the algorithm provided in Figure 4 of Ichoua et al. (2003). In both instance sets, we rounded travel times, service times and also the earliest and latest start times in time windows to the closest multiple of 5 minutes because the length of each time slot in our model is set to 5 minutes. As an exception, for time-dependent instances, we rounded down travel times to the nearest multiple of 5 minutes in order to preserve the FIFO rule.

We supposed that the normal session length for nurses is 9 hours after which overtime penalties are incurred. Moreover, we set the maximum available time for completing tours to 11 hours. Based on `http://www.payscale.com`, we estimated the fixed costs of hiring an RN and an HHA to be 94.41\$ and 216\$ per day that are equivalent to 10.49\$ and 24\$ per hour respectively. We also supposed that nurses are paid with double rates for working beyond the normal session length. Besides, we set the per hour delay cost for serving a patient to 15.73\$ that is equal to 1.5 times of an HHA's salary rate. Since we let the model decide about the number of nurses and we pay fixed costs for hiring RNs and HHA, the model tends to minimize waiting times to visit patients by a few nurses. Therefore, we did not consider waiting costs in the home-health care scheduling problem. However, in the case that the number of nurses is fixed a priori, one can consider waiting costs. We obtained 120 stochastic home-health care scheduling instances by generating 10 instances for each combination of the number of patients and the synchronization rate. Also, we obtained 480 time-dependent home-health care scheduling instances by generating 10 instances for each combination of the number of patients, the synchronization rate and the three traffic scenarios in Table 5.1 of Ichoua et al. (2003).

5.10.2 Results for home-health care scheduling instances

We report the results of the home-health care scheduling problem with stochastic travel and service times in Table 5.1. In this table, each row represents the average over 10 instances. Under "*Data Info.*", "*Pat. No.*" and "Syn (%)" respectively give the number of patients and the percentage of customers requiring synchronized visits. Under "*L-shaped algorithm*" and "*branch-and-cut algorithm*", we report the results of the proposed algorithms. As the branch-and-cut algorithm outperforms the L-shaped algorithm, we give more details for the branch-and-cut algorithm. "*Time (sec)*", give the computational time of algorithms in seconds. $LB_{.}$ and $UB_{.}$ indicate the best lower and upper bounds of algorithms, respectively, and $Gap_{.}$ computes the gap between these bounds. The subscripts of $LB_{.}$, $UB_{.}$, and $Gap_{.}$ are either "L" or "B" which represent the L-shaped and the branch-and-cut algorithms respectively. Also, Under "*L-shaped algorithm*", "*Ite.*" gives the number of times that the L-shaped algorithm has iterated between the master problem and subproblems. Under "*branch-and-cut algorithm*", we also have the following columns. "*Nodes No.*" gives the number of nodes that are examined in the branch-and-cut algorithm. "*Feas. Cut No.*" and "*Opt. Cut No.*" respectively indicate the number of generated feasibility and optimality cuts. We set a time limit of 24 hours for running instances. However, in order to show that the proposed branch-and-cut algorithm finds high quality upper bounds in less computational time, we report Column "$\Delta^{UB}_{4h,24h}(\%)$", which computes the gap between the upper bounds obtained after 4 and 24 hours. "VSS" indicates the value of the stochastic solution in percentage. We obtain this value by $VSS = 100(UB_{det} - UB_B)/UB_{det}$ where UB_{det} indicates the objective value of the stochastic problem for the solution obtained by solving the "mean-value" problem. By "mean-value" problem, we refer to the problem with a single scenario that is the average of all scenarios. For the fixed "mean-value" solution, if the stochastic problem turns out infeasible in at least one scenario, then we have $UB_{det} = \infty$ and $VSS = 100\%$.

In Table 5.1, the average values of Gap_B are 0.00%, 0.60% and 3.57% for instances with 10, 15 and 20 patients respectively, while the average of Gap_L for the same size instances are 1.27%, 1.88% and 6.70%. These values demonstrate that the branch-and-cut algorithm significantly outperforms the L-shaped algorithm, especially in larger-sized instances. We can see that as the size of instances increases, the problem gets more difficult and average values of Gap_B increase. Small values of $\Delta^{UB}_{4h,24h}$ demonstrate that the branch-and-cut algorithm finds high quality upper bounds in the first 4 hours of computational time. Figure 5.2 depicts the convergence of the lower and upper bounds averaged over all instances within the computational time. We can observe that significant portion of improvement in lower and upper bounds occurs during the first 4 hours. Moreover, in Table 5.1, we observe that all average values of VSS are 100% that demonstrates that considering stochasticity in modeling the home-health care scheduling problem is very important and solutions obtained by solving the "mean-value" problem are infeasible in the stochastic problem.

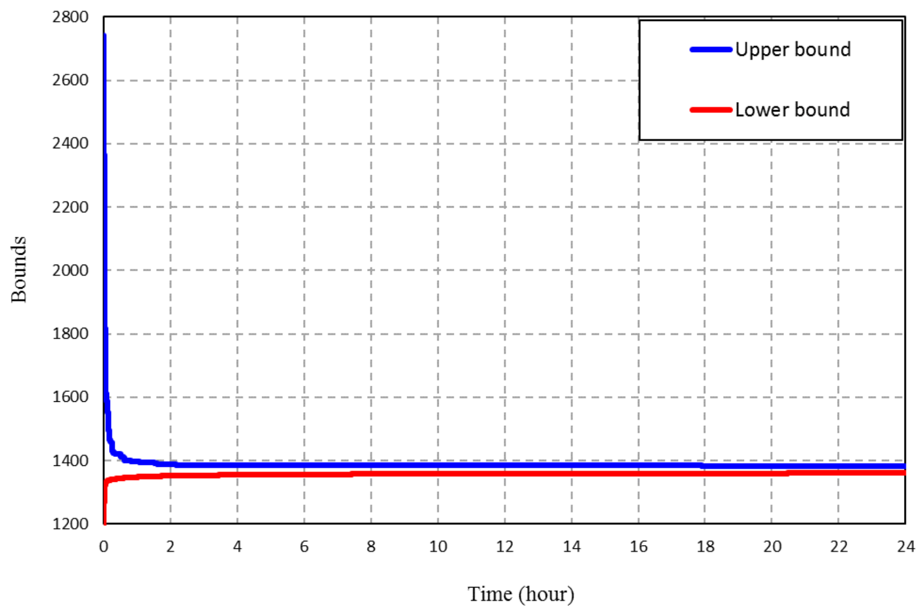

Figure 5.2 Convergence of lower and upper bounds in branch-and-bound algorithm for home-health care scheduling instances.

Table 5.2 presents the computational results of the branch-and-bound algorithm with different solution methods for subproblems. In this table, each row gives the average of results for 40 instances with different synchronization rates. Under "*Proposed Method*", we provide results for the case that subproblems are solved using the method proposed in Section 7. "*Primal Simplex*", "*Dual Simplex*", and "*Interior Point*" present computational results for cases that we used standard linear programming algorithms to solve subproblems. Furthermore, "*Ave. Time (sec)*" shows the average solution time for solving a single subproblem and "*Nodes No.*" indicate the number of nodes explored within a computational time of 30 minutes. Also, under columns *Primal Simplex*, *Dual Simplex*, and *Interior Point*, "*Time ratio*" gives the ratio of the corresponding *Ave. Time (sec)* to the *Ave. Time (sec)* of our proposed method. Average values of *Time ratio* demonstrate that our subproblem analysis method is 169, 483, and 196 times faster than primal simplex, dual simplex, and interior point methods respectively. Moreover, our proposed method explores a significantly higher number of nodes and provides lower optimality gaps within the time limit.

In Table 5.3, we provide computational results for the time-dependent home-health care scheduling problem. Each row presents the average of results for 30 instances with three speed scenarios given by Ichoua et al. (2003). In this table, under "Optimal Instances No. (out of 30)", we give the number of instances solved to optimality. Moreover, "*VTS*"

Table 5.1 – Computational results of the branch-and-cut and L-shaped algorithms for the home-healthcare scheduling problem with stochastic travel and service times.

Data Info.		*L-shaped algorithm*					*branch-and-cut algorithm*								
Pat. No.	*Syn. (%)*	*Ite.*	*Time (sec)*	LB_L	UB_L	$Gap_L(\%)$	*Node No.*	*Feas. Cut No.*	*Opt. Cut No.*	*Time (sec)*	LB_B	UB_B	$Gap_B(\%)$	$\Delta^{UB}_{4h,24h}(\%)$	$VSS(\%)$
10	10	25	3083	928	928	0.00	157	23	1984	109	928	928	0.00	0.00	100
	20	37	54356	955	958	0.29	918	46	3872	248	958	958	0.00	0.00	100
	30	30	86400	938	957	1.95	8680	652	9001	4193	957	957	0.00	0.00	100
	40	32	86400	988	1018	2.84	15837	2379	11801	10016	1016	1016	0.00	0.00	100
Average		31	57559	952	965	1.27	6398	775	6664	3641	965	965	0.00	0.00	100
15	10	42	86400	1276	1291	1.15	23347	285	9299	6343	1291	1291	0.00	0.00	100
	20	39	86400	1299	1314	1.11	7242	103	9291	3377	1314	1314	0.00	0.00	100
	30	22	86400	1412	1445	2.30	46217	2013	21401	62179	1435	1444	0.63	0.00	100
	40	22	86400	1448	1492	2.95	49339	4341	22244	79729	1465	1491	1.77	0.97	100
Average		31	86400	1359	1385	1.88	31536	1685	15558	37907	1376	1385	0.60	0.24	100
20	10	32	86400	1580	1628	3.05	43560	3127	34686	86406	1585	1628	2.74	0.83	100
	20	18	86400	1650	1732	4.96	53276	20942	20202	86406	1651	1719	4.12	0.81	100
	30	20	86400	1833	1956	5.87	37652	4750	31542	86409	1842	1901	3.24	0.01	100
	40	9	86400	1870	2307	12.92	40040	8875	22970	86407	1880	1958	4.19	1.36	100
Average		19	86400	1734	1906	6.70	43632	9424	27350	86407	1739	1802	3.57	0.75	100

Table 5.2 – Comparison of different solution methods for subproblems in stochastic home-health care scheduling instances within a time limit of 30 minutes.

Data Info.	*Proposed Method*			*Primal Simplex*				*Dual Simplex*				*Interior Point*			
Pat. No.	*Ave. Time (sec)*	*Nodes No.*	Gap_B *(%)*	*Ave. Time (sec)*	*Time ratio*	*Nodes No.*	Gap_B *(%)*	*Ave. Time (sec)*	*Time ratio*	*Nodes No.*	Gap_B *(%)*	*Ave. Time (sec)*	*Time ratio*	*Nodes No.*	Gap_B *(%)*
10	0.007	1835	1.23	0.596	86	403	3.64	1.563	220	60	8.28	0.653	94	194	6.24
15	0.012	2227	2.55	1.884	161	32	INF	4.581	378	0	INF	1.912	164	9	INF
20	0.016	802	24.56	4.291	259	1	INF	14.832	850	0	INF	5.516	330	0	INF
Average	0.012	1621	9.45	2.257	169	145	INF	6.992	483	20	INF	2.694	196	68	INF

shows the value of time-dependent solutions in percentage. We compute this value by $VTS = 100(UB - Opt_C)/Opt_C$ where UB stands for the best objective value obtained by solving the time-dependent problem. Also Opt_C indicates the optimal objective value of time-dependent problem for the solution obtained by solving the "constant-speed problem". By "constant-speed problem" we refer to the problem in which all time-dependent parameters are replaced by average values. As we can see, the proposed mixed integer programming model can solve almost all instances with 40 patients optimally. Also it finds the optimal solutions for instances with 50 and 60 patients where the synchronization rate is 20% and 10% respectively. For instances with 40, 50, and 60 patients the average optimality gap is 0.04%, 0.30%, and 0.79% respectively. We can see that the average value of VTS is 13.85%. It shows that solutions obtained by our method considerably outperform those solutions obtained by solving the constant-speed problem. In Table 5.3, we can also see that the number of generated nodes and the computational time are increasing in terms of the number of patients and the synchronization rate.

Table 5.3 – Computational results for the home-healthcare scheduling problem with time-dependent travel and service times.

Data Info.		*Deterministic time-dependent model*						
Pat. No.	*Syn. (%)*	*Optimal Instances No. (out of 30)*	*Nodes No.*	*Time (sec)*	*LB*	*UB*	*Gap(%)*	*VTS(%)*
30	10	30	85	814	2168	2168	0.00	13.17
	20	30	140	1209	2313	2313	0.00	14.62
	30	30	322	3351	2377	2377	0.00	13.67
	40	30	532	5907	2496	2496	0.00	14.51
Average		30	270	2820	2338	2338	0.00	13.99
40	10	30	243	2743	2691	2691	0.00	13.59
	20	30	614	6509	2860	2860	0.00	14.81
	30	29	1075	14375	3097	3098	0.03	13.47
	40	27	2075	37550	3290	3294	0.12	13.55
Average		29	1002	15294	2985	2986	0.04	13.86
50	10	30	436	5448	3341	3341	0.00	13.35
	20	30	1246	18278	3632	3633	0.00	13.61
	30	21	2304	53309	3779	3791	0.31	14.07
	40	14	1546	69118	4062	4095	0.91	12.66
Average		24	1383	36538	3704	3715	0.30	13.43
60	10	30	905	12079	3972	3972	0.00	14.44
	20	23	1820	44947	4199	4205	0.14	14.30
	30	9	1312	76139	4447	4482	0.80	14.34
	40	4	703	84603	4811	4918	2.22	13.44
Average		17	1185	54442	4357	4394	0.79	14.13

5.10.3 Operating room scheduling instances

For the operating room scheduling problem, we generated a set of instances with stochastic surgery, anesthesia and cleaning times and did not generate time-dependent instances as time dependency does make sense in this application. We set the number of surgeries to {11, 15, 20, 25}. For each instance, we generated 500 random scenarios that is the same as the number of scenarios used by Batun et al. (2011). We would have preferred to use the same instances for which Batun et al. (2011) reported their computational results. However, because of the data disclosure policy in Mayo Clinic, authors could not provide the same data and encouraged us to generate surgery and anesthesia durations using distributions provided in Table 1 of Gul et al. (2011). Gul et al. (2011) extracted these distributions from the data of 4034 patients at Mayo Clinic in the first 21 weeks of 2006. As no data is available in Gul et al. (2011) for cleaning times, we generated them from [0 min, 15 min] uniformly. After transforming the generated durations to travel and service times in the equivalent VRPS as explained in Section 2, we rounded travel and service times to the closest multiple of 5 minutes, that is the length of time slots in our model.

To generate operating room scheduling instances, we introduce a parameter ρ that denotes the average working time of a surgeon. We set ρ to {5, 7, 9} hours. We set the number of surgeons to $\lceil \gamma/\rho \rceil$ where γ denotes the sum of surgeries durations averaged over all scenarios. We assigned surgeries to surgeons as follows. The idea of the following procedure is to make balanced workloads for surgeons. We first sorted surgeries in a decreasing order of the average surgery duration. Then we assigned surgeries one by one from the sorted list to surgeons. To assign each surgery, among all surgeons, we chose the one with the lowest sum of assigned surgeries durations. After assigning all surgeries, we sequenced surgeries randomly for each surgeon. We supposed that the normal session length, during which no overtime penalty is paid, is 9 hours. We also considered the possibility of having overtime for at most 2 hours. As it does not make sense to consider time windows for the start time surgeries, we set earliest and latest start times to the beginning and the end of the day respectively. We also supposed that surgeons are available at the beginning of the scheduling horizon.

We used all cost coefficients provided by Batun et al. (2011). The fixed cost of opening an operating room is 4437\$. There is no waiting cost for operating rooms as we consider fixed cost for them. However, as there is no fixed cost for surgeons, we considered the waiting cost to be 88.74\$ per minute for them. We also set the overtime cost for surgeons and operating rooms to 133.11\$ and 12.37\$ per minute respectively. We generated 10 instances for each combination of ρ and the number of surgeries for a total of 120 instances.

5.10.4 Results for operating room scheduling instances

We report the results of the operating room scheduling problem with stochastic durations in Table 5.4. In this table, each row represents the average over 10 instances. Under "*Data Info.*", "*Pat. No.*" and "*Sur. Time Limit*" respectively give the number of patients and the value of parameter ρ used for the generation of instances. Other columns of this table are the same as similar columns in Table 5.1.

In Table 5.4, we observe that the values of Gap_B are considerably less than those of Gap_L. This observation demonstrates that the branch-and-cut algorithm strongly dominates the L-shaped algorithm in the operating room scheduling context too. In Table 5.4, the average values of Gap_B are 0.00%, 0.00%, 0.11% and 2.20% for instances with 11, 15, 20 and 25 surgeries respectively. We observe that most of instances with up to 20 surgeries are optimally solved. Furthermore, our branch-and-cut algorithm can solve instances with 25 surgeries and average surgeon time limits of 9 hours optimally. These results demonstrate that our branch-and-cut algorithm is significantly more effective than the algorithm proposed by Batun et al. (2011) which can solve instances with up to 10 and 11 surgeries optimally.

Moreover, in Table 5.4, the average values of VSS are 8.10%, 39.12%, 38.98%, and 40.06% which show that value of stochastic solution for instances with more surgeries is higher and the application of our algorithm is more justifiable and beneficial in such cases. In addition, we observe that VSS increases as the average surgeon time limit increases. This is because, in instances with higher average surgeon time limits (ρ), it is more likely that the mean-value solution results in unexpected overtime or infeasibility in the stochastic problem. The other noticeable point is that the values of $\Delta^{UB}_{4h,24h}$ are less than 0.78% which shows that our branch-and-cut algorithm improves the upper bound values within the first 4 hours of computational time. It is also noteworthy that the average of computational time to obtain optimal solutions of instances with 11 surgeries is 2231 seconds, while the solution time of the algorithm proposed by Batun et al. (2011) for two sets of instances with the same size is 4866 and 9992 seconds.

Similar to Table 5.3, Table 5.5 provides the computational results of the branch-and-bound algorithm for stochastic operating room scheduling instances with different solution methods of subproblems. In all cases, we set the time limit to 30 minutes. We observe that our proposed solution method for subproblems is 78, 76, and 59 times faster than primal simplex, dual simplex, and interior point methods, respectively. We can also observe that our method explores 198 nodes within the time limit, while other methods time out in the root node of the branch-and-bound tree. It is also noteworthy that the value of *Time ratio* increases in the number of patients.

Table 5.4 – Computational results of the branch-and-cut and L-shaped algorithms for the operating room scheduling problem with stochastic durations.

Data Info.		*L-shaped algorithm*					*branch-and-cut algorithm*								
Pat. No.	*Sur. Time Limit*	*Ite.*	*Time (sec)*	LB_L	UB_L	$Gap_L(\%)$	*Node No.*	*Feas. Cut No.*	*Opt. Cut No.*	*Time (sec)*	LB_B	UB_B	$Gap_B(\%)$	$\Delta^{UB}_{4h,24h}(\%)$	$VSS(\%)$
11	5	16	54059	25898	26016	0.44	1284	0	6164	3473	26016	26016	0.00	0.00	6.47
	7	17	2796	23459	23459	0.00	509	1	4785	1772	23459	23459	0.00	0.00	8.45
	9	15	19489	24384	24678	1.17	512	0	3719	1446	24484	24484	0.00	0.00	9.38
Average		16	25448	24580	24717	0.54	768	0	4889	2231	24653	24653	0.00	0.00	8.10
15	5	14	80136	32625	33969	4.19	6232	0	13430	10790	33127	33127	0.00	0.07	7.41
	7	11	70304	26983	27972	4.05	5016	7	7668	4200	27494	27494	0.00	0.00	9.95
	9	12	43979	23821	24692	3.73	759	54	4367	1064	24396	24396	0.00	0.00	100.00
Average		12	64807	27810	28878	3.99	4002	20	8488	5351	28339	28339	0.00	0.02	39.12
20	5	13	86400	44237	46472	5.09	9280	0	11693	11251	44832	44832	0.00	0.00	5.88
	7	8	36093	33678	35179	4.88	20125	1	9513	8753	34262	34262	0.00	0.00	11.06
	9	11	35910	30866	32323	4.93	14176	161	9945	8177	31821	31935	0.34	0.00	100.00
Average		11	52801	36261	37991	4.96	14527	54	10384	9393	36972	37010	0.11	0.00	38.98
25	5	8	86400	57045	60082	5.35	16782	0	39141	84461	57736	59577	3.23	0.43	3.32
	7	7	86400	40581	45347	11.96	19005	6	27569	86400	42110	43519	3.37	0.78	16.86
	9	4	48794	37642	40502	8.15	18906	1717	9503	7329	39956	39956	0.00	0.00	100.00
Average		7	73865	45089	48644	8.49	18231	574	25404	59397	46600	47684	2.20	0.40	40.06

Table 5.5 – Comparison of different solution methods for subproblems in stochastic operating room scheduling instances within a time limit of 30 minutes.

Data Info.	*Proposed Method*			*Primal Simplex*				*Dual Simplex*				*Interior Point*			
Pat. No.	*Ave. Time (sec)*	*Nodes No.*	Gap_B *(%)*	*Ave. Time (sec)*	*Time ratio*	*Nodes No.*	Gap_B *(%)*	*Ave. Time (sec)*	*Time ratio*	*Nodes No.*	Gap_B *(%)*	*Ave. Time (sec)*	*Time ratio*	*Nodes No.*	Gap_B *(%)*
11	0.053	307	2.79	1.91	37	0	24.64	1.96	37	0	23.86	1.97	38	0	24.31
15	0.058	280	6.62	3.35	58	0	INF	3.76	61	0	INF	2.93	49	0	INF
20	0.088	114	10.03	7.38	84	0	INF	9.23	91	0	INF	5.79	67	0	INF
25	0.107	89	12.81	17.57	132	0	INF	15.28	114	1	INF	9.20	80	0	INF
Average	0.077	198	8.06	7.55	78	0	INF	7.56	76	0	INF	4.97	59	0	INF

5.11 Conclusion

In this paper, we studied a vehicle routing problem with synchronized visits (VRPS) and stochastic/time-dependent travel and service times. In addition to considering a home-health care scheduling problem, we cast an operating room scheduling problem with stochastic durations as a VRPS. We developed a two-stage stochastic integer programming model to formulate VRPS with stochastic times. In contrast to the deterministic models in the VRPS literature, our proposed formulation is free of big-M constraints. We obtained this advantage by splitting the available time into smaller time slots that resulted in a large number of second-stage integer variables. We proved that the integrality constraints on second-stage variables are redundant, thus can be relaxed. Having continuous variables in the second stage, we applied the L-shaped algorithm and its branch-and-cut implementation as solution methods. Moreover, we improved the proposed approach by devising valid inequalities and a lower bounding functional. We also analyzed the subproblems of the L-shaped algorithm and proposed a solution method for them that is significantly faster than standard linear programming algorithms. Furthermore, we discussed how to modify the proposed formulation to solve VRPS with deterministic time-dependent travel and service times.

Computational experiments revealed that, in the stochastic home-health care scheduling problem, the branch-and-cut algorithm solves instances with 15 patients and 10% to 30% of synchronized visits to optimality. In addition, it finds solutions with an average optimality gap of 3.57% for instances with 20 patients. In the stochastic operating room scheduling problem, the branch-and-cut algorithm is capable of finding optimal solutions for instances with 20 surgeries. This is a considerable improvement over the state-of-art algorithm that reports on instances with 11 surgeries. Moreover, the modified model for the time-dependent problem obtains solutions with an average optimality gap of 0.79% for home-health care scheduling instances with 60 patients and 10% to 40% of synchronized visits.

CHAPTER 6 ARTICLE 3: EXPLOITING THE STRUCTURE OF TWO-STAGE ROBUST OPTIMIZATION MODELS WITH INTEGER ADVERSARIAL VARIABLES

Seyed Hossein Hashemi Doulabi

Department of Mathematics and Industrial Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Patrick Jaillet

Department of Electrical Engineering and Computer Science
Laboratory for Information and Decision Systems

Operations Research Center, MIT, Cambridge, USA

Gilles Pesant

Department of Computer and Software Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Louis-Martin Rousseau

Department of Mathematics and Industrial Engineering, Polytechnique Montreal

Interuniversity Research Center on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada

Abstract. This paper addresses a class of two-stage robust optimization models with integer adversarial variables. We discuss the importance of this class of problems in modeling two-stage robust resource planning problems where some tasks have uncertain arrival times and duration periods. We apply Dantzig-Wolfe decomposition to exploit the structure of these models and show that the original problem reduces to a single-stage robust problem. We propose a Benders algorithm for the reformulated single-stage problem. Since the master problem and subproblem in the Benders algorithm are mixed integer programs, it is computationally demanding to solve them optimally at each iteration of the algorithm. Therefore, we develop novel stopping conditions for these mixed integer programs and provide the relevant convergence proofs. We also develop a heuristic algorithm called dual algorithm. In this heuristic, we dualize the linear programming relaxation of the adversarial problem in the reformulated problem and iteratively generate cuts to shape the convex hull of the uncertainty set. We combine this heuristic with the Benders algorithm to create a more effective algorithm called Benders-dual algorithm. Extensive computational experiments on a two-stage nurse planning problem are performed to compare these algorithms.

Keywords. Integer programming, Dantzig-Wolfe decomposition, two-stage robust optimization, nurse planning problem.

6.1 Introduction

In the operations research literature there are many different methodologies to address uncertainty in optimization problems. Stochastic approaches are one of the main classes and are applicable if probability distributions of uncertain parameters are known. However, these approaches are usually criticized for requiring information on the probability distributions and also for computational complexities. Robust optimization, a more recent methodology, generally assumes that uncertain parameters belong to an uncertainty set, and aims to find a robust solution immunizing the decision maker against the worst-case scenario within this uncertainty set.

In robust optimization problems choosing an appropriate uncertainty set is critical and can highly affect the robustness and the optimal objective value of the obtained solution. The decision maker should select a suitable uncertainty set to reasonably represent the randomness of the uncertain parameters while taking into account the computational issues arising in the solution algorithm. From the literature on robust optimization, the most prevalent uncertainty sets are box uncertainty sets (Soyster 1973), ellipsoidal uncertainty sets (Ben-Tal and Nemirovski 1999, El Ghaoui and Lebret 1997, El Ghaoui et al. 1998), polyhedral uncertainty sets and Γ-cardinality uncertainty sets (Bertsimas and Sim 2004). In box uncertainty sets, uncertain parameters are assumed to take their values from different intervals independently. Box uncertainty sets usually result in overly conservative solutions because all parameters are allowed to take their worst values simultaneously. Ellipsoidal uncertainty sets alleviate this issue by restricting the uncertain parameters to an ellipsoidal space and this prevents them from taking worst values at the same time. Polyhedral uncertainty sets confine the uncertain parameters to a polyhedral space and can be viewed as a special case of ellipsoidal uncertainty sets (Ben-Tal and Nemirovski 1999). In Γ-cardinality uncertainty sets, for each constraint the number of uncertain parameters deviating from their nominal values must be less than Γ.

Robust optimization was initially proposed for single-stage optimization problems where the decision maker must choose a complete solution before the disclosure of information about the real values of uncertain parameters (Soyster 1973, Ben-Tal and Nemirovski 1999). Then it was extended to multi-stage problems where the values of uncertain parameters are revealed gradually in some stages (Ben-Tal et al. 2004, Delage and Iancu 2015). In multi-stage robust problems the decision maker does not choose a complete solution at the beginning, but instead makes partial decisions sequentially after observing the values of uncertain parameters over different stages. The worst-case data realization in each stage is obtained by solving what is referred to sometimes as an "*adversarial problem*"(Ardestani-Jaafari and Delage 2016).

In a single-stage robust optimization problem, constraints must be satisfied for all possible realizations of uncertain parameters. Therefore, by repeating constraints for different values of uncertain parameters, we can view a robust problem as a mathematical program with a large number of constraints. Depending on the structure of the uncertainty set, two techniques are usually applied to solve single-stage robust problems. The first approach is to iteratively generate violated constraints of the mathematical program explained above using a constraint generation algorithm (Fischetti and Monaci 2012, Bertsimas et al. 2016). In the second approach, the problem is reformulated as its deterministic robust counterpart and then solved directly. Soyster (1973) presented such a deterministic counterpart model for robust linear problems with box uncertainty sets. Ben-Tal and Nemirovski (1999) proposed a second order cone program for uncertain linear programs with ellipsoidal uncertainty sets. They also showed that in the case of polyhedral uncertainty sets the robust counterpart model is a linear program. Bertsimas and Sim (2004) showed that robust linear programs with Γ-cardinality uncertainty sets can be reformulated as deterministic linear programs.

Multi-stage robust problems are more complicated than single-stage robust problems and are generally intractable (Ben-Tal et al. 2004). There are two common solution approaches for these problems. Both approaches transform the multi-stage problem to a single-stage problem and then apply the solution methods of the single-stage robust problem. In the first approach the recourse decisions are restricted to a function of uncertain parameters resulting in a single-stage robust problem. In this context, affine adaptability, also referred to as linear decision rules, assumes recourse decisions to be affine functions of uncertain parameters. This method is very popular and is applied in various areas such as supply chain management (Ben-Tal et al. 2005), inventory control (Ben-Tal et al. 2009), portfolio management (Fonseca and Rustem 2012), warehouse management (Ang et al. 2012), capacity management (Ouorou 2013) and network design (Poss and Raack 2013). Chen and Zhang (2009) introduced the extended affine adaptability by re-parameterizing the primitive parameters and then applying the affine adaptability. Bertsimas et al. (2011) proposed a more accurate approximation of recourse decisions using polynomial adaptability. A drawback of the functional adaptability is its inability to handle problems with integer recourse decisions. Another approach is finite adaptability in which the uncertainty set is split into a number of smaller subsets, each with its own set of recourse decisions. The number of these subsets can be either fixed a priori or decided by the optimization model (Vayanos et al. 2011, Bertsimas and Caramanis 2010, Hanasusanto et al. 2014, Postek and Den Hertog 2014, Bertsimas and Dunning 2016). An important advantage of the finite adaptability is that, in contrast to the functional adaptability approach, it easily handles problems with integer recourse variables.

In the literature, convex uncertainty sets are used to model robust problems. The main advantage of these uncertainty sets is that they can be simply formulated by continuous variables and the problem remains tractable in many cases such as linear programs. However, it is sometimes inevitable or desirable to use integer variables to formulate the uncertainty set. Nguyen and Lo (2012) studied a single-stage robust portfolio problem where the weights of portfolios are fixed such that a generic objective function is optimized for the worst possible ranking of portfolios. Thus, in this application it is necessary to use integer variables to formulate the ranking of portfolios. Feige et al. (2007) and Gupta et al. (2014) also studied some classical covering problems where in their adversarial problems, integer variables were used to choose a set of active clients in a graph. Moreover, in some cases integer variables are used to approximate non-convex uncertainty sets. For instance, Siddiq (2013) and Chan et al. (2015) studied a robust facility location problem and discussed how non-convex uncertainty sets can be approximated by discretization.

There are many papers in the literature that have proposed Benders algorithms to solve two-stage robust optimization problems (Jiang et al. 2012, Zheng et al. 2012, Bertsimas et al. 2013, Jiang et al. 2013, Zhao and Guan 2013, Remli and Rekik 2013, Zhang et al. 2015, Dashti et al. 2016). In these papers, assuming that the problem is set as a $min(max(min(.)))$ problem, the authors have dualized the inner minimization to reformulate the problem to a $min(max(.))$ problem with bilinear terms in the objective function. Then, they have applied a Benders algorithm to solve the first-stage problem together with cuts generated from an outer approximation algorithm which solves the maximization problem.

Column-and-constraint generation algorithm is another common exact approach to solve a two-stage robust optimization problem (Zeng and Zhao 2013, Zhao and Zeng 2012, An et al. 2014, An and Zeng 2015, Danandeh et al. 2014, Ding et al. 2016, Wang et al. 2014b, Lee et al. 2014, 2015, Li et al. 2015, 2017, Chen et al. 2016, Wang et al. 2016a). The underlying idea of this approach is to make copies of recourse decision variables and also second-stage constraints for each possible realization of uncertain parameters which results in a large-scale mixed-integer programming model. As it is impossible to solve this model directly, a column-and-constraint generation algorithm is essential to generate critical uncertain scenarios and their corresponding recourse decision variables and second-stage constraints.

In this work we assume that the uncertainty appears on the right-hand side values and the corresponding technology matrix of recourse decision variables has a block-diagonal structure. The main difference of our approach compared to the previous exact approaches for two-stage robust optimization models is that we apply Dantzig-Wolfe decomposition on the vector of uncertain right hand-side values in order to exploit the block-diagonal structure in the technology matrix of recourse decision variables, and reformulate the two-stage problem as a single-stage problem. This single-stage problem has a different structure compared to the single-stage problem which is obtained by dualizing the second-stage problem as explained previously. As a result, when applying the Benders algorithm, we do not have any bilinear term in the objective function of the subproblem. Moreover, in contrast to the reformulations technique which dualizes the inner minimization problem, our proposed reformulation is applicable when recourse decision variables are integer. Furthermore, the single-stage problem obtained from our reformulation has much fewer copies of recourse variables and second-stage constraints compared to the case where the two-stage problem is directly formulated as a single-stage problem by making copies of recourse variables and second-stage constraints for each possible realization of uncertain parameters. Therefore, in some applications, such as the nurse planning problem studied in this work, we can apply a Benders algorithm on the single-stage problem resulted form the proposed reformulation approach without any need to use a column-and-constraint generation approach. The reformulation approach that we propose is inspired from the one proposed in Siddiq (2013) that presented a reformulation for a specific facility location problem. The advantage of our reformulation is that it is more general and applicable to any two-stage robust problem with block-diagonal structure in the technology matrix of recourse decision variables.

The main contribution of our work is a novel reformulation and some solution methods for a class of two-stage robust problems with integer adversarial variables. We discuss the importance of this class of robust models in two-stage robust resource planning problems where some tasks with uncertain arrivals and durations are expected. We apply Dantzig-Wolfe decomposition to exploit the block-diagonal structure in the technology matrix of recourse decision variables. This decomposition reduces the original two-stage problem to a single-stage problem. We then develop a Benders algorithm for the reformulated problem. Since the master problem and subproblem in the Benders algorithm are mixed-integer programs, it is computationally expensive to solve them to optimality. Hence, we propose novel stopping conditions for these mixed integer programs and prove the convergence of the algorithm. We also develop a heuristic algorithm, namely dual algorithm, and combine it with the Benders algorithm to create a more effective algorithm called Benders-dual algorithm. We compare the computational performance of the proposed algorithms in a nurse planning application.

We organize the remainder of this paper as follows. In Section 6.2, we introduce the structure of the two-stage robust optimization problems studied in this paper and discuss its wide range applications in robust resource planning problems including a nurse planning problem.

In Section 6.3, we use Dantzig-Wolfe decomposition to reformulate the original two-stage robust problem as a single-stage robust problem and apply the proposed reformulated approach to our nurse planning problem. In Section 6.4, we develop solution methods for the reformulated problem, and provide extensive computational results on our nurse planning problem in Section 6.5. Finally we give some concluding remarks and future research directions in Section 6.6.

6.2 Model and applications

We study a class of two-stage robust optimization problems with the following structure.

$$\text{(P1)} \qquad \min_{x \in \mathcal{X}} \left(c_1^\intercal x + \max_{u \in \mathcal{U}} \left(\min_{y \in \mathcal{Y}} c_2^\intercal y \right) \right) \tag{6.1}$$

Subject to :

$$Ax + Bu + Cy \leq b \tag{6.2}$$

In the above formulation, x and y are the vector of decision variables in the first and the second stage respectively. u is the vector of uncertain parameters, also referred to as adversarial variables, that are restricted to the uncertainty set $\mathcal{U}$. c_1 and c_2 are given cost vectors, A, B and C are known matrices with appropriate dimensions and b denotes the known vector of right-hand side values. $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ represent the integrality and bound constraints that we may have for variables in the first and second stages. Objective (6.1) minimizes the sum of the first- and second-stage costs. In this model, $x \in \mathcal{X}$ must be selected such that for all realizations $u \in \mathcal{U}$ there is $y \in \mathcal{Y}$ such that constraint (6.2) is satisfied. We assume that $\mathcal{U}$ is a finite uncertainty set. This assumption is necessary for the convergence proofs of the proposed Benders algorithms discussed in Section 4.1. The first- and second-stage variables can be continuous, integer or mixed, but all adversarial variables are supposed to be integer. We also assume that C is a block-diagonal matrix. The main focus of this research is to exploit this block-diagonal structure and develop algorithms to solve the reformulated problem efficiently. In the rest of this paper we sometimes include constraints $x \in \mathcal{X}$, $u \in \mathcal{U}$ and $y \in \mathcal{Y}$ together with constraint (6.2). We also point out that in the literature of two-stage robust optimization the inner $max(min(.))$ problem in the above problem setting is sometimes referred to as the "*adversarial problem*".

Let us now explain how Model (P1) can fit a wide range of two-stage robust resource planning problems where a set of tasks denoted by $\mathcal{T}$ with uncertain arrivals and durations are expected within a given planning horizon. Let $\mathcal{D}$ denote the set of days in the planning horizon. To perform each task, a set of resources is needed. Let $\mathcal{R}$ denote the set of all resources. We suppose that the daily required amount of resource $r \in \mathcal{R}$ to perform task $t \in \mathcal{T}$, denoted by h_{tr}, is known and discrete. Also let $\mathcal{P}_t$ denote the set of all discrete scenarios for the uncertain arrival and duration of task t and let a_{tp} and l_{tp} be the arrival day and the duration of task t in scenario $p \in \mathcal{P}_t$ respectively. In the first stage, the decision maker must decide on the amount of resources made available over the planning horizon, paying a daily cost of c_{1r} per unit of resource $r \in \mathcal{R}$ over that period. In the adversarial problem, considering the first-stage decisions, the worst possible combination of scenarios about tasks is assumed to take place. Because of this worst-case realization, the amount of resource $r \in \mathcal{R}$ supplied for day $d \in \mathcal{D}$ in the first stage may not be enough. In this case, in the second stage, the decision maker needs to get some additional amounts of resource $r \in \mathcal{R}$ on day d to meet the demand level. In the second stage, the daily cost of resource $r \in \mathcal{R}$ is $c_{2r} > c_{1r}$ and there may be some additional restrictions on the maximum daily amounts of extra resources available.

The above problem can arise in various contexts such as project resource planning, maintenance planning and workforce planning where the durations of activities and their start times are uncertain. To formulate this problem as an example of Model (P1) we define the following variables and sets.

Variables:

x_{dr} : The amount of resource r supplied on day d in the first stage.

u_{tp} : 1 if scenario $p \in \mathcal{P}_t$ (previously explained) realizes for task t ; 0 otherwise. (adversarial variables)

y_{dr} : The additional amount of resource r supplied on day d in the second stage.

Sets:

$\mathcal{P}_{td}$: The subset of scenarios in $\mathcal{P}_t$ where task t is in process on day d, i.e., $\mathcal{P}_{td} = \{p \in \mathcal{P}_t : a_{tp} \leq d, a_{tp} + l_{tp} > d\}$ (note that a_{tp} and l_{tp} are defined above as the arrival day and the duration of task t in scenario $p \in \mathcal{P}_t$ respectively).

The two-stage robust resource planning problem is formulated as follows :

$$\min_{x \in \mathcal{X}} \left(\sum_{d \in \mathcal{D}} \sum_{r \in \mathcal{R}} c_{1r} x_{dr} + \max_u \left(\min_{y \in \mathcal{Y}} \sum_{d \in \mathcal{D}} \sum_{r \in \mathcal{R}} (c_{2r} y_{dr}) \right) \right) \tag{6.3}$$

Subject to :

$$x_{dr} + y_{dr} \geq \sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_{td}} h_{tr} u_{tp} \qquad r \in \mathcal{R}, d \in \mathcal{D} \tag{6.4}$$

$$\sum_{p \in \mathcal{P}_t} u_{tp} = 1 \qquad t \in \mathcal{T} \tag{6.5}$$

$$u_{tp} \in \{0, 1\} \qquad t \in \mathcal{T}, p \in \mathcal{P}_t \tag{6.6}$$

In the above model, constraint (6.4) corresponds to the daily demand constraints over the planning horizon and constraints (6.5) and (6.6) define the uncertainty set. $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ enforce non-negativity and integrality constraints of the first- and second-stage variables. In constraint (6.4) for each $(r, d) \in \mathcal{R} \times \mathcal{D}$ we have a block in the technology matrix corresponding to vector y.

As an illustrative instance of the problem defined above, we will concentrate in this paper on a nurse planning problem. In this problem, we plan wards' nurses of a hospital for a medium term. The daily workloads of nurses depend on the number of patients brought from operating rooms to wards. Patients are already scheduled in operating rooms over the planning horizon. Before transferring patients from operating rooms to wards they may stay in ICUs for some days. The lengths of stays in ICUs and wards are uncertain and discrete. For each patient a number of scenarios about the lengths of stays in ICUs and wards are available. In the first stage of this problem, we assign some nurses to wards over the planning horizon. In the second stage if the nurses' workload on a day is more than the service capacity of nurses assigned to that day, some extra nurses are hired. Nurses hired in the second-stage are paid more than those hired in the first-stage. The problem is formulated as follows :

Parameters:

c_1 :	The daily cost of a nurse hired in the first stage.
c_2 :	The daily cost of a nurse hired in the second stage.
M_d :	The maximum number of nurses available for hiring on day d in the second-stage.
δ :	The amount of service time provided by a first- or second-stage nurse per day (in hours).
ρ :	The average of required service time for each patient per day (in hours).
l_{tp}^{ICU} :	The length of stay in ICUs for patient t in scenario $p \in \mathcal{P}_t$.
l_{tp}^{Ward} :	The length of stay in wards for patient t in scenario $p \in \mathcal{P}_t$.
d'_t :	The surgery day for patient t.

Set:

$\mathcal{D}$:	The set of days in the planning horizon.
$\mathcal{T}$:	The set of patients (tasks) already scheduled in operating rooms over the planning horizon.
$\mathcal{T}_d$:	The set of patients scheduled on day d.
$\mathcal{P}_t$:	The set of scenarios for patient t. Each scenario gives information on the lengths of stays in ICUs and wards.

$\mathcal{P}_{td}$: The subset of scenarios in $\mathcal{P}_t$ where patient t is in wards on day d, i.e., $\mathcal{P}_{td} = \left\{p \in \mathcal{P}_t : d'_t \leq d, d'_t + l_{tp}^{ICU} + l_{tp}^{Ward} > d\right\}$.

Variables:

x_d : The number of nurses assigned to day d in the first stage.

u_{tp} : 1 if patient t follows scenario p after its surgery, 0 otherwise. (adversarial variable)

y_d : The number of nurses hired on day d in the second stage.

$$\min_x \left(\sum_{d \in \mathcal{D}} c_1 x_d + \max_u \left(\min_y \sum_{d \in \mathcal{D}} (c_2 y_d) \right) \right) \tag{6.7}$$

Subject to :

$$\delta x_d + \delta y_d \geq \rho \sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_{td}} u_{tp} \qquad d \in \mathcal{D} \tag{6.8}$$

$$x_d \geq 0, \text{integer} \qquad d \in \mathcal{D} \tag{6.9}$$

$$\sum_{p \in \mathcal{P}_t} u_{tp} = 1 \qquad t \in \mathcal{T} \tag{6.10}$$

$$u_{tp} \in \{0, 1\} \qquad t \in \mathcal{T}, p \in \mathcal{P}_t \tag{6.11}$$

$$0 \leq y_d \leq M_d, \text{integer} \qquad d \in \mathcal{D} \tag{6.12}$$

Constraint (6.8) is the daily demand constraints over the planning horizon. In this constraint, the reason that we have used coefficient ρ as the average workload of each patient rather than the actual workloads is that we do not know patients' workloads in advance and therefore we cannot adjust the level of second-stage nurses accordingly in the morning. Constraint (6.10) and (6.11) define the discrete uncertainty set. Constraints (6.9) and (6.12) represent the bounds and integrality constraints for first- and second-stage variables respectively. In the next section, we use the nurse planning problem to illustrate the proposed reformulation. We also present extensive computational results on this problem in Section 6.5.

6.3 Reformulation

In this section we propose a reformulation of Model (P1) and use it to develop solution methods in Section 6.4. For this we need the following additional notation, used throughout the rest of the paper.

$\mathcal{K}$: The index set of blocks in matrix C.

C_k : The k-th block in matrix C.

Row_k : The number of rows in block C_k.

Col_k : The number of columns in block C_k.

y_k : The subset of variables y involved in block C_k.

$\mathcal{Y}_k$: The set of integrality and bound constraints corresponding to variables y_k.

c_{2k} : The subset of c_2 corresponding to variables y_k.

b_k : The right-hand side values in front of block C_k.

A_k : The rows in matrix A in front of block C_k.

B_k : The rows in matrix B in front of block C_k.

With respect to the block-diagonal structure of matrix C we can rewrite constraint (6.2) as follows.

$$A_k x + B_k u + C_k y_k \leq b_k \qquad k \in \mathcal{K} \tag{6.13}$$

Furthermore, we define some notation related to $B_k u$ in (6.13).

$\mathcal{S}'_k$: The set of all realizations for $B_k u$, i.e., $\mathcal{S}'_k = \{v \in \mathbb{R}^{Row_k} | v = B_k u, u \in \mathcal{U}\}$.

$\mathcal{S}_k$: The index set of $\mathcal{S}'_k$, i.e., $\mathcal{S}_k = \{1, 2, ..., |\mathcal{S}'_k|\}$.

e_{ks} : The s-th member of $\mathcal{S}'_k$ (defined for $s \in \mathcal{S}_k$).

w_{ks} : Is a binary variable and takes 1 if $B_k u$ is equal to e_{ks}, 0 otherwise.

Using all the above notation, we reformulate Model (P1) as follows.

$$\text{(P2)} \qquad \min_x \left(c_1^\intercal x + \max_{u,w} \left(\min_y \sum_{k \in \mathcal{K}} c_{2k}^\intercal y_k \right) \right) \tag{6.14}$$

Subject to :

$$A_k x + \sum_{s \in \mathcal{S}_k} e_{ks} w_{ks} + C_k y_k \leq b_k \qquad k \in \mathcal{K} \tag{6.15}$$

$$x \in \mathcal{X} \tag{6.16}$$

$$\sum_{s \in S_k} w_{ks} = 1 \qquad k \in \mathcal{K} \tag{6.17}$$

$$B_k u = \sum_{s \in \mathcal{S}_k} e_{ks} w_{ks} \qquad k \in \mathcal{K} \tag{6.18}$$

$$w_{ks} \in \{0, 1\} \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{6.19}$$

$$u \in \mathcal{U} \tag{6.20}$$

$$y_k \in \mathcal{Y}_k \qquad k \in \mathcal{K} \tag{6.21}$$

In the following we introduce a new model that is equivalent to Model (P2) as we will show later in Lemma 6.1 and Theorem 6.1. To introduce Model (P3), for each $k \in \mathcal{K}$ we make $|\mathcal{S}_k|$ copies of variables $y_k \in \mathbb{R}^{Col_k}$ and define variables $y'_{ks} \in \mathbb{R}^{Col_k}(s \in \mathcal{S}_k)$. Model (P3) is given by (6.22)-(6.29).

$$\text{(P3)} \qquad \min_{x} \left(c_1^\intercal x + \max_{u,w} \left(\min_{y'} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal y'_{ks} w_{ks} \right) \right) \tag{6.22}$$

Subject to :

$$A_k x + e_{ks} + C_k y'_{ks} \leq b_k \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{6.23}$$

$$x \in \mathcal{X} \tag{6.24}$$

$$\sum_{s \in S_k} w_{ks} = 1 \qquad k \in \mathcal{K} \tag{6.25}$$

$$B_k u = \sum_{s \in S_k} e_{ks} w_{ks} \qquad k \in \mathcal{K} \tag{6.26}$$

$$w_{ks} \in \{0, 1\} \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{6.27}$$

$$u \in \mathcal{U} \tag{6.28}$$

$$y'_{ks} \in \mathcal{Y}'_k \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{6.29}$$

The structure of Model (P3) is such that, if w_{ks} takes 1, y'_{ks} is equal to the optimal solution of y_k in Model (P2). Indeed by introducing constraint (6.23) we have made $|\mathcal{S}_k|$ copies of constraint (6.15) to compute the values of y'_{ks} independently. Moreover, to ensure that the optimal objective values of models (P2) and (P3) are the same, $c_{2k}^\intercal y'_{ks}$ in (6.22) is multiplied by w_{ks}.

Lemma 6.1 *Suppose that Model (P2) is feasible. Then $\hat{x}$ is a first-stage feasible solution of Model (P2) if and only if it is a first-stage feasible solution of Model (P3). Moreover, the objective values of models (P2) and (P3) for the first-stage solution $\hat{x}$ are the same if $\max\limits_{u,w}$ and $\min\limits_{y'}$ are solved optimally. Furthermore, for this first-stage solution the optimal values of variables y'_{ks} in Model (P3) represent the second-stage optimal policies in Model (P2).*

Proof. Appendix C.1. $\square$

The following theorem states the relations between models (P2) and (P3).

Theorem 6.1 *Models (P2) and (P3) are equivalent, that is either*

- *both models are unbounded, or*
- *both models are infeasible, or*

- *both models are feasible and bounded with the same optimal objective value and the same optimal solution for the first-stage variables. In this case the optimal solution of variables y'_{ks} in Model (P3) represents the optimal policies for variables y_k in Model (P2).*

PROOF. Lemma 6.1 directly results in cases 1 and 3. To prove case 2, we note that with respect to Lemma 6.1 for any feasible solution in Model (P2) there is an equivalent feasible solution in Model (P3). Therefore, Model (P3) is infeasible if and only if Model (P2) is infeasible. □

The next theorem shows that Model (P3) can be reduced to a single-stage problem.

THEOREM 6.2 *In Model (P3) the objective function* $\max_{u,w}\left(\min_{y'}(.)\right)$ *can be replaced by* $\min_{y'}\left(\max_{u,w}(.)\right)$.

PROOF. Appendix C.2. □

Therefore, we can rewrite Model (P3) as follows :

$$\text{(P4)} \qquad \min_{x,y'}\left(c_1^\intercal x + \max_{u,w}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} c_{2k}^\intercal y'_{ks} w_{ks}\right)\right) \tag{6.30}$$

$$(6.23) - (6.29)$$

In fact, the reformulation presented in this section shows that we can transform the two-stage robust problem (P1) to a single-stage robust problem. The other interesting point about Model (P4) is that the solution spaces of variables (x, y') and (u, w) are independent. Therefore, we can introduce Model (P5) as follows.

$$\text{(P5)} \qquad \min_{(x,y')\in(\mathcal{X},\mathcal{Y}')}\left(c_1^\intercal x + \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} c_{2k}^\intercal y'_{ks} w_{ks}\right)\right) \tag{6.31}$$

In the above model, $(\mathcal{X},\mathcal{Y}') = \{(x,y')|\text{constraints (6.23)-(6.24) and (6.29) are satisfied}\}$ and $(\mathcal{U},\mathcal{W}) = \{(u,w)|\text{constraints (6.25)-(6.28) are satisfied}\}$. We use the latter model to present our solution methods in Section 6.4. In the rest of this paper, we refer to the inner $max(.)$ in (6.31) as the adversarial problem.

In the following, we give the corresponding nurse planning problem reformulation in the form of Model (P5). The definitions of variables x_d and u_{ip} from the nurse planning problem remain unchanged.

New set:

$\mathcal{S}_d$: The set of all possible realizations for the number of patients in wards on day d.

New variables:

w_{ds} : 1 if exactly s patients are in wards on day d, 0 otherwise.

y'_{ds} : The number of nurses hired on day d in the second stage if exactly s patients are in wards on this day.

$$\min_{x,y'} \left(\sum_{d \in \mathcal{D}} c_1 x_d + \max_{u,w} \left(\sum_{d \in \mathcal{D}} \sum_{s \in \mathcal{S}_d} c_2 w_{ds} y'_{ds} \right) \right) \tag{6.32}$$

Subject to :

$$\delta x_d + \delta y_{ds} \geq \rho \times s \qquad d \in \mathcal{D}, s \in \mathcal{S}_d \tag{6.33}$$

$$x_d \geq 0, \text{integer} \qquad d \in \mathcal{D} \tag{6.34}$$

$$\sum_{s \in \mathcal{S}_d} w_{ds} = 1 \qquad d \in \mathcal{D} \tag{6.35}$$

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_{td}} u_{tp} = \sum_{s \in \mathcal{S}_d} s w_{ds} \qquad d \in \mathcal{D} \tag{6.36}$$

$$\sum_{p \in \mathcal{P}_t} u_{tp} = 1 \qquad t \in \mathcal{T} \tag{6.37}$$

$$w_{ds} \in \{0,1\} \qquad d \in \mathcal{D}, s \in \mathcal{S}_d \tag{6.38}$$

$$u_{tp} \in \{0,1\} \qquad t \in \mathcal{T}, p \in \mathcal{P}_t \tag{6.39}$$

$$0 \leq y_{ds} \leq M_d, \text{integer} \qquad d \in \mathcal{D}, s \in S_d \tag{6.40}$$

6.4 Solution methods

In this section we propose three solution methods for Model (P5). We present a Benders algorithm that iterates between a master problem and a subproblem to tighten the optimality gap. We also propose a heuristic, called dual algorithm. In this heuristic, we dualize the linear programming relaxation of the inner max problem in Model (P5). Then we iteratively generate some cuts to shape the convex hull of the uncertainty set. We also present a Benders-dual algorithm that applies the dual heuristic within the framework of the Benders algorithm.

6.4.1 Benders Algorithm

In our Benders algorithm, valid lower and upper bounds are obtained by solving the master problem and the subproblem respectively. The algorithm iterates between these problems until the bounds converge. In the following we present the master problem and subproblem. Then we explain the framework of the Benders algorithm.

Suppose that m adversarial scenarios $(\hat{u}^j, \hat{w}^j) \in (\mathcal{U}, \mathcal{W}), j = 1, 2, ..., m$ are available. In our Benders algorithm, we iteratively detect new scenarios and update m. We define the master problem of the Benders algorithm as follows.

$$\text{(MP)} \qquad \min_{(x,y') \in (\mathcal{X}, \mathcal{Y}'), \theta} \theta \tag{6.41}$$

Subject to :

$$\theta \geq c_1^T x + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^{\intercal} y'_{ks} \hat{w}_{ks}^j \qquad j = 1, 2, ..., m \tag{6.42}$$

Theorem 6.3 *The optimal objective value of Model (MP) is a valid lower bound for Model (P5).*

Proof. Appendix EC.3. □

For a feasible solution $(\hat{x}', \hat{y}') \in (\mathcal{X}, \mathcal{Y}')$ a valid upper bound is obtained by solving the inner max problem in (6.31). We refer to the following problem as the subproblem of the Benders algorithm.

$$\text{(SP)} \qquad \max_{(u,w) \in (\mathcal{U}, \mathcal{W})} \left(c_1^{\intercal} \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^{\intercal} \hat{y}'_{ks} w_{ks} \right) \tag{6.43}$$

Algorithm 6.1 provides the pseudo code of the Benders algorithm. In this algorithm, UB and LB respectively denote the best upper and lower bounds found during the algorithm. Also m denotes the number of adversarial scenarios already generated and included as an instance of constraint (6.42) in the master problem. In Line 3, we obtain an initial solution $(\hat{x}, \hat{y}')$ by a heuristic algorithm that is explained at the end of Section 6.4.2. Lines 4 to 9 represent the main loop of the Benders algorithm. In Line 5, we use solution $(\hat{x}, \hat{y}')$ to modify the objective function of the subproblem. Then we solve the subproblem in Line 6 to obtain an adversarial scenario $(\hat{u}, \hat{w})$. The algorithm updates UB if the optimal objective value of the subproblem is less than the current UB. In Line 7, we add a new optimality cut (6.42) to the master problem. This cut corresponds to the new adversarial scenario found by solving the subproblem. In the next step, the algorithm solves the master problem to obtain a new feasible solution $(\hat{x}, \hat{y}')$ and updates LB. As stated before the optimal objective value of the master problem is a valid lower bound for the original robust problem. Since we add a new instance of constraint (6.42) in each iteration of the algorithm to the master problem, the lower bound obtained from the master problem is non-decreasing during the algorithm.

The stopping conditions of the Benders algorithm are then checked in Line 9. $\delta_{acc}^{Benders}$ and $AlgTimeLimit$ respectively denote the maximum acceptable optimality gap and the available computational time.

Algorithm 6.1. Benders algorithm

```
1: Input parameters : $\delta_{acc}^{Benders}$ and $AlgTimeLimit$.
2: Set $UB$=$\infty$, $LB$=-$\infty$, and $m = 0$.
3: Find an initial solution $(\hat{x}, \hat{y}')$ by a heuristic.
4: repeat
5:     Modify the objective function of subproblem (SP) using $(\hat{x}, \hat{y}')$.
6:     Solve the subproblem and update $UB$ if it is necessary.
7:     Add a new optimality cut (42) to the master problem and set $m = m + 1$.
8:     Solve the master problem and update $LB$.
9: until $(100(UB - LB)/LB \leq \delta_{acc}^{Benders}$ or time limit $AlgTimeLimit$ is reached)
```

The main shortcoming of the Benders algorithm is that the master problem and subproblem are mixed integer programs (MIPs) and therefore it would be very time consuming to optimally solve them in all iterations of the algorithm. In the following we present novel stopping conditions for these MIPs. Before explaining these conditions we define "ε-dominant incumbents" for the master problem and subproblem as follows : For a constant $\varepsilon > 0$, an ε-dominant incumbent of the master problem is a feasible solution in the master problem with an objective value that is less than the lower bound for the recent subproblem by a margin of ε. Similarly, an ε-dominant incumbent of the subproblem is a feasible solution in the subproblem with an objective value that is at least ε higher than the upper bound of the recent master problem. The stopping conditions are presented as follows.

Stopping condition for the master problem (subproblem) : The mixed integer program terminates when the optimal solution is found or at least $Time_{MP}$ seconds ($Time_{SP}$ seconds) has passed from the moment that the first ε-dominant incumbent of the master problem (subproblem) is found.

In Table 6.1, we present a numerical example with $\varepsilon = 5$ to explain this stopping condition for both the master problem and the subproblem. In this table, the results of the master problem and of the subproblem are presented in separate columns. We report the lower and upper bounds for the related mixed integer programs. Since these MIPs are not optimally solved there are gaps between the lower and upper bounds. Columns "Order"also give the order in which these MIPs are solved. In this example, in iterations 1 to 4, the upper bound of the master problem is at least $\varepsilon = 5$ units less than the lower bound for the previous subproblem. Moreover, in iterations 2 to 4, the lower bound for the subproblem is at least $\varepsilon = 5$ units higher than the upper bound of the master problem in the previous iteration. In Iteration 5, when solving the subproblem, we observe that the lower bound does not increase to $\varepsilon = 5$ units higher than the upper bound of the master problem in Iteration 4. Therefore, the stopping condition is not met and the subproblem has to be solved optimally. Similarly, in the same iteration, when we solve the master problem, the upper bound does not decrease to $\varepsilon = 5$ units less than the lower bound for the subproblem in that iteration. Thus, the stopping condition is not satisfied and the master problem has to be solved to optimality.

Table 6.1 – A numerical example to explain the stopping conditions of MIPs in the Benders algorithm.

Iteration	**Subproblem**			**Master problem**		
	Order	**LB**	**UB**	**Order**	**LB**	**UB**
1	1	**120**	470	2	45	**110**
2	3	**340**	650	4	30	**300**
3	5	**320**	360	6	155	**165**
4	7	**170**	185	8	157	**160**
5	9	**160**	160	10	160	**160**

The upper bound in the master problem and the lower bound in the subproblem correspond to feasible solutions of MIPs. Therefore, the stopping condition for the subproblem means that the subproblem terminates before reaching the optimality if we find a critical adversarial scenario. We refer to an adversarial scenario as a critical one if by adding its corresponding cut (6.42) to the master problem, the objective value of solution $(\hat{x}, \hat{y}')$ found in the previous iteration, increases by at least ε units. Lower bounds in the master problem and the upper bounds in the subproblem are valid lower and upper bounds for the original robust problem respectively. Since we do not solve the master problem optimally the lower bounds obtained from the master problem can be decreasing. As an example, in Table 1 we can observe that the lower bound for the master problem decreases from 45 in the first iteration to 30 in the second iteration. However, we know that the lower bound for the master problem in an early iteration is also a valid lower bound for the master problem in next iterations because optimality cuts (6.42) are added incrementally to the master problem. Therefore, we can impose the best lower bound LB by adding the following constraint to the master problem.

$$\theta \geq LB \tag{6.44}$$

Moreover, the best upper bound UB updated during the Benders algorithm can be imposed to the master problem using the following constraint.

$$\theta \leq UB \tag{6.45}$$

Constraint (6.45) is valid because the optimal objective value of the subproblem is an upper bound on the optimal objective value of the original robust problem. As it will be discussed later, constraints (6.44)-(6.45) are vital for proving the convergence of the Benders algorithm with stopping conditions for the master problem and subproblem. We cannot impose constraints similar to constraints (6.44)-(6.45) to confine lower and upper bounds for the subproblem because the coefficients in the objective function of the subproblem are modified during the algorithm.

When we apply the stopping conditions, most of the time the subproblem is not solved optimally. Therefore, the best upper bound obtained by Algorithm 6.1 is poor if the algorithm times out. In this case, we call Procedure 6.1 at the end of Algorithm 6.1 to improve the quality of the best upper bound. This procedure sorts all solutions $(\hat{x}, \hat{y}')$ found by the master problem based on their upper bounds. The upper bound of each solution $(\hat{x}, \hat{y}')$ is the upper bound of its corresponding subproblem obtained in Line 6 of Algorithm 6.1. Procedure 6.1 evaluates these solutions separately by solving the subproblem without any stopping condition. When solving a subproblem if we obtain a feasible solution with an objective value higher than the best upper bound UB, the subproblem terminates and Procedure 6.1 evaluates the next solution $(\hat{x}, \hat{y}')$ in the sorted list. This is because in this case, another solution with a better upper bound is already known. We consider a time limit $EvaTimeLimit$ for this procedure.

Procedure 6.1. Evaluation of the generated solutions $(\hat{x}, \hat{y}')$

```
Input parameters : $EvaTimeLimit$ and $\delta_{acc}^{Benders}$.
if ($100(UB - LB)/LB > \delta_{acc}^{Benders}$) then
  Sort solutions $(\hat{x}, \hat{y}')$ in the solution pool.
  for ($i$=1 to $NumberSolutions$) do
    Solve the subproblem for $i$-th solution $(\hat{x}, \hat{y}')$ and update $UB$ if necessary.
    if ($100(UB - LB)/LB \leq \delta_{acc}^{Benders}$ or $EvaTimeLimit$ is reached) then
      break ;
    end if
  end for
end if
```

In the following we discuss the convergence of the Benders algorithm with and without stopping conditions for the subproblem and master problem. We use the following notation to present the next lemmas and theorems.

$\mathcal{W}$:	The set of vectors w for which there is $u \in \mathcal{U}$ such that $(u, w) \in (\mathcal{U}, \mathcal{W})$.
n :	The number of adversarial scenarios in $(\mathcal{U}, \mathcal{W})$.
n' :	The number of unique vectors w that the algorithm visits the subproblem before it converges.
n'' :	The number of times that the algorithm visits an already encountered vector w before it converges.
ε :	A positive constant used in stopping conditions of the master problem and subproblem.
Opt :	The optimal objective value of the original robust problem.
O_i^{SP} :	The optimal objective value of the subproblem in iteration i.
U_i^{MP} :	The upper bound of the master problem in iteration i.
$f(j)$:	The iteration in which for the j-th times the algorithm generates an adversarial scenario with a new vector w in the subproblem.
$g(i)$:	The iteration in which for the i-th times the algorithm re-visits any of the generated vectors w in the subproblem.
I_i :	An indicator that is equal to 1 if in iteration i the algorithm generates an adversarial scenario with a repeated vector w, 0 otherwise.

THEOREM 6.4 *The Benders algorithm without stopping conditions for the master problem and subproblem converges in at most $|\mathcal{W}| + 1$ iterations that is bounded above by $n + 1$ iterations.*

PROOF. Appendix C.4. □

The following lemmas are used in the proof of Theorem 6.5.

LEMMA 6.2 *In the Benders algorithm with stopping conditions for the master problem and subproblem, if the algorithm finds an adversarial scenario with a repeated vector w in the subproblem of iteration i, then it is the optimal solution of the subproblem and the optimal objective value of the subproblem is equal to the upper bound of the recent master problem in iteration $i - 1$, i.e. $U_{i-1}^{MP} = O_i^{SP}$.*

PROOF. Appendix C.5. This lemma is used in the proofs of lemmas 6.3, 6.4 and 6.5. □

LEMMA 6.3 *In the Benders algorithm with the stopping conditions for the master problem and subproblem, if the algorithm finds an adversarial scenario with a repeated vector w in the subproblem of iteration i and $O_i^{SP} - Opt > \varepsilon$ holds, then in at most $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations either the algorithm finds an adversarial scenario with a new vector w or $O_{i+k}^{SP} - Opt \leq \varepsilon$ holds.*

PROOF. Appendix C.6. This lemma is used in the proof of Theorem 6.5. □

LEMMA 6.4 *In the Benders algorithm with the stopping conditions for the master problem and subproblem, if the algorithm finds an adversarial scenario with a repeated vector w in the subproblem of iteration i and $O_i^{SP} - Opt \leq \varepsilon$ holds, then in the next iteration either the Benders algorithm converges or an adversarial scenario with a new vector w is found.*

PROOF. Appendix C.7. This lemma is used in the proof of Theorem 6.5. □

LEMMA 6.5 *In the Benders algorithm with the stopping conditions, relation $O_{g(i_1)}^{SP} \geq O_{g(i_2)}^{SP}$ holds for any integer numbers i_1 and i_2 satisfying $1 \leq i_1 < i_2 \leq n''$.*

PROOF. Appendix C.8. This lemma is used in the proof of Theorem 6.5. □

THEOREM 6.5 *The Benders algorithm with the stopping conditions converges in at most $\sum_{j=1}^{n'} (1 + (\lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor + 1) I_{f(j)+1})$ iterations that is bounded above by $|\mathcal{W}|(\lfloor (O_{g(1)}^{SP} - Opt)/\varepsilon \rfloor + 2)$ iterations.*

PROOF. Appendix C.9. □

6.4.2 Dual Algorithm

In this section we present a heuristic algorithm. This algorithm dualizes the linear programming relaxation of the inner max problem in Model (P5) to transform the min-max problem to a single minimization problem. Let's assume that constraints forming the convex hull of the inner max problem in Model (P5) are as follows.

$$Du + Ew \leq b_2 \tag{6.46}$$

In constraint (6.46), u and w are the vectors of adversarial variables, D and E are technology matrices with appropriate dimensions and b_2 is the known vector of right-hand side values. D, E and b_2 are independent from the values of $(\hat{x}, \hat{y}')$ that are fixed in the outer min problem in Model (P5). This is because the solution space of adversarial variables does not depend on the variables in the outer min problem. If we have constraints (6.46) we can replace constraints (6.25)- (6.28) with them. In this case, the inner max problem is a linear programming model for fixed values of $(\hat{x}, \hat{y}')$ in the outer min problem. Therefore by dualizing the inner max problem we obtain the following model.

$$\text{(D-P5)} \qquad \min_{\hat{x},\hat{y}',\gamma} (c_1^\intercal x + b_2^\intercal \gamma) \tag{6.47}$$

Subject to :

$$(23) - (24), (29)$$

$$E_{ks}^\intercal \gamma \geq c_{2k}^\intercal y'_{ks} \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{6.48}$$

$$D^\intercal \gamma = 0 \tag{6.49}$$

$$\gamma \geq 0 \tag{6.50}$$

In the above model, γ is the vector of dual variables for constraint (6.46) and E_{ks} is the column in E that includes coefficients of variable w_{ks}. We can observe that the min-max problem in Model (P5) reduces to a single min problem and can be solved directly as a mixed-integer programming model. In the literature, the above dualization technique is prevalent to simplify single-stage robust problems where the inner max problem is a linear programming model. However, in our model, the inner max problem is a mixed integer program and constraints (6.46) forming the convex hull of the uncertainty set are unknown. In the following we present a heuristic algorithm that relaxes the integrality constraints of the variables in the inner max problem of Model (P5). Then by iteratively generating some cuts, it attempts to shape the solution space of the relaxed inner max problem into its convex hull before the relaxation. To present this heuristic we first need to define the following two models (P6) and (P7).

$$\text{(P6)} \qquad \min_{x,y'} \left(c_1^\intercal x + \max_{u,w} \left(\sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} c_{2k}^\intercal y'_{ks} w_{ks} \right) \right) \tag{6.51}$$

Subject to :

$$(23) - (29)$$

$$Fu + Gw \leq b_3 \tag{6.52}$$

In Model (P6), constraint (6.52) is the set of valid cuts that the heuristic algorithm generates iteratively. This constraint set is empty at the beginning of the algorithm. In this constraint, F and G are technology matrices with appropriate dimensions and b_3 the known vector of right-hand side values. We obtain the following Model (P7) by relaxing the integrality constraints of variables y and w in Model (P6) and then dualizing the inner max problem.

$$\text{(P7)} \qquad \min_{x,y',\pi,\lambda,\alpha,\beta} \left(c_1^\intercal x + b_3^\intercal \pi + b_4^\intercal \lambda + \sum_{k\in\mathcal{K}} \alpha_k \right) \tag{6.53}$$

Subject to :

$$(6.23) - (6.24), (6.29)$$

$$B_k^\intercal \beta_k + H^\intercal \lambda + F^\intercal \pi = 0 \tag{6.54}$$

$$\alpha_k + e_{ks}^\intercal \beta_k + G_{ks}^\intercal \pi \geq c_{2k}^\intercal y'_{ks} \qquad k \in \mathcal{K}, s \in \mathcal{S}_k \tag{6.55}$$

In Model (P7), β_k and π are vectors of dual variables for constraints (6.26) and (6.52) respectively, α_k is the dual variable of constraint (6.25) defined for each $k \in \mathcal{K}$ and G_{ks} is the column in G that includes coefficients of variable w_{ks}. To write the dual of the inner max problem in Model (P6) we have supposed that linear constraints hidden in uncertainty set $\mathcal{U}$ in constraint (6.28) are represented by $Hu \leq b_4$. In Model (P7), λ denotes the vector of dual variables for $Hu \leq b_4$.

Algorithm 6.2 provides the pseudo code of the dual algorithm. In Line 2, we suppose that no instance of constraint (6.52) is available at the beginning of the algorithm and F, G and b_3 are empty. "*Ite*"is the iteration counter of the loop starting in Line 3. In Line 5, we solve Model (P7) to obtain a feasible solution for $(\hat{x}, \hat{y}')$. For a fixed solution $(\hat{x}, \hat{y}')$ in Model (P6), the inner max problem is an integer program and we denote it by InnerMax$(\hat{x}, \hat{y}')$. In Line 7, we call Procedure 6.2. In each iteration of this procedure, we solve the linear programming relaxation of InnerMax$(\hat{x}, \hat{y}')$ and obtain a new fractional scenario $(\hat{u}, \hat{w})$. Then this procedure generates some valid cuts to remove this fractional scenario. This procedure continues until it cannot detect any other violated cut or time limit *AlgTimeLimit* is reached. We use an integer programming solver to perform Procedure 6.2 and let it generate valid cuts as explained above. In calling the integer programming solver, we limit the maximum number of nodes to be explored in the branch and bound tree to one. In Line 8 in Algorithm 6.2, we extract the cuts generated by the integer programming solver and update F, G and b_3 in models (P6) and (P7). In Lines 9, the algorithm checks stopping criteria. One of these stopping criteria checks if the percentage of the objective value improvement obtained in the current iteration is less than or equal to parameter δ_{acc}^{Dual}.

The dual algorithm is a heuristic algorithm and does not necessarily find the optimal solution. Appendix C.10 presents an example to demonstrate that the dual algorithm does not guarantee the optimality. In the Benders algorithm presented by Algorithm 6.1, in Line 3 we solve Model (P7) with empty F, G and b_3 to find an initial solution $(\hat{x}, \hat{y}')$.

Algorithm 6.2. Dual algorithm

```
1: Input parameters : $LocalTimeLimit$ , $AlgTimeLimit$ and $\delta_{acc}^{Dual}$.
2: Set $Ite = 0$ and empty $F$, $G$ and $b_3$ in Models (P6) and (P7).
3: repeat
4:     $Ite++$.
5:     Solve Model (P7) with time limit $LocalTimeLimit$ to obtain a feasible solution $(\hat{x}, \hat{y}')$.
6:     Set $Obj_{Ite}$ equal to the objective value of Model (P7).
7:     Apply Procedure 6.2 to generate some cuts (6.52).
8:     Extract the generated cuts and update $F$, $G$ and $b_3$ in models (P6) and (P7).
9: until (No cut is generated in Line 7 in this iteration or time limit $AlgTimeLimit$ is
   reached or $(100(Obj_{Ite} - Obj_{Ite-1})/Obj_{Ite-1} \leq \delta_{acc}^{Dual})$)
```

Procedure 6.2. Cut generation for the dual algorithm

```
repeat
    Solve the linear programming relaxation of InnerMax$(\hat{x}, \hat{y}')$ to obtain $(\hat{u}, \hat{w})$.
    Detect some valid cuts to remove the fractional solution $(\hat{u}, \hat{w})$.
    Update $F$, $G$ and $b_3$ in InnerMax$(\hat{x}, \hat{y}')$.
until (No valid cut is generated or time limit $AlgTimeLimit$ is reached)
```

6.4.3 Benders-dual Algorithm

In this section, we combine the Benders and dual algorithms to create a more efficient algorithm. In this hybrid algorithm the Benders algorithm guarantees the convergence of the algorithm. The dual algorithm improves the overall efficiency by generating valid cuts for the adversarial problem and also by finding better solutions $(\hat{x}, \hat{y}')$ through solving Model (P7).

Algorithm 6.3 provides the pseudo code of the hybrid algorithm. In Line 3 of this algorithm, we obtain an initial solution $(\hat{x}, \hat{y}')$ by solving Model (P7) while F, G and b_3 are ignored. Lines 4 to 8 together with Line 20 are the same as the main loop of the Benders algorithm given in Algorithm 6.1. The algorithm finds an adversarial scenario by solving the subproblem in Line 6. We then add a new optimality cut to the master problem and solve it to find a new solution $(\hat{x}, \hat{y}')$. Then if time limit $WarmupTimeLimit$ is already reached, the algorithm enters an inner loop starting in Line 11. This loop is taken from the dual algorithm and improves the current solution $(\hat{x}, \hat{y}')$ by iteratively generating cuts (6.52) in Line 13 and then solving Model (P7) in Line 15. Then we check the stopping criteria of the dual algorithm in

Algorithm 6.3. Benders-Dual algorithm

```
1: Input parameters : $WarmupTimeLimit$, $AlgTimeLimit$, $LocalDualTimeLimit$, $EvaTimeLimit$, $\delta_{acc}^{Dual}$ and $\delta_{acc}^{Benders}$.
2: Set $UB = \infty$, $LB = -\infty$, $m = 0$ and empty $F$, $G$ and $b_3$ in models (P6) and (P7).
3: Find an initial solution $(\hat{x}, \hat{y}')$ by a heuristic.
4: repeat
5:     Modify the objective function of subproblem (SP) using solution $(\hat{x}, \hat{y}')$.
6:     Solve the subproblem with the proposed stopping condition and update $UB$ if it is necessary.
7:     Add a new optimality cut to the master problem and set $m = m + 1$.
8:     Solve the master problem with the proposed stopping condition, update $LB$ and save $(\hat{x}, \hat{y}')$ in the solution pool.
9:     if ($WarmupTimeLimit$ is reached) then
10:        Set $Ite = 0$.
11:        repeat
12:            $Ite + +$.
13:            Apply Procedure 6.2 using solution $(\hat{x}, \hat{y}')$ to generate some cuts (6.52).
14:            Extract the generated cuts and update $F$, $G$ and $b_3$ in models (P6) and (P7).
15:            Solve Model (P7) to obtain a solution $(\hat{x}, \hat{y}')$ and save it in the solution pool.
16:            Set $Obj_{Ite}$ to the objective value of Model (P7).
17:        until (No cut is generated in Line 14 or time limit $LocalDualTimeLimit$ is reached or $100(Obj_{Ite} - Obj_{Ite-1})/Obj_{Ite-1} \leq \delta_{acc}^{Dual}$)
18:        Choose the best solution $(\hat{x}, \hat{y}')$ from the solution pool.
19:    end if
20: until ($100(UB - LB)/LB \leq \delta_{acc}^{Benders}$ or $AlgTimeLimit$ is reached)
21: Apply Procedure 6.1 with parameters $EvaTimeLimit$, and $\delta_{acc}^{Benders}$ to improve $UB$.
```

Line 17. To check if time limit *LocalDualTimeLimit* is reached, the algorithm tracks the time from the start of the inner loop in Line 11. After leaving the inner loop in Line 17 and before starting a new iteration of the algorithm, we have to decide on the new solution $(\hat{x}, \hat{y}')$ to modify the objective function of the subproblem in Line 5. Therefore, during the algorithm we save all generated solutions $(\hat{x}, \hat{y}')$ in a solution pool. Then in Line 18 among all solutions in the pool, we choose the one with the lowest worst objective value against all generated adversarial scenarios as the current solution $(\hat{x}, \hat{y}')$. Similar to the Benders algorithm with stopping conditions for the master problem and the subproblem, we apply Procedure 6.1 at the end of the algorithm in order to improve the best upper bound UB. As explained before, this procedure evaluates solutions $(\hat{x}, \hat{y}')$ separately within a total time limit $EvaTimeLimit$. The other point about the Benders-dual algorithm is that time limit $WarmupTimeLimit$ is used in Line 9 in order to prevent from entering the inner loop in Line 11 before this time limit because on small instances the Benders algorithm converges very fast without any need of the dual algorithm.

There are generally two advantages for combining the Benders and dual algorithms. First, by generating cuts (6.52) in the dual algorithm and including them in the subproblem, we hope that the algorithm can solve the subproblem faster in next iterations. Also it is possible to improve the best solution $(\hat{x}, \hat{y}')$ by solving Model (P7) in Line 17 in Algorithm 6.3.

6.5 Computational results

In this section, we present extensive computational results for the nurse planning problem introduced in Section 6.2. We implemented all algorithms in C++ and used IBM ILOG CPLEX 12.6 to solve the mixed integer programs. We ran experiments on a computer with two Intel Xeon X5675 processors, 3.07 Ghz, and a total of 12 cores. We ran different instances using different cores separately.

6.5.1 Instances

We generated 1000 instances with different parameter settings. The parameters considered in the generation of the instances include the length of the planning horizon (L), the incentive factor (IF) and the number of operating rooms over the planning horizon (OR). We set the number of weeks in the planning horizon to {2, 3, 4, 5}. We also assume that surgeries are scheduled only on workdays. We define the incentive factor (IF) as the ratio of c_2/c_1 where c_1 and c_2 are the daily cost of first-stage and second-stage nurses in Objective function (6.7). A higher value of the incentive factor shows that the hospital pays more to second-stage nurses than first-stage ones. We set the incentive factor to {1.1, 1.3, 1.5, 1.7, 1.9}. We suppose that first-stage nurses are paid 1 unit cost per hour which for 8 work hours results in $c_1 = 8$. Furthermore, we also fix the number of operating rooms over the planning horizon at {1, 2, 3, 4, 5}. For each operating room we generate 3, 4, or 5 surgeries randomly with a uniform distribution. Considering a full factorial experiment, 100 combinations of L, IF and OR are possible and we generate 10 instances for each problem setting for a total of 1000 instances. For each patient, we generate two scenarios for the length of stays in ICU and wards. In each scenario, both lengths of stays are uniformly generated from interval [1 day, 10 days]. The total number of global scenarios which include information for all patients can be computed by $2^{|T|}$ where $|T|$ is the number of patients in the planning horizon. It is worth noting that even in our small-sized instances there is a large number of global scenarios.

We also assume that each nurse works for 8 hours a day ($\delta = 8$) and the average daily service time for each patient is 2 hours ($\rho = 2$).

6.5.2 Parameters

In Algorithms 6.1 to 6.3, we set $AlgTimeLimit$ to 2 hours. In the Benders-Dual algorithm, we fix the convergence limits $\delta_{acc}^{Benders}$ and δ_{acc}^{Dual} at 0.1%. We use the same values $\delta_{acc}^{Benders}$ and $delta_{acc}^{Dual}$ for the Bender and dual algorithms respectively. We also consider 5 seconds for $Time_{LB}$ and $Time_{UB}$ in the stopping conditions of the master problem and the subproblem in Algorithms 6.1 and 6.3. Furthermore, in Procedure 6.1 of Algorithms 6.1 and 6.3 we set $EvaTimeLimit$ to 2 hours. Therefore, considering parameters $AlgTimeLimit$ and $EvaTimeLimit$, we run a problem instance for at most 4 hours by Algorithms 6.1 and 6.3 and 2 hours by Algorithm 6.2. In Algorithms 6.2 and 6.3, we fix $LocalDualTimeLimit$ at 30 seconds. In the Benders-dual algorithm, we consider 20 minutes for $WarmupTimeLimit$.

6.5.3 Results

A short summary of the results are as follows : 1) The dual algorithm converges in a few number of iterations and therefore results in larger optimality gaps compared to the other algorithms. 2) In the Benders algorithm the optimality gap decreases considerably during the algorithm (from around 53% to less than 4%). 3) The Benders-dual algorithm relatively outperforms the Benders algorithm in terms of the objective value. In both algorithms the lower bound improve quickly in initial iterations. 4) The value of adjustability, a criterion defined to compare the quality of the objective value for the two-stage robust model with that of the non-adaptive robust problem, shows that by solving the adaptive problem we find solutions which are around 5% less costly. 5) We have also observed that in instances where second-stage nurses are paid with a lower rate, the number of nurses hired in the first-stage is fewer and the value of adjustability is higher.

We report the results of the Benders and dual algorithms for instances with the planning horizon of 2, 3, 4 and 5 weeks in Tables 6.2 to 6.5 respectively. Tables 6.6 to 6.9 also present the results of the Benders-dual algorithm for different planning horizons. In all of these tables, each row represents the average over 10 instances. Under "*Data Info.*", "*IF*", "*OR*", and "*Sur.*"respectively give the incentive factor, the number of operating room, and the number of surgeries over the planning horizon. "*Time (sec)*"gives the total computational time of algorithms in seconds. Also "*Ite.*"gives the number of iterations that algorithms repeat their main loops. Furthermore, "$LB^*_{\cdot}$"and "$UB^*_{\cdot}$"indicate the best lower and upper bounds in the last iteration of algorithms and "$Gap^*_{\cdot}$"computes the gap between these bounds. The subscripts of "$LB^*_{\cdot}$", "$UB^*_{\cdot}$"and "$Gap^*_{\cdot}$"in Tables 6.2 to 6.9 are either "B", "D"or "BD"to represent the Benders, dual and Benders-dual algorithms respectively. Since the dual algorithm does not provide any lower bound we do not report LB^*_D in Tables 6.2 to 6.5. We use the best lower bound of the Benders algorithm (LB^*_B) to compute Gap^*_D (i.e., $Gap^*_D = 100((UB^*_D - LB^*_B)/LB^*_B)$.

In Tables 6.2 to 6.5, under "Benders algorithm", "LB_1"and "UB_1"give the lower and upper bounds in the first iteration of the Benders algorithm and Gap_1 computes the gap between these bounds. "N^*"gives the average number of first-stage nurses in the planning horizon for the best obtained solution. In the nurse planning problem if we forbid hiring the second-stage nurses, the problem will be a non-adjustable robust problem. In Appendix C.11 we explain how to solve the non-adjustable version of the robust nurse planning problem presented in Section 6.2. In Tables 6.2 to 6.5, under "*Benders algorithm*", we report some computational results of the non-adjustable problem and compare them with the results of the Benders algorithm. "N_{nonadj}"gives the average number of first-stage nurses in the optimal solution of the non-adjustable problem. For the best and non-adjustable solutions we present more details including the ranges and the standard deviations in Appendix EC.12. The results in this appendix show that the standard deviations are insignificant compared to averages presented in Tables 6.2 to 6.5. "V_{adj}"represents the value of adjustability. We compute it by $100(Opt_{nonadj} - UB^*_B)/(UB^*_B)$ where Opt_{nonadj} is the optimal objective value of the non-adjustable problem. The value of adjustability shows how much the objective value can be worse if we do not have the flexibility of hiring second-stage nurses. In Tables 6.2 to 6.5, "Imp", gives the percentage of the upper bound improvement obtained during the Benders algorithm. We compute it by $100(UB_1 - UB^*_B)/UB_1$.

In Tables 6.2 to 6.9, under "*Dual algorithm*"and "*Benders-dual algorithm*", "*TC*"gives the total number of cuts that Procedure 6.2 generates in the dual and Benders-dual algorithms. Some cuts generated in initial iterations of Algorithms 6.2 and 6.3 are dominated by other cuts generated in next iterations. In this case, CPLEX removes the dominated cuts from the inner max problem of Model (P6). "*TCLI*"gives the total number of non-dominated cuts in the last iteration of the algorithms.

In Tables 6.6 to 6.9, "*C1*"to "*C8*"give the number of different cuts generated by Procedure 6.2 in the Benders-dual algorithm. Cuts referred by these columns are respectively cliques cuts, covers cuts, flow covers cuts, fractional cuts, generalized upper bound covers cuts, implied bound cuts, mixed rounding cuts and zero-half cuts. More explanations about these cuts are available in CPLEX User's Manual 12.6 (IBM 2015). "$\Delta(UB^*)$"also gives the percentage by which the best upper bound of the Benders-dual algorithm is superior to the best upper bound of the Benders algorithm (i.e., $\Delta(UB^*) = 100(UB^*_B - UB^*_{BD})/UB^*_B$).

In Tables 6.2 to 6.5, the averages of initial gaps in the first iteration of the Benders algorithm (Gap_1) are 66.29%, 55.10%, 47.56%, and 42.80%. At the end of the Benders algorithm the optimality gaps decrease to 0.05%, 2.61%., 5.13%, and 6.98% respectively. Comparison of "Gap_1"and "Gap^*_B"demonstrates that the Benders algorithm significantly improves the optimality gap. Moreover, in these tables, the averages of "Imp"are 5.14%, 4.80%, 3.31%, and 2.34%. These averages show that the Benders algorithm improves the upper bound during the algorithm and the improvement of optimality gap is not only because of improving the lower bound. We also observe that the upper bound improvement decreases as the length of the planning horizon increases. This observation confirms that instances with longer planning horizons are more difficult and the Benders algorithm becomes less effective in solving them. Similarly instances with more operating rooms are more difficult and the Benders algorithm performs more iterations before stopping for such instances.

In Tables 6.2 to 6.5, the averages of adjustability values (V_{adj}) are 6.76%, 7.46%, 6.30%, and 5.33%. These values demonstrate that hiring nurses in the second stage improves the objective value of the non-adjustable problem around 6.46% on average. We can see that there is no significant relation between the length of the planning horizon and the value of adjustability. However, as depicted in Figure 6.1 there is a strong relation between the adjustability value and incentive factor. As shown in this figure as the incentive factor increases the adjustability value decreases. We expected this observation because a higher incentive factor means that second-stage nurses are proportionally paid more than first-stage nurses. As a result the decision maker is more inclined in hiring first-stage nurses rather than second-stage nurses, and the adjustability value decreases. In Tables 6.2 to 6.5, we observe that for all instances the values of N_{nonadj} are higher than the values of N^*. This is because in the case of non-adjustability, we are not allowed to hire any second-stage nurse, and therefore we hire more first-stage nurses. Figure 6.2 also depicts the relation between the incentive factor and the average number of first-stage nurses in the best solution (N^*). As expected we can see that N^* is increasing in the incentive factor. It is also interesting to see that for instances with the planning horizon of two weeks, N^* does not change as the incentive factor increases. This behavior is not due to any poor performance of the Benders algorithm because as we can see in Table 6.2 the Benders algorithm solves most instances optimally. For instances in Table 6.2 we increased the value of IF gradually and observed that values of N^* increase for $IF \geq 2$ and reach the values of N_{nonadj} at $IF = 11$.

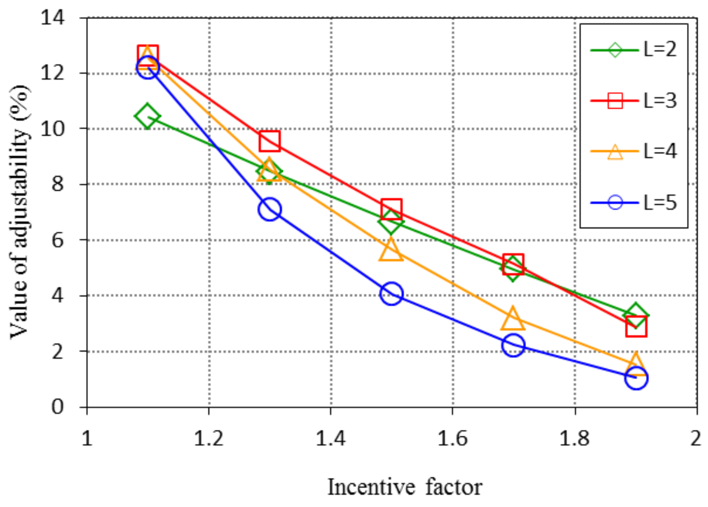

Figure 6.1 Changes in the value of adjustability in terms of incentive factor for different planning horizons.

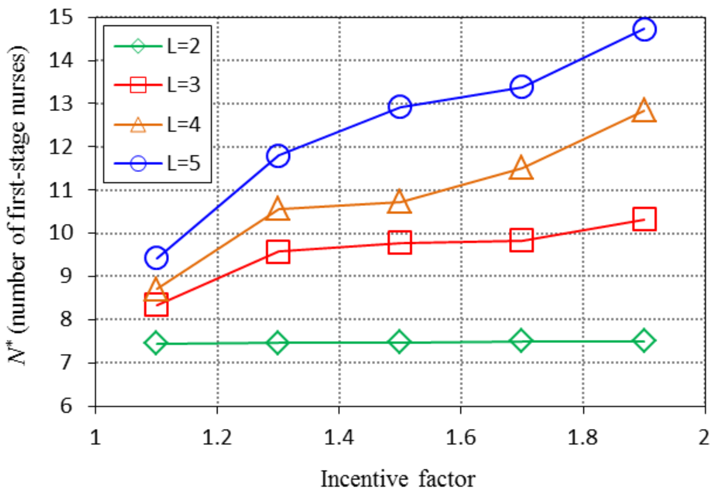

Figure 6.2 Changes in the number of nurses in terms of incentive factor for different planning horizons.

In Tables 6.2 to 6.5, we can see that for most instances the dual algorithm converges quickly after only a few iterations and the average optimality gaps are worse than those of the Benders algorithm. This is because the dual algorithm is a heuristic, while the Benders algorithm is an exact algorithm and converges to the optimal solution. We can also see that the values of "*TC*"and "*TCLI*" are very close even for instances where the dual algorithm does not converge very fast.

In Tables 6.6 to 6.9, the averages of best gaps for the Benders-dual algorithm (Gap^*_{BD}) are 0.61%, 3.30%, 5.46%, and 7.58%. These averages are higher than the averages of optimality gaps for the Benders algorithm. However, in these tables, the averages of "$\Delta(UB^*)$"are -0.46%, 0.09%, 0.94%, and 1.19% respectively. These values show that the Benders-dual algorithm finds better upper bounds than the Benders algorithm in instances with planning horizons of 3, 4, and 5 weeks. The averages of "$\Delta(UB^*)$"are especially more noticeable in the last two tables considering the averages of "Gap^*_B"and "$Imp(\%)$"for the Benders algorithm in Tables 6.4 and 6.5. In Tables 6.6 to 6.9 we observe that for instances with longer planning horizons the averages of computational time (*Time (sec)*) and also the total number of generated cuts (*TC*) are higher. Furthermore, a comparison of "*TC*"and "*TCLI*" shows that on average between 28% and 31% of cuts generated by Procedure 6.2 in the Benders-dual algorithm are dominated by other cuts generated in next iterations of the algorithm. Figure 6.3 also depicts the number of generated cuts during the Benders-dual algorithm for instances with $L = 5$ and $OR = 5$ in Table 6.9. To draw this figure, iterations after time limit *WarmupTimeLimit* are taken into account because before this time limit Procedure 6.2 does not generate any cut in the Benders-dual algorithm. Since the number of iterations in the Benders-dual algorithm is different for each of the considered instances, we present the iteration progress in the horizontal axis of Figure 6.3 in terms of percentage. In this figure, the total number of generated cuts increases linearly in terms of the number of iterations. We can observe that there is a linear trend for each single cut except for Type 1 cuts (C1) that are generated more in the beginning. This is because Type 1 cuts are clique cuts and most of them are detected and added to the model in the presolve procedure of CPLEX.

Figure 6.4 provides insights into lower and upper bound improvements during the Benders-dual algorithm. In this figure, it is noticeable that in early iterations of the algorithm, the lower bound increases very rapidly from 1889 to 2278 and then within the initial $WarmupTimeLimit$ it improves gradually. However, after *WarmupTimeLimit* the algorithm does not improve the lower bound significantly. The reason is that after *WarmupTimeLimit* the Benders-dual algorithm starts to call Procedure 6.2 to generate cuts. As a result the main loop of Algorithm 6.3 becomes more time consuming and the algorithm calls the master problem less often than before. Therefore the lower bound does not improve with the same rate. Moreover, in this figure we can see that before *WarmupTimeLimit* the upper bound decreases slowly at a linear rate, but after this time limit it improves very quickly because of calls to Procedure 6.2.

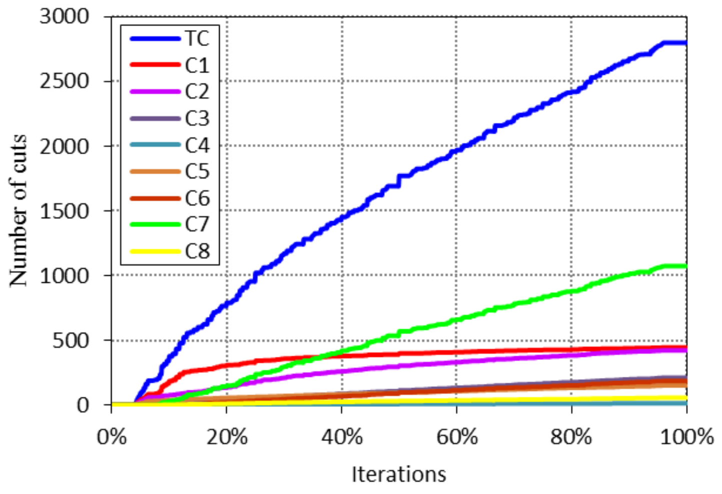

Figure 6.3 Number of cuts generated during the Benders-dual algorithm for instances with $L = 5$ and $OR = 5$.

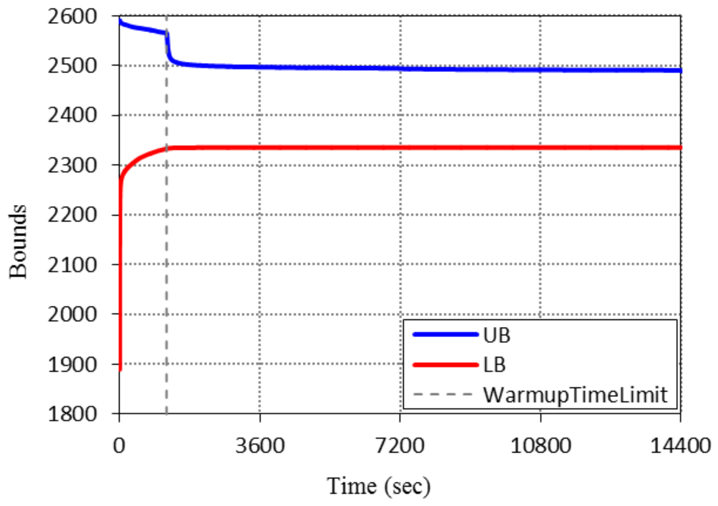

Figure 6.4 Best lower and upper bounds during the Benders-dual algorithms for all instances in Tables 6.6 to 6.9.

6.6 Conclusion

In this paper, we have considered a class of two-stage robust optimization models with integer adversarial variables. We discussed that this class of robust problems is useful in modeling two-stage robust resource planning problems where some tasks with uncertain arrivals and durations are expected. We exploited the structure of the problem using Dantzig-Wolfe decomposition and reduced the original two-stage robust problem to a single-stage robust problem. We then proposed a Benders algorithm with a master problem and a subproblem which are mixed-integer programs. As it is unwieldy to optimally solve these mixed integer programs in each iteration of the Benders algorithm we presented novel stopping conditions for them and provided the relevant convergence proofs. We also developed a heuristic algorithm, namely dual algorithm, and combined it with the Benders algorithm to create a more effective algorithm capable of finding solutions with tighter optimality gaps. We performed extensive computational experiments to compare the proposed algorithms in a nurse planning problem. The computational results demonstrated that the Benders and hybrid algorithms find solutions with an average optimality gap of less than 5% in instances with planning horizons up to five weeks. A possible future research direction would be to explore the extension of the proposed algorithms to multi-stage robust problems with integer adversarial variables. Another extension would be to develop a column-and-constraint generation to solve the reformulated problem in the case that it has a large number of variables and constraints. Finally, applying the proposed algorithms to other applications should definitely be of interest.

Table 6.2 – Computational results of the Benders and dual algorithms for instances with a planning horizon of two weeks ($L = 2$).

Data Info.			*Benders algorithm*												*Dual algorithm*					
IF	*OR*	*Sur.*	LB_1	UB_1	LB_B^*	UB_B^*	*Time (sec)*	*Ite.*	N^*	N_{nonadj}	V_{adj} (%)	Gap_1 (%)	Gap_B^* (%)	*Imp* (%)	*TC*	*TCLI*	*Time (sec)*	*Ite.*	UB_D^*	Gap_D^* (%)
1.1	1	39	294	339	325	325	1	27	2.63	3.15	8.48	15.98	0	4.18	59	57	4	4	337	3.7
	2	79	600	655	626	626	12	47	5	6.2	10.89	9.31	0	4.36	476	458	187	24	642	2.55
	3	119	902	966	926	926	63	64	7.45	9.1	10.07	7	0	4.09	621	600	500	28	943	1.89
	4	157	1182	1260	1211	1211	1258	90	9.73	12.07	11.65	6.59	0	3.92	790	762	495	30	1230	1.58
	5	202	1501	1583	1530	1531	5888	110	12.45	15.19	11.12	5.47	0.08	3.27	618	594	168	21	1551	1.41
1.3	1	39	241	353	331	331	2	30	2.63	3.15	6.63	47.53	0	6.12	12	11	0	2	353	6.62
	2	79	396	691	638	638	13	48	5	6.2	8.78	81.42	0	7.62	137	136	87	7	686	7.48
	3	119	506	1019	942	942	61	66	7.46	9.1	8.16	104.95	0	7.53	28	28	0	2	1015	7.65
	4	157	699	1352	1233	1233	1094	87	9.74	12.07	9.66	94.94	0	8.8	38	37	0	2	1346	9.18
	5	202	1014	1701	1553	1558	8575	119	12.44	15.19	9.19	69.84	0.32	8.41	44	38	0	2	1694	9.07
1.5	1	39	241	353	336	336	2	31	2.63	3.15	4.85	47.53	0	4.54	18	17	0	2	352	4.73
	2	79	378	694	650	650	12	50	5	6.2	6.75	87.44	0	6.27	34	33	0	2	693	6.49
	3	119	506	1019	959	959	44	65	7.52	9.1	6.3	104.95	0	5.91	55	53	0	3	1012	5.57
	4	157	699	1352	1255	1255	1167	93	9.76	12.07	7.74	94.94	0	7.17	41	39	0	2	1345	7.21
	5	202	1014	1701	1578	1581	7906	113	12.44	15.19	7.62	69.84	0.19	7.07	47	43	0	2	1695	7.45
1.7	1	39	241	353	341	341	1	28	2.67	3.15	3.32	47.53	0	3.16	18	17	0	2	353	3.24
	2	79	378	694	662	662	13	50	5.06	6.2	4.86	87.44	0	4.59	38	37	0	2	694	4.75
	3	119	506	1019	974	974	44	65	7.54	9.1	4.64	104.95	0	4.42	54	52	0	3	1013	3.96
	4	157	699	1352	1276	1276	1097	87	9.8	12.07	5.97	94.94	0	5.62	48	46	1	2	1346	5.5
	5	202	1014	1701	1602	1605	7529	110	12.45	15.19	5.98	69.84	0.2	5.63	49	46	1	2	1696	5.88
1.9	1	39	241	353	346	346	2	29	2.67	3.15	1.84	47.53	0	1.77	12	11	0	2	353	1.83
	2	79	378	694	673	673	11	49	5.06	6.2	3.11	87.44	0	2.98	39	38	0	2	694	3.07
	3	119	506	1019	989	989	51	66	7.54	9.1	3.03	104.95	0	2.93	44	43	0	2	1016	2.73
	4	157	699	1352	1297	1297	2058	96	9.8	12.07	4.25	94.94	0	4.07	60	57	0	2	1347	3.85
	5	202	1014	1701	1625	1632	8667	115	12.44	15.19	4.22	69.84	0.41	4.04	39	35	0	2	1694	4.23
Average		119	634	1011	955	956	1823	69	7.48	9.14	6.76	66.29	0.05	5.14	137	132	58	6	1004	4.86

Table 6.3 – Computational results of the Benders and dual algorithms for instances with a planning horizon of three weeks ($L = 3$).

Data Info.			*Benders algorithm*													*Dual algorithm*					
IF	*OR*	*Sur.*	LB_1	UB_1	LB_B^*	UB_B^*	*Time (sec)*	*Ite.*	N^*	N_{nonadj}	V_{adj} (%)	Gap_1 (%)	Gap_B^* (%)	Imp (%)	*TC*	*TCLI*	*Time (sec)*	*Ite.*	UB_D^*	Gap_D^* (%)
1.1	1	59	615	677	646	646	34	42	3.36	4.31	12.13	10.1	0	4.6	649	626	447	32	662	2.56
	2	121	1247	1329	1268	1274	9253	105	6.59	8.64	13.97	6.6	0.46	4.12	1016	979	1381	39	1299	2.44
	3	182	1819	1913	1839	1880	14400	132	9.66	12.61	12.75	5.16	2.23	1.73	1012	977	994	31	1875	1.94
	4	240	2384	2479	2409	2478	13665	209	9.38	16.57	12.33	3.98	2.87	0.03	1125	1086	1376	31	2450	1.72
	5	300	2974	3108	3006	3108	14400	293	12.73	20.72	11.96	4.52	3.38	0	1466	1419	1953	35	3055	1.62
1.3	1	59	450	716	660	660	29	39	3.36	4.31	9.63	66.56	0	7.72	388	378	464	21	699	5.87
	2	121	918	1434	1298	1304	10628	111	6.55	8.64	11.33	66.9	0.52	9.05	480	458	290	18	1392	7.28
	3	182	1414	2110	1875	1924	14400	152	9.6	12.61	10.19	52.07	2.57	8.86	598	572	287	15	2027	8.07
	4	240	1906	2759	2452	2560	14400	243	12.6	16.57	8.74	46.73	4.41	7.23	551	527	279	12	2672	9.05
	5	300	2584	3423	3065	3227	14400	471	15.74	20.72	7.85	33.91	5.27	5.73	970	931	941	20	3273	6.73
1.5	1	59	364	724	674	674	28	44	3.43	4.31	7.42	99.58	0	6.86	44	43	0	2	720	6.88
	2	121	712	1452	1326	1328	5832	102	6.86	8.64	9.3	105.37	0.15	8.49	83	79	1	3	1437	8.34
	3	182	1378	2119	1916	1970	13680	205	9.79	12.61	7.6	55.61	2.78	7.05	144	137	1	4	2100	9.59
	4	240	1812	2783	2505	2626	14400	304	12.7	16.57	5.98	54.26	4.84	5.64	89	85	1	3	2778	10.93
	5	300	2404	3480	3130	3310	14400	448	16.08	20.72	5.13	45.17	5.74	4.87	96	92	1	3	3467	10.76
1.7	1	59	364	724	685	685	26	49	3.51	4.31	5.63	99.58	0	5.29	39	38	0	2	719	4.95
	2	121	712	1452	1349	1351	5508	95	6.91	8.64	7.46	105.37	0.13	6.92	92	90	1	3	1441	6.81
	3	182	1378	2119	1953	2002	14400	164	9.83	12.61	5.87	55.61	2.51	5.53	127	123	1	3	2106	7.85
	4	240	1812	2783	2552	2685	14400	241	12.81	16.57	3.67	54.26	5.18	3.53	87	83	1	3	2775	8.73
	5	300	2404	3480	3183	3377	14400	345	16.08	20.72	3.04	45.17	6.09	2.94	120	114	2	4	3463	8.78
1.9	1	59	364	724	697	697	25	48	3.51	4.31	3.9	99.58	0	3.72	43	42	0	2	722	3.58
	2	121	712	1452	1372	1372	3894	95	6.93	8.64	5.86	105.37	0	5.52	78	75	0	3	1442	5.16
	3	182	1378	2119	1985	2053	13680	182	9.91	12.61	3.26	55.61	3.38	3.15	112	106	1	3	2106	6.05
	4	240	1812	2783	2600	2754	14400	376	13.24	16.57	1.08	54.26	5.89	1.05	85	79	1	3	2780	6.93
	5	300	2386	3480	3242	3468	14400	523	18.01	20.72	0.33	46.08	6.97	0.33	127	120	2	4	3462	6.76
Average		180	1452	2065	1907	1977	9964	201	9.57	12.57	7.46	55.1	2.61	4.8	385	370	337	12	2037	6.38

Table 6.4 – Computational results of the Benders and dual algorithms for instances with a planning horizon of four weeks ($L = 4$).

Data Info.			*Benders algorithm*												*Dual algorithm*					
IF	*OR*	*Sur.*	LB_1	UB_1	LB_B^*	UB_B^*	*Time* (*sec*)	*Ite.*	N^*	N_{nonadj}	V_{adj} (%)	Gap_1 (%)	Gap_B^* (%)	*Imp* (%)	*TC*	*TCLI*	*Time* (*sec*)	*Ite.*	UB_D^*	Gap_D^* (%)
1.1	1	80	951	1033	987	987	1352	79	3.82	5.01	13.61	8.6	0	4.45	1044	1011	2372	42	1014	2.69
	2	163	1897	2018	1923	1975	14400	128	7.59	9.88	12.06	6.38	2.7	2.13	1470	1422	3099	41	1969	2.39
	3	241	2706	2846	2738	2846	14400	299	6.69	14.34	12.91	5.14	3.92	0	1319	1269	1901	33	2796	2.12
	4	318	3527	3710	3567	3710	14400	334	10.52	18.73	13.15	5.17	3.99	0	1576	1520	1805	32	3641	2.06
	5	397	4322	4662	4382	4661	14400	493	14.92	23.09	10.97	7.88	6.36	0.01	2931	2861	5535	42	4472	2.05
1.3	1	80	757	1103	1011	1011	1079	78	3.82	5.01	10.94	51.11	0	8.35	669	641	270	24	1064	5.26
	2	163	1570	2179	1962	2021	13680	204	7.48	9.88	9.48	47.57	3.02	7.22	1018	972	1251	27	2081	6.11
	3	241	2510	3112	2789	2946	14400	510	10.61	14.34	9.04	25.65	5.65	5.3	1530	1445	1925	30	2933	5.16
	4	318	3401	4039	3639	3905	14400	720	13.9	18.73	7.49	18.77	7.31	3.3	1969	1874	2708	33	3779	3.86
	5	397	4054	5039	4476	4892	14400	766	17.02	23.09	5.74	24.89	9.28	2.9	2645	2554	6300	40	4698	4.91
1.5	1	80	576	1122	1034	1034	1027	80	3.92	5.01	8.49	95.76	0	7.8	132	130	0	3	1107	7.08
	2	163	1214	2213	2004	2074	14400	212	7.57	9.88	6.74	85.05	3.46	6.3	146	142	1	3	2197	9.64
	3	241	2066	3212	2855	3029	14400	485	10.79	14.34	6.07	57.99	6.1	5.71	175	167	2	3	3192	11.82
	4	318	2878	4197	3722	4019	14400	736	13.99	18.73	4.43	46.23	7.99	4.23	165	152	3	4	4175	12.18
	5	397	3606	5172	4577	5043	14400	942	17.36	23.09	2.57	43.82	10.17	2.49	198	185	7	4	5151	12.54
1.7	1	80	576	1122	1052	1052	694	88	4.02	5.01	6.56	95.76	0	6.13	98	98	0	2	1109	5.39
	2	163	1214	2213	2042	2120	13679	188	7.63	9.88	4.41	85.05	3.79	4.21	130	123	1	3	2199	7.7
	3	241	2066	3212	2917	3113	14400	543	10.82	14.34	3.2	57.99	6.72	3.09	143	134	2	4	3192	9.43
	4	318	2878	4197	3796	4134	14400	714	14.26	18.73	1.53	46.23	8.9	1.5	145	133	3	4	4183	10.2
	5	397	3606	5172	4670	5155	14400	732	20.85	23.09	0.33	43.82	10.39	0.32	170	161	4	4	5154	10.37
1.9	1	80	576	1122	1070	1070	764	102	4.02	5.01	4.77	95.76	0	4.53	90	88	0	2	1117	4.34
	2	163	1214	2213	2079	2170	14400	247	7.67	9.88	2.01	85.05	4.37	1.95	121	118	1	3	2201	5.87
	3	241	2066	3212	2970	3185	14400	589	11.14	14.34	0.87	57.99	7.23	0.85	118	112	2	3	3199	7.73
	4	318	2878	4197	3873	4195	14400	680	18.28	18.73	0.04	46.23	8.32	0.03	136	127	3	3	4184	8.05
	5	397	3577	5172	4758	5172	14400	784	23.09	23.09	0	45.02	8.7	0	206	188	5	4	5156	8.35
Average		240	2267	3100	2836	3021	11659	429	10.87	14.21	6.3	47.56	5.13	3.31	734	705	1088	16	3039	6.69

Table 6.5 – Computational results of the Benders and dual algorithms for instances with a planning horizon of five weeks ($L = 5$).

Data Info.			*Benders algorithm*												*Dual algorithm*					
IF	*OR*	*Sur.*	LB_1	UB_1	LB_B^*	UB_B^*	*Time (sec)*	*Ite.*	N^*	N_{nonadj}	V_{adj} (%)	Gap_1 (%)	Gap_B^* (%)	Imp (%)	TC	$TCLI$	*Time (sec)*	*Ite.*	UB_D^*	Gap_D^* (%)
1.1	1	101	1267	1366	1303	1309	10929	122	4.04	5.29	13.08	7.8	0.49	4.17	1454	1408	4939	52	1341	2.98
	2	202	2437	2607	2467	2587	14400	182	7.31	10.33	11.79	6.97	4.85	0.74	1739	1682	3945	41	2537	2.83
	3	302	3621	3807	3657	3807	14400	311	7.28	15.31	12.64	5.13	4.09	0	1923	1854	3522	39	3734	2.11
	4	401	4712	5010	4766	5010	12281	423	11.74	20.19	12.83	6.31	5.1	0	2505	2445	4937	37	4882	2.43
	5	503	5876	6402	5951	6402	10507	307	16.74	25.3	10.64	8.95	7.57	0	3811	3693	7200	34	6128	2.97
1.3	1	101	951	1462	1333	1341	11157	163	4.06	5.29	10.38	60.96	0.57	8.26	751	714	404	22	1418	6.32
	2	202	2216	2822	2513	2646	14400	482	7.55	10.33	9.31	27.89	5.3	6.2	1819	1741	2776	38	2641	5.11
	3	302	3394	4122	3720	3972	14400	591	11.33	15.31	7.96	21.45	6.76	3.6	2016	1931	3609	34	3917	5.24
	4	401	4522	5456	4848	5412	13697	476	15.47	20.19	4.47	20.68	11.62	0.81	2706	2604	5040	36	5078	4.73
	5	503	5636	6848	6059	6848	13398	408	20.57	25.3	3.44	21.79	13.02	0	3481	3367	6480	31	6434	6.2
1.5	1	101	786	1480	1363	1370	10342	171	4.15	5.29	8.03	92.55	0.53	7.42	153	148	1	3	1466	7.57
	2	202	1738	2892	2578	2739	14400	476	7.72	10.33	5.59	70.92	6.24	5.28	109	107	1	2	2886	11.92
	3	302	3031	4287	3808	4103	14400	640	11.47	15.31	4.51	43.01	7.74	4.31	162	153	3	3	4268	12.11
	4	401	3866	5652	4966	5533	14400	793	16	20.19	2.15	47.23	11.44	2.09	203	194	3	3	5643	13.64
	5	503	5048	7083	6202	7083	14400	522	25.3	25.3	0	40.65	14.21	0	260	247	10	4	7064	13.89
1.7	1	101	786	1480	1387	1393	7985	172	4.23	5.29	6.26	92.55	0.44	5.88	166	161	1	3	1469	5.92
	2	202	1738	2892	2627	2806	14400	443	7.77	10.33	3.09	70.92	6.79	2.99	124	121	1	2	2884	9.8
	3	302	3031	4287	3879	4219	14400	672	11.6	15.31	1.63	43.01	8.75	1.6	165	157	3	3	4273	10.16
	4	401	3866	5652	5074	5644	14400	733	18.05	20.19	0.14	47.23	11.23	0.13	194	188	4	3	5644	11.23
	5	503	5048	7083	6312	7083	12289	473	25.3	25.3	0	40.65	12.21	0	264	251	8	4	7069	11.98
1.9	1	101	786	1480	1410	1416	7644	244	4.29	5.29	4.54	92.55	0.4	4.33	208	202	1	4	1465	3.86
	2	202	1738	2892	2675	2874	14400	580	8.91	10.33	0.66	70.92	7.41	0.64	161	155	2	3	2881	7.7
	3	302	3031	4287	3951	4287	14400	634	14.94	15.31	0.01	43.01	8.52	0	155	150	3	3	4271	8.13
	4	401	3866	5652	5176	5652	13680	650	20.19	20.19	0	47.23	9.21	0	210	201	4	3	5644	9.06
	5	503	5085	7083	6439	7083	12686	455	25.3	25.3	0	39.57	10.01	0	250	237	7	4	7066	9.75
Average		302	3123	4163	3779	4105	12952	445	12.45	15.28	5.33	42.8	6.98	2.34	1000	964	1716	16	4084	7.51

Table 6.6 – Computational results of the Benders-dual algorithm for instances with a planning horizon of two weeks ($L = 2$).

Data Info.			*Benders-dual algorithm*															
IF	*OR*	*Sur.*	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	TC	$TCLI$	*Time (sec)*	*Ite.*	LB^*_{BD}	UB^*_{BD}	Gap^*_{BD} (%)	$\Delta(UB^*)$ (%)
1.1	1	39	0	0	0	0	0	0	0	0	0	0	2	35	325	325	0	0
	2	79	0	0	0	0	0	0	0	0	0	0	43	97	626	626	0	0
	3	119	0	0	0	0	0	0	0	0	0	0	326	172	926	926	0	0
	4	157	60	258	258	153	54	118	770	257	1928	1380	6759	322	1209	1216	0.59	-0.44
	5	202	79	343	355	132	90	123	1100	169	2390	1669	7494	313	1524	1539	1.04	-0.54
1.3	1	39	0	0	0	0	0	0	0	0	0	0	2	35	331	331	0	0
	2	79	0	0	0	0	0	0	0	0	0	0	57	102	638	638	0	0
	3	119	5	37	32	24	7	8	75	51	240	181	1083	212	942	943	0.1	-0.11
	4	157	87	372	237	190	73	120	869	347	2295	1661	7637	334	1229	1246	1.34	-1.05
	5	202	69	425	271	130	87	118	1016	197	2312	1708	8074	322	1544	1576	2.1	-1.19
1.5	1	39	0	0	0	0	0	0	0	0	0	0	2	34	336	336	0	0
	2	79	0	0	0	0	0	0	0	0	0	0	53	100	650	650	0	0
	3	119	13	54	19	46	11	16	143	112	413	299	1646	224	959	960	0.16	-0.17
	4	157	88	392	218	202	60	112	904	361	2336	1692	7610	339	1252	1272	1.6	-1.38
	5	202	110	441	232	165	76	149	1113	177	2463	1777	8118	324	1570	1608	2.43	-1.75
1.7	1	39	0	0	0	0	0	0	0	0	0	0	2	33	341	341	0	0
	2	79	0	0	0	0	0	0	0	0	0	0	22	69	662	662	0	0
	3	119	0	0	0	0	0	0	0	0	0	0	163	112	974	974	0	0
	4	157	110	220	148	99	88	97	664	152	1577	1109	5379	237	1274	1285	0.78	-0.69
	5	202	122	367	206	155	157	144	1132	137	2420	1612	9016	300	1599	1626	1.69	-1.3
1.9	1	39	0	0	0	0	0	0	0	0	0	0	2	33	346	346	0	0
	2	79	0	0	0	0	0	0	0	0	0	0	17	65	673	673	0	0
	3	119	0	0	0	0	0	0	0	0	0	0	110	103	989	989	0	0
	4	157	125	168	156	72	119	66	481	124	1311	946	5063	259	1296	1311	1.1	-1.04
	5	202	195	281	278	110	210	98	885	119	2176	1534	7983	297	1623	1661	2.34	-1.75
Average		119	43	134	96	59	41	47	366	88	874	623	3067	179	954	962	0.61	-0.46

Table 6.7 – Computational results of the Benders-dual algorithm for instances with a planning horizon of three weeks ($L = 3$).

Data Info.			*Benders-dual algorithm*															
IF	*OR*	*Sur.*	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	TC	$TCLI$	*Time (sec)*	*Ite.*	LB^*_{BD}	UB^*_{BD}	Gap^*_{BD} (%)	$\Delta(UB^*)$ (%)
1.1	1	59	0	0	0	0	0	0	0	0	0	0	101	83	646	646	0	0
	2	121	58	254	293	163	48	132	1184	287	2418	1807	8442	280	1264	1279	1.2	-0.41
	3	182	90	303	338	98	77	126	1279	131	2442	1828	10806	274	1828	1858	1.63	1.17
	4	240	96	290	323	34	100	132	954	61	1990	1408	11583	235	2396	2439	1.79	1.58
	5	300	105	291	301	12	135	143	965	39	1989	1341	12938	251	2988	3050	2.05	1.88
1.3	1	59	0	0	0	0	0	0	0	0	0	0	139	102	660	660	0	0
	2	121	73	336	236	155	61	140	1210	379	2590	1974	8969	280	1285	1321	2.8	-1.32
	3	182	104	451	272	99	86	159	1175	141	2487	1874	12881	274	1851	1922	3.84	0.07
	4	240	77	520	306	53	117	244	1148	98	2563	1850	13749	272	2423	2525	4.24	1.33
	5	300	116	540	192	22	137	226	922	41	2194	1546	14185	269	3026	3154	4.22	2.23
1.5	1	59	0	0	0	0	0	0	0	0	0	0	58	81	674	674	0	0
	2	121	108	347	199	161	57	147	1415	358	2793	2154	9413	287	1310	1356	3.46	-2.09
	3	182	87	463	209	118	63	169	1250	125	2485	1862	12539	275	1892	1976	4.41	-0.3
	4	240	86	484	221	68	97	254	1166	74	2451	1796	12577	273	2475	2605	5.25	0.8
	5	300	156	473	164	42	101	250	1106	47	2338	1690	14400	269	3089	3262	5.6	1.44
1.7	1	59	0	0	0	0	0	0	0	0	0	0	37	53	685	685	0	0
	2	121	118	207	113	110	101	79	651	180	1557	1038	8637	228	1344	1369	1.83	-1.32
	3	182	116	404	206	117	103	158	1337	119	2559	1829	13336	279	1934	2018	4.35	-0.8
	4	240	115	369	177	63	128	235	1094	64	2246	1590	13805	272	2525	2676	5.94	0.34
	5	300	168	451	200	54	143	215	1209	53	2493	1771	14400	272	3150	3375	7.13	0.08
1.9	1	59	0	0	0	0	0	0	0	0	0	0	41	59	697	697	0	0
	2	121	195	193	162	78	212	80	630	155	1705	1154	7963	229	1368	1390	1.57	-1.33
	3	182	248	339	278	62	268	166	1071	120	2552	1768	11888	279	1967	2078	5.64	-1.24
	4	240	218	352	277	46	260	188	1027	85	2452	1660	12786	278	2568	2762	7.54	-0.31
	5	300	248	274	212	25	239	181	857	64	2100	1330	12542	275	3201	3455	7.92	0.39
Average		180	103	294	187	63	101	137	866	105	1856	1331	9529	229	1890	1969	3.3	0.09

Table 6.8 – Computational results of the Benders-dual algorithm for instances with a planning horizon of four weeks ($L = 4$).

Data Info.			*Benders-dual algorithm*															
IF	*OR*	*Sur.*	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	TC	$TCLI$	*Time (sec)*	*Ite.*	LB^*_{BD}	UB^*_{BD}	Gap^*_{BD} (%)	$\Delta(UB^*)$ (%)
1.1	1	80	43	132	199	125	31	99	523	512	1663	1217	5703	256	983	985	0.19	0.16
	2	163	109	207	345	108	64	120	1354	155	2462	1851	12149	248	1913	1951	1.94	1.21
	3	241	89	229	310	32	95	134	1170	76	2135	1576	14288	229	2719	2786	2.46	2.08
	4	318	103	273	262	8	103	119	755	38	1661	1109	14031	235	3542	3635	2.64	1.99
	5	397	158	483	290	2	151	197	1024	35	2338	1649	14321	235	4349	4475	2.9	3.98
1.3	1	80	75	151	184	147	42	106	621	624	1949	1453	6380	258	1007	1021	1.39	-0.99
	2	163	125	396	259	92	73	160	1346	210	2661	2044	14158	253	1929	2017	4.54	0.19
	3	241	59	400	266	28	119	194	1022	96	2183	1596	14361	237	2742	2891	5.42	1.87
	4	318	135	535	263	9	141	182	841	64	2170	1587	14400	242	3573	3781	5.82	3.16
	5	397	229	655	185	7	121	181	1083	41	2502	1905	14400	238	4388	4655	6.1	4.83
1.5	1	80	117	191	175	146	84	114	719	538	2086	1566	5778	252	1030	1047	1.62	-1.31
	2	163	157	401	239	118	78	166	1535	215	2909	2255	13696	254	1970	2082	5.69	-0.43
	3	241	78	431	160	57	99	216	1068	91	2199	1637	14400	237	2801	2995	6.94	1.1
	4	318	164	504	188	37	137	232	1051	68	2381	1778	14400	241	3653	3930	7.59	2.19
	5	397	267	377	168	19	144	216	1011	55	2256	1537	14400	240	4481	4984	11.18	1.16
1.7	1	80	47	29	27	15	23	14	86	56	297	206	2062	127	1052	1054	0.18	-0.19
	2	163	180	295	167	75	135	115	968	127	2062	1429	13714	252	2021	2117	4.72	0.12
	3	241	115	375	160	62	141	203	1104	87	2246	1626	14400	238	2859	3090	8.11	0.72
	4	318	205	377	178	37	166	202	970	70	2205	1504	13853	245	3728	4112	10.27	0.56
	5	397	312	209	130	13	168	167	625	54	1678	882	14390	247	4581	5141	12.24	0.26
1.9	1	80	73	63	74	32	56	28	193	128	647	468	2603	129	1070	1073	0.27	-0.28
	2	163	274	256	238	54	230	113	1002	152	2319	1621	11568	249	2059	2179	5.77	-0.41
	3	241	212	258	198	38	248	159	838	95	2046	1316	12190	244	2913	3180	9.17	0.12
	4	318	271	276	194	30	240	186	767	85	2049	1236	12844	249	3801	4162	9.51	0.8
	5	397	318	241	126	11	193	217	708	51	1865	1019	14392	245	4683	5141	9.77	0.6
Average		240	157	310	199	52	123	154	895	149	2039	1443	11955	235	2794	2979	5.46	0.94

Table 6.9 – Computational results of the Benders-dual algorithm for instances with a planning horizon of five weeks ($L = 5$).

Data Info.			*Benders-dual algorithm*															
IF	*OR*	*Sur.*	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	$C8$	TC	$TCLI$	*Time (sec)*	*Ite.*	LB^*_{BD}	UB^*_{BD}	Gap^*_{BD} (%)	$\Delta(UB^*)$ (%)
1.1	1	101	74	138	262	103	44	122	765	470	1978	1462	11384	230	1290	1320	2.27	-0.82
	2	202	80	158	336	62	64	118	1171	121	2110	1605	14179	228	2447	2526	3.23	2.36
	3	302	141	207	367	15	125	126	1055	70	2106	1532	14400	219	3632	3728	2.65	2.05
	4	401	236	376	426	4	137	172	1300	61	2713	1998	14400	218	4716	4884	3.56	2.49
	5	503	409	684	361	3	138	154	1375	45	3168	2402	14400	216	5903	6131	3.85	4.23
1.3	1	101	106	171	216	97	41	140	731	508	2009	1479	11104	250	1318	1362	3.39	-1.59
	2	202	108	366	249	60	79	143	1126	166	2297	1764	14400	226	2468	2628	6.48	0.67
	3	302	89	443	292	16	163	197	951	97	2246	1658	14400	217	3625	3876	6.91	2.41
	4	401	192	725	259	3	139	233	1115	68	2734	2059	14400	219	4725	5085	7.62	6
	5	503	473	852	270	6	124	186	1659	53	3623	2866	14400	213	5941	6375	7.31	6.9
1.5	1	101	180	224	245	112	80	142	928	508	2420	1789	10702	254	1346	1393	3.5	-1.72
	2	202	118	357	159	69	72	161	1180	144	2260	1729	14392	231	2528	2712	7.3	0.97
	3	302	208	499	204	40	176	217	1228	111	2682	2036	14255	223	3695	4057	9.75	1.15
	4	401	317	389	247	25	206	291	1395	92	2962	1967	13738	214	4848	5504	13.57	0.49
	5	503	467	185	150	16	159	232	934	70	2212	1204	14396	219	6082	7039	15.73	0.62
1.7	1	101	208	134	135	53	125	71	377	193	1296	809	8242	167	1381	1405	1.73	-0.86
	2	202	85	263	108	56	79	125	853	93	1661	1168	14400	230	2587	2794	8.02	0.39
	3	302	214	421	199	56	210	217	1272	83	2672	1966	14356	227	3788	4154	9.65	1.53
	4	401	305	220	167	23	167	210	813	57	1962	1095	13437	221	4940	5626	13.89	0.32
	5	503	469	226	195	22	181	251	1083	76	2503	1427	14364	214	6164	7039	14.2	0.62
1.9	1	101	248	177	177	79	168	76	461	235	1619	977	8904	157	1404	1432	2.02	-1.17
	2	202	248	225	196	48	217	116	795	123	1968	1308	12555	233	2637	2848	8.01	0.86
	3	302	203	249	161	30	245	161	715	74	1837	1103	11966	226	3850	4252	10.45	0.81
	4	401	327	257	182	25	217	201	814	68	2090	1185	14328	227	5029	5626	11.86	0.46
	5	503	505	235	170	23	208	255	1010	77	2484	1391	14393	226	6259	7044	12.53	0.54
Average		302	240	327	229	42	143	173	1004	147	2304	1599	13276	220	3704	4034	7.58	1.19

CHAPTER 7 GENERAL DISCUSSION

In the articles presented in Chapters 4, 5, and 6, we studied the applications of decomposition-based integer programming algorithms in deterministic, stochastic, and robust optimization problems in the healthcare context. In Chapter 4, we proposed a branch-and-price-and-cut algorithm for a weekly operating room planning and scheduling problem. In the second paper, we formulated a daily operating room scheduling problem in the presence of stochastic durations. The deterministic version of the daily operating room scheduling problem is a simplified version of the planning and scheduling problem studied in Chapter 4. However, we realized that the stochastic daily operating room scheduling problem is even more challenging than the deterministic operating room planning and scheduling problem presented in Chapter 4. Moreover, the second paper shed light on how we can formulate operating room scheduling problems as vehicle routing problems. The structure of the problem studied in Chapter 6 was somewhat different from those in Chapters 4 and 5. However, this chapter was a complement of Chapters 4 and 5 as it showed that decomposition-based integer programming algorithms could be also applied in robust optimization problems successfully.

The main common point in the presented articles is that the applications of all proposed models are in healthcare. In the first article, we focused on an operating room scheduling problem. In the second paper, we studied a vehicle routing problem with applications in home-health care and operating room scheduling problems. Also, in the third paper, we considered a nurse planning problem as the main application of the proposed two-stage robust optimization model. This problem was also closely relevant to the problems studied in the first two articles as the workload of nurses depended on the number of patients brought to the ward from operating rooms. The presented papers together show that applying decomposition-based integer programming algorithms are useful in healthcare planning, scheduling, and routing problems.

In all presented articles, we provided extensive computational results in order to evaluate the efficiency of the proposed algorithms. Moreover, we compared all proposed algorithms and models with some other algorithms that either were taken from the literature or we developed ourselves.

Another interesting point about the proposed algorithms is that setting stopping conditions for parts of them was so critical and important. In the column generation algorithm proposed in Chapter 4, we proposed a time-based stopping condition for optimizing subproblems formulated by the constraint programming model. Similarly, in Chapter 5, we proposed an optimality-gap-based stopping condition for the master problem of the proposed L-shaped algorithm. Also, as discussed in Chapter 6, one of the main contributions of the third paper was to develop novel stopping conditions for both the master problem and subproblem of the proposed algorithms.

CHAPTER 8 CONCLUSION AND RECOMMENDATIONS

In this dissertation, we studied the application of decomposition-based integer programming, stochastic programming, and robust optimization algorithms in three healthcare planning, scheduling, and routing problems. In the following, we first present a summary of the contributions achieved by this dissertation. Then we highlight its limitations and possible future research directions.

8.1 Contributions

In the first essay of this dissertation, we addressed an integrated operating room planning and scheduling problem by simultaneously considering the assignment of surgeries to operating rooms and their daily scheduling over a week. We considered many practical details such as the maximum daily working time of surgeons, prevention from the overlapping of surgeries performed by the same surgeon, sequence-dependent cleaning times, and surgery deadlines. We developed an integer programming model based on pattern variables for operating rooms' schedules and proposed a branch-and-price-and-cut solution algorithm that applied a constraint programming model for the subproblems. We significantly improved the efficiency of the algorithm by devising some dominance rules and a fast-infeasibility-detection algorithm for the subproblems. Computational results demonstrated that the proposed approach was capable of finding solutions with an optimality gap of 2.81% for instances with 40 to 120 surgeries. It also considerably outperformed an integer programming model from the literature and a pure constraint programming model that we formulated.

As the second essay of this dissertation, for the first time, we considered vehicle routing problems with synchronized visits (VRPS) and stochastic/time-dependent travel and service times. We studied two healthcare applications of VRPS. The first application was a home-health care scheduling problem with stochastic/time-dependent travel and service times and the second one was an operating rooms scheduling problem with stochastic anesthesia, cleaning and surgery durations. To the best of our knowledge, it was the first time an operating room scheduling problem was formulated as a VRPS.

To formulate the stochastic version of VRPS, we developed a two-stage stochastic programming model with integer variables in both stages. In contrast to the deterministic models in the VRPS literature, our proposed formulation did not have any big-M constraints. We obtained this advantage by time discretization that resulted in a large number of second-stage integer variables. We proved that the integrality constraints on second-stage variables were redundant and could be relaxed. Having continuous variables in the second stage, we applied the L-shaped algorithm and its branch-and-cut implementation as solution methods. Moreover, we improved the proposed approach by devising valid inequalities and a lower bounding functional. We also analyzed the structure of subproblems in the L-shaped algorithm and proposed a solution method for them that was faster than standard linear programming algorithms. Furthermore, we discussed how to modify the proposed formulation to solve VRPS with deterministic time-dependent travel and service times.

Computational experiments revealed that, in the home-health care scheduling problem, the branch-and-cut algorithm solved instances with 15 patients and 10% to 30% of synchronized visits to optimality. In addition, it found solutions with an average optimality gap of 3.57% for instances with 20 patients. In the stochastic operating room scheduling problem, the branch-and-cut algorithm was capable of finding optimal solutions for instances with 20 surgeries. This is a considerable improvement over the state-of-art algorithm that reports on instances with 11 surgeries. Moreover, the modified model for the time-dependent problem obtained optimal solutions for a large portion of home-health care scheduling instances with 30 to 60 patients and 10% to 40% of synchronized visits.

In the third essay, we considered a class of two-stage robust optimization problems with integer adversarial variables. The main application of this class of problems is in two-stage robust resource planning problems where tasks' arrivals and durations are uncertain. We considered a two-stage nurse planning problem as an example of such robust resource planning problems. In this problem, there was uncertainty in the arrivals of patients from ICUs to wards and in their length of stays.

We applied Dantzig-Wolfe decomposition to exploit the block-diagonal structure of the coefficient matrix in the recourse model and reformulated the problem to single-stage robust problem. We developed a Benders algorithm for the reformulated model where the master problem and subproblem in the Benders algorithm were both mixed integer programs. As it was computationally unwieldy to solve these mixed integer programs to optimality at each iteration of the algorithm, we devised novel stopping conditions for them and proved the convergence to optimal solutions in the presence of these conditions. Moreover, we proposed a heuristic algorithm called dual algorithm. The underlying idea of this algorithm was to dualize the linear programming relaxation of the adversarial problem in the reformulated model and to generate cuts iteratively to shape the convex hull of the uncertainty set. We also created a more powerful algorithm by combining the Benders algorithm with the dual algorithm. Extensive computational experiments on the nurse planning problem showed that the hybrid algorithm was capable of finding high quality solutions in instances with up to 500 patients in a planning horizon of five weeks.

8.2 Limitations and future research directions

We believe that future research can extend the first essay in several ways. First, we did not take into account upstream and downstream constraints (e.g. beds in preparatory and recovery rooms). To address this issue, one may need to add new variables to formulate the patient scheduling in pre- and post-operating-room stages. Another future research direction is to consider the same integrated operating room planning and scheduling problem where surgery durations are stochastic. We can consider this research as an extension of the second essay where we addressed an operating room scheduling problem with stochastic durations.

In the routing and scheduling problem studied in the second essay, we considered the simplest version of synchronized visits where the required vehicles must be available at the same time in the customer's location. Considering time lags in the synchronization of required vehicles is interesting and can be the subject of future research. This aspect of synchronization is known as temporal constraints and was not addressed in VRPS with stochastic times. In addition, the second essay separately accounted for stochastic/time-dependent travel and service times in vehicles routing problems with synchronized visits. As discussed in Section 5.8, we do not expect that the proposed approach can handle the case that stochasticity and time-dependency are available simultaneously. Future studies can focus on dealing with these aspects at the same time.

The proposed solution approach in the third essay is useful when the number of recourse decision variables after reformulating the problem is limited. However, in many applications, this condition is not satisfied and we cannot apply the proposed approach directly. Future studies can extend the proposed approach by developing an efficient column-and-constraint generation algorithm that generates recourse decision variables and relevant constraints when they are needed. Moreover, in the second essay, we specifically focused on two-stage robust optimization problems with integer adversarial variables. Future research can extend our approach for multi-stage problems.

REFERENCES

Adulyasak, Yossiri, Patrick Jaillet. 2015. Models and algorithms for stochastic and robust vehicle routing with deadlines. *Transportation Science* **50**(2) 608–626.

Agra, Agostinho, Marielle Christiansen, Rosa Figueiredo, Lars Magnus Hvattum, Michael Poss, Cristina Requejo. 2013. The robust vehicle routing problem with time windows. *Computers & Operations Research* **40**(3) 856–866.

Aickelin, Uwe, Kathryn A. Dowsland. 2000. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling* **3**(3) 139–153.

Akjiratikarl, Chananes, Pisal Yenradee, Paul R Drake. 2007. Pso-based algorithm for home care worker scheduling in the uk. *Computers & Industrial Engineering* **53**(4) 559–583.

An, Yu, Bo Zeng. 2015. Exploring the modeling capacity of two-stage robust optimization : Variants of robust unit commitment model. *IEEE Transactions on Power Systems* **30**(1) 109–122.

An, Yu, Bo Zeng, Yu Zhang, Long Zhao. 2014. Reliable p-median facility location problem : two-stage robust models and algorithms. *Transportation Research Part B : Methodological* **64** 54–72.

Ang, Marcus, Yun Fong Lim, Melvyn Sim. 2012. Robust storage assignment in unit-load warehouses. *Management Science* **58**(11) 2114–2130.

Ardestani-Jaafari, Amir, Erick Delage. 2016. Robust optimization of sums of piecewise linear functions with application to inventory problems. *Operations Research* **64**(2) 474–494.

Arnaout, Jean-Paul M, Sevag Kulbashian. 2008. Maximizing the utilization of operating rooms with stochastic times using simulation. *Proceedings of the 40th conference on winter simulation*. Winter Simulation Conference, 1617–1623.

Atamtürk, Alper. 2005. Cover and pack inequalities for (mixed) integer programming. *Annals of Operations Research* **139**(1) 21–38.

Augusto, Vincent, Xiaolan Xie, Viviana Perdomo. 2008. Operating theatre scheduling using lagrangian relaxation. *European Journal of Industrial Engineering* **2**(2) 172–189.

Augusto, Vincent, Xiaolan Xie, Viviana Perdomo. 2010. Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers & Industrial Engineering* **58**(2) 231–238.

Bagheri, Mohsen, Ali Gholinejad Devin, Azra Izanloo. 2016. An application of stochastic programming method for nurse scheduling problem in real word hospital. *Computers & Industrial Engineering* **96** 192–200.

Baligh, Helmy H, Danny J Laughhunn. 1969. An economic and linear model of the hospital. *Health Services Research* **4**(4) 293.

Balseiro, SR, Irene Loiseau, Juan Ramonet. 2011. An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research* **38**(6) 954–966.

Bard, Jonathan F, Hadi W Purnomo. 2005. Preference scheduling for nurses using column generation. *European Journal of Operational Research* **164**(2) 510–534.

Bard, Jonathan F, Hadi W Purnomo. 2006. Incremental changes in the workforce to accommodate changes in demand. *Health Care Management Science* **9**(1) 71–85.

Bard, Jonathan F, Hadi W Purnomo. 2007. Cyclic preference scheduling of nurses using a lagrangian-based heuristic. *Journal of Scheduling* **10**(1) 5–23.

Bard, Jonathan F, Yufen Shao, Ahmad I Jarrah. 2014a. A sequential grasp for the therapist routing and scheduling problem. *Journal of Scheduling* **17**(2) 109–133.

Bard, Jonathan F, Yufen Shao, Xiangtong Qi, Ahmad I Jarrah. 2014b. The traveling therapist scheduling problem. *IIE Transactions* **46**(7) 683–706.

Barnhart, Cynthia, Christopher A Hane, Pamela H Vance. 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research* **48**(2) 318–326.

Batun, Sakine, Brian T Denton, Todd R Huschka, Andrew J Schaefer. 2011. Operating room pooling and parallel surgery processing under uncertainty. *INFORMS Journal on Computing* **23**(2) 220–237.

Bäumelt, Zdeněk, Jan Dvořák, Přemysl Šucha, Zdeněk Hanzálek. 2016. A novel approach for nurse rerostering based on a parallel algorithm. *European Journal of Operational Research* **251**(2) 624–639.

Beliën, Jeroen, Erik Demeulemeester. 2008. A branch-and-price approach for integrating nurse and surgery scheduling. *European Journal of Operational Research* **189**(3) 652–668.

Bellanti, F, Giuliana Carello, Federico Della Croce, Roberto Tadei. 2004. A greedy-based neighborhood search approach to a nurse rostering problem. *European Journal of Operational Research* **153**(1) 28–40.

Ben-Tal, Aharon, Golany Boaz, Shtern Shimrit. 2009. Robust multi-echelon multi-period inventory control. *European Journal of Operational Research* **199**(3) 922–935.

Ben-Tal, Aharon, Boaz Golany, Arkadi Nemirovski, Jean-Philippe Vial. 2005. Retailer-supplier flexible commitments contracts : a robust optimization approach. *Manufacturing & Service Operations Management* **7**(3) 248–271.

Ben-Tal, Aharon, Alexander Goryashko, Elana Guslitzer, Arkadi Nemirovski. 2004. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming* **99**(2) 351–376.

Ben-Tal, Aharon, Arkadi Nemirovski. 1999. Robust solutions of uncertain linear programs. *Operations Research Letters* **25**(1) 1–13.

Bennett, Ashlea R, Alan L Erera. 2011. Dynamic periodic fixed appointment scheduling for home health. *IIE Transactions on Healthcare Systems Engineering* **1**(1) 6–19.

Bertsimas, Dimitris, Constantine Caramanis. 2010. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control* **55**(12) 2751–2766.

Bertsimas, Dimitris, Iain Dunning. 2016. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research* **64**(4) 980–998.

Bertsimas, Dimitris, Iain Dunning, Miles Lubin. 2016. Reformulation versus cutting-planes for robust optimization. *Computational Management Science* **13**(2) 195–217.

Bertsimas, Dimitris, Dan Andrei Iancu, Pablo Parrilo, et al. 2011. A hierarchy of near-optimal policies for multistage adaptive optimization. *IEEE Transactions on Automatic Control* **56**(12) 2809–2824.

Bertsimas, Dimitris, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, Tongxin Zheng. 2013. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems* **28**(1) 52–63.

Bertsimas, Dimitris, Melvyn Sim. 2004. The price of robustness. *Operations Research* **52**(1) 35–53.

Binart, Sixtine, Pierre Dejax, Michel Gendreau, Frédéric Semet. 2016. A 2-stage method for a field service routing problem with stochastic travel and service times. *Computers & Operations Research* **65** 64–75.

Blake, John T, Michael W Carter. 2002. A goal programming approach to strategic resource allocation in acute care hospitals. *European Journal of Operational Research* **140**(3) 541–561.

Bowers, John, Helen Cheyne, Gillian Mould, Miranda Page. 2015. Continuity of care in community midwifery. *Health Care Management Science* **18**(2) 195–204.

Braekers, Kris, Richard F Hartl, Sophie N Parragh, Fabien Tricoire. 2016. A bi-objective home care scheduling problem : Analyzing the trade-off between costs and client inconvenience. *European Journal of Operational Research* **248**(2) 428–443.

Bredström, David, Mikael Rönnqvist. 2008. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research* **191**(1) 19–31.

Brucker, Peter, Rong Qu, Edmund Burke. 2011. Personnel scheduling : Models and complexity. *European Journal of Operational Research* **210**(3) 467–473.

Burke, Edmund K, Timothy Curtois, Gerhard Post, Rong Qu, Bart Veltman. 2008. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research* **188**(2) 330–341.

Burke, Edmund K, Timothy Curtois, Rong Qu, G Vanden Berghe. 2010a. A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society* **61**(11) 1667–1679.

Burke, Edmund K, Timothy Curtois, L Fijn van Draat, Jk Van Ommeren, G Post. 2011. Progress control in iterated local search for nurse rostering. *Journal of the Operational Research Society* **62**(2) 360–367.

Burke, Edmund K, Patrick De Causmaecker, Greet Vanden Berghe, Hendrik Van Landeghem. 2004. The state of the art of nurse rostering. *Journal of Scheduling* **7**(6) 441–499.

Burke, Edmund K, Jingpeng Li, Rong Qu. 2010b. A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems. *European Journal of Operational Research* **203**(2) 484–493.

Cappanera, Paola, Maria Grazia Scutellà. 2015. Joint assignment, scheduling, and routing models to home care optimization : a pattern-based approach. *Transportation Science* **49**(4) 830–852.

Cardoen, Brecht, Erik Demeulemeester, Jeroen Beliën. 2009a. Optimizing a multiple objective surgical case sequencing problem. *International Journal of Production Economics* **119**(2) 354–366.

Cardoen, Brecht, Erik Demeulemeester, Jeroen Beliën. 2009b. Sequencing surgical cases in a day-care environment : an exact branch-and-price approach. *Computers & Operations Research* **36**(9) 2660–2669.

Cardoen, Brecht, Erik Demeulemeester, Jeroen Beliën. 2010. Operating room planning and scheduling : A literature review. *European Journal of Operational Research* **201**(3) 921–932.

Carraway, Robert L, Thomas L Morin, Herbert Moskowitz. 1989. Generalized dynamic programming for stochastic combinatorial optimization. *Operations Research* **37**(5) 819–829.

Castillo-Salazar, J Arturo, Dario Landa-Silva, Rong Qu. 2016. Workforce scheduling and routing problems : literature survey and computational study. *Annals of Operations Research* **239**(1) 39–67.

Chan, Timothy CY, Zuo-Jun Max Shen, Auyon Siddiq. 2015. Robust facility location under demand location uncertainty. *arXiv preprint arXiv :1507.04397* .

Cheang, Brenda, Haibing Li, Andrew Lim, Brian Rodrigues. 2003. Nurse rostering problems—a bibliographic survey. *European Journal of Operational Research* **151**(3) 447–460.

Chen, Lu, Minh Hoàng Hà, André Langevin, Michel Gendreau. 2014. Optimizing road network daily maintenance operations with stochastic service and travel times. *Transportation Research Part E : Logistics and Transportation Review* **64** 88–102.

Chen, Xin, Wenchuan Wu, Boming Zhang. 2016. Robust restoration method for active distribution networks. *IEEE Transactions Power Systems* **31**(5).

Chen, Xin, Yuhan Zhang. 2009. Uncertain linear programs : Extended affinely adjustable robust counterparts. *Operations Research* **57**(6) 1469–1482.

Choi, Sangdo, Wilbert E Wilhelm. 2014. On capacity allocation for operating rooms. *Computers & Operations Research* **44** 174–184.

Chu, Geoffrey, Peter J Stuckey. 2009. Minimizing the maximum number of open stacks by customer search. *International Conference on Principles and Practice of Constraint Programming*. Springer, 242–257.

Chu, Geoffrey, Peter J Stuckey. 2012. A generic method for identifying and exploiting dominance relations. *Principles and Practice of Constraint Programming*. Springer, 6–22.

Dabia, Said, Stefan Ropke, Tom Van Woensel, Ton De Kok. 2013. Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science* **47**(3) 380–396.

Danandeh, Anna, Long Zhao, Bo Zeng. 2014. Job scheduling with uncertain local generation in smart buildings : Two-stage robust approach. *IEEE Transactions on Smart Grid* **5**(5) 2273–2282.

Dantzig, George B, Philip Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8**(1) 101–111.

Dashti, Hossein, Antonio J Conejo, Ruiwei Jiang, Jianhui Wang. 2016. Weekly two-stage robust generation scheduling for hydrothermal power systems. *IEEE Transactions on Power Systems* **31**(6) 4554–4564.

Day, Robert, Robert Garfinkel, Steven Thompson. 2012. Integrated block sharing : A win–win strategy for hospitals and surgeons. *Manufacturing & Service Operations Management* **14**(4) 567–583.

De Rosa, Barbara, Gennaro Improta, Gianpaolo Ghiani, Roberto Musmanno. 2002. The arc routing and scheduling problem with transshipment. *Transportation Science* **36**(3) 301–313.

Delage, Erick, Dan A. Iancu. 2015. Robust multistage decision making. *Tutorials in Operations Research* 20–46.

Denton, Brian, James Viapiano, Andrea Vogl. 2007. Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health Care Management Science* **10**(1) 13–24.

Denton, Brian T, Oguzhan Alagoz, Allen Holder, Eva K Lee. 2011. Medical decision making : open research challenges. *IIE Transactions on Healthcare Systems Engineering* **1**(3) 161–167.

Desaulniers, Guy, Jacques Desrosiers, Marius M Solomon. 2006. *Column generation*, vol. 5. Springer Science & Business Media.

Di Martinelly, C, P Baptiste, MY Maknoon. 2014. An assessment of the integration of nurse timetable changes with operating room planning and scheduling. *International Journal of Production Research* **52**(24) 7239–7250.

Di Mascolo, Maria, Marie-Laure Espinouse, Can Erdem Ozkan. 2014. Synchronization between human resources in home health care context. *International Conference on Health Care Systems Engineering*. Springer, 73–86.

Ding, Tao, Shiyu Liu, Wei Yuan, Zhaohong Bie, Bo Zeng. 2016. A two-stage robust reactive power optimization considering uncertain wind power integration in active distribution networks. sustainable energy. *IEEE Transactions on Sustainable Energy* **7**(1) 301–311.

Donati, Alberto V, Roberto Montemanni, Norman Casagrande, Andrea E Rizzoli, Luca M Gambardella. 2008. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research* **185**(3) 1174–1191.

Dowsland, Kathryn Anne, Jonathan Mark Thompson. 2000. Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society* **51**(7) 825–833.

Drexl, Michael. 2012. Synchronization in vehicle routing—a survey of vrps with multiple synchronization constraints. *Transportation Science* **46**(3) 297–316.

Drexl, Michael, Hans-Jürgen Sebastian. 2007. On some generalized routing problems. Tech. rep., Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken (NN).

Ehmke, Jan Fabian, Ann Melissa Campbell, Timothy L Urban. 2015. Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research* **240**(2) 539–550.

El Ghaoui, Laurent, Hervé Lebret. 1997. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications* **18**(4) 1035–1064.

El Ghaoui, Laurent, Francois Oustry, Hervé Lebret. 1998. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization* **9**(1) 33–52.

Ernst, Andreas T, Houyuan Jiang, Mohan Krishnamoorthy, David Sier. 2004. Staff scheduling and rostering : A review of applications, methods and models. *European Journal of Operational Research* **153**(1) 3–27.

Errico, Fausto, Guy Desaulniers, Michel Gendreau, Walter Rei, L-M Rousseau. 2016. A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times. *European Journal of Operational Research* **249**(1) 55–66.

Eveborn, Patrik, Patrik Flisberg, Mikael Rönnqvist. 2006. Laps care—an operational system for staff planning of home care. *European Journal of Operational Research* **171**(3) 962–976.

Fahle, Torsten, Stefan Schamberger, Meinolf Sellmann. 2001. Symmetry breaking. *International Conference on Principles and Practice of Constraint Programming*. Springer, 93–107.

Fei, Hongying, Chengbin Chu, Nadine Meskens. 2009. Solving a tactical operating room planning problem by a column-generation-based heuristic procedure with four criteria. *Annals of Operations Research* **166**(1) 91.

Fei, Hongying, Chengbin Chu, Nadine Meskens, Abdelhakim Artiba. 2008. Solving surgical cases assignment problem by a branch-and-price approach. *International Journal of Production Economics* **112**(1) 96–108.

Fei, Hongying, Nadine Meskens, Chengbin Chu. 2006. An operating theatre planning and scheduling problem in the case of a" block scheduling" strategy. *International Conference on Service Systems and Service Management*, vol. 1. IEEE, 422–428.

Fei, Hongying, Nadine Meskens, Chengbin Chu. 2010. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering* **58**(2) 221–230.

Feige, Uriel, Kamal Jain, Mohammad Mahdian, Vahab Mirrokni. 2007. Robust combinatorial optimization with exponential scenarios. *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 439–453.

Fikar, Christian, Patrick Hirsch. 2017. Home health care routing and scheduling : A review. *Computers & Operations Research* **77** 86–95.

Fischetti, Matteo, Michele Monaci. 2012. Cutting plane versus compact formulations for uncertain (integer) linear programs. *Mathematical Programming Computation* **4**(3) 239–273.

Fisher, Marshall L, Arnold J Greenfield, Ramchandran Jaikumar, Joseph T Lester III. 1982. A computerized vehicle routing application. *Interfaces* **12**(4) 42–52.

Focacci, Filippo, Michaela Milano. 2001. Global cut framework for removing symmetries. *International Conference on Principles and Practice of Constraint Programming*. Springer, 77–92.

Fonseca, Raquel J, Berç Rustem. 2012. International portfolio management with affine policies. *European Journal of Operational Research* **223**(1) 177–187.

Franceschetti, Anna, Dorothée Honhon, Tom Van Woensel, Tolga Bektaş, Gilbert Laporte. 2013. The time-dependent pollution-routing problem. *Transportation Research Part B : Methodological* **56** 265–293.

Garcia de la Banda, Maria, Peter J Stuckey, Geoffrey Chu. 2011. Solving talent scheduling with dynamic programming. *INFORMS Journal on Computing* **23**(1) 120–137.

Gerchak, Yigal, Diwakar Gupta, Mordechai Henig. 1996. Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science* **42**(3) 321–334.

Gilmore, Paul C, Ralph E Gomory. 1961. A linear programming approach to the cutting-stock problem. *Operations Research* **9**(6) 849–859.

Goldberg, Andrew V, Robert E Tarjan. 1988. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)* **35**(4) 921–940.

Gómez, Andrés, Ricardo Mariño, Raha Akhavan-Tabatabaei, Andrés L Medaglia, Jorge E Mendoza. 2015. On modeling stochastic travel and service times in vehicle routing. *Transportation Science* **50**(2) 627–641.

Grötschel, Martin, Manfred W Padberg. 1979. On the symmetric travelling salesman problem i : inequalities. *Mathematical Programming* **16**(1) 265–280.

Gu, Zonghao, George L Nemhauser, Martin WP Savelsbergh. 1998. Lifted cover inequalities for 0-1 integer programs : Computation. *INFORMS Journal on Computing* **10**(4) 427–437.

Guerriero, Francesca, Rosita Guido. 2011. Operational research in the management of the operating theatre : a survey. *Health Care Management Science* **14**(1) 89–114.

Guinet, Alain, Sondes Chaabane. 2003. Operating theatre planning. *International Journal of Production Economics* **85**(1) 69–81.

Gul, Serhat, Brian T Denton, John W Fowler, Todd Huschka. 2011. Bi-criteria scheduling of surgical services for an outpatient procedure center. *Production and Operations Management* **20**(3) 406–417.

Gupta, Anupam, Viswanath Nagarajan, R Ravi. 2014. Thresholded covering algorithms for robust and max–min optimization. *Mathematical Programming* **146**(1) 583–615.

Hachicha, Hejer Khlif, Farah Zeghal Mansour. 2016. Two-milp models for scheduling elective surgeries within a private healthcare facility. *Health Care Management Science* 1–17.

Haghani, Ali, Soojung Jung. 2005. A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research* **32**(11) 2959–2986.

Han, Jinil, Chungmok Lee, Sungsoo Park. 2013. A robust scenario approach for the vehicle routing problem with uncertain travel times. *Transportation Science* **48**(3) 373–390.

Hanasusanto, Grani Adiwena, Daniel Kuhn, Wolfram Wiesemann. 2014. Two-stage robust integer programming. *Optimization Online* .

Hans, Erwin, Gerhard Wullink, Mark Van Houdenhoven, Geert Kazemier. 2008. Robust surgery loading. *European Journal of Operational Research* **185**(3) 1038–1050.

Hashemi Doulabi, Seyed Hossein, Louis-Martin Rousseau, Gilles Pesant. 2014. A constraint programming-based column generation approach for operating room planning and scheduling. *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*. Springer, 455–463.

Hashimoto, Hideki, Mutsunori Yagiura, Toshihide Ibaraki. 2008. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization* **5**(2) 434–456.

He, Fang, Rong Qu. 2012. A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research* **39**(12) 3331–3343.

Hiermann, Gerhard, Matthias Prandtstetter, Andrea Rendl, Jakob Puchinger, Günther R Raidl. 2015. Metaheuristics for solving a multimodal home-healthcare scheduling problem. *Central European Journal of Operations Research* **23**(1) 89–113.

Hill, AV, VA Mabert, DW Montgomery. 1988. A decision support system for the courier vehicle scheduling problem. *Omega* **16**(4) 333–345.

Hsu, Vernon Ning, Renato de Matta, Chung-Yee Lee. 2003. Scheduling patients in an ambulatory surgical center. *Naval Research Logistics* **50**(3) 218–238.

Huang, Yixiao, Lei Zhao, Tom Van Woensel, Jean-Philippe Gross. 2017. Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B : Methodological* **95** 169–195.

Hulshof, Peter JH, Nikky Kortbeek, Richard J Boucherie, Erwin W Hans, Piet JM Bakker. 2012. Taxonomic classification of planning decisions in health care : a structured review of the state of the art in OR/MS. *Health Systems* **1**(2) 129–175.

IBM. 2015. *CPLEX Optimization Studio CPLEX User's Manual, Version 12 Release 6*.

Ichoua, Soumia, Michel Gendreau, Jean-Yves Potvin. 2003. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* **144**(2) 379–396.

Jabali, O, T Woensel, AG de Kok. 2012. Analysis of travel times and co2 emissions in time-dependent vehicle routing. *Production and Operations Management* **21**(6) 1060–1074.

Jaillet, Patrick, Jin Qi, Melvyn Sim. 2016. Routing optimization under uncertainty. *Operations Research* **64**(1) 186–200.

Jaumard, Brigitte, Hamed Pouya, Rami Fahim, Andres Barrios. 2016. Planning network migration. *International Conference on Communications*. IEEE, 1–6.

Jaumard, Brigitte, Frederic Semet, Tsevi Vovor. 1998. A generalized linear programming model for nurse scheduling. *European Journal of Operational Research* **107**(1) 1–18.

Jebali, Aida, Atidel B Hadj Alouane, Pierre Ladet. 2006. Operating rooms scheduling. *International Journal of Production Economics* **99**(1) 52–62.

Jebali, Aida, Ali Diabat. 2015. A stochastic model for operating room planning under capacity constraints. *International Journal of Production Research* **53**(24) 7252–7270.

Jensen, Johan Ludwig William Valdemar. 1906. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica* **30**(1) 175–193.

Jiang, Ruiwei, Jianhui Wang, Muhong Zhang, Yongpei Guan. 2013. Two-stage minimax regret robust unit commitment. *IEEE Transactions on Power Systems* **28**(3) 2271–2282.

Jiang, Ruiwei, Muhong Zhang, Guang Li, Yongpei Guan. 2012. Benders' decomposition for the two-stage security constrained robust unit commitment problem. *IIE Annual Conference*. Institute of Industrial Engineers, 1.

Jung, Soojung, Ali Haghani. 2001. Genetic algorithm for the time-dependent vehicle routing problem. *Transportation Research Record : Journal of the Transportation Research Board* (1771) 164–171.

Jünger, Michael, Giovanni Rinaldi, Stefan Thienel. 2000. Practical performance of efficient minimum cut algorithms. *Algorithmica* **26**(1) 172–195.

Kao, Edward PC. 1978. A preference order dynamic program for a stochastic traveling salesman problem. *Operations Research* **26**(6) 1033–1045.

Kenyon, Astrid S, David P Morton. 2003. Stochastic vehicle routing with random travel times. *Transportation Science* **37**(1) 69–82.

Kim, Kibaek, Sanjay Mehrotra. 2015. A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. *Operations Research* **63**(6) 1431–1451.

Korf, Richard E. 2004. Optimal rectangle packing : New results. *ICAPS*. 142–149.

Kuo, Yiyo. 2010. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering* **59**(1) 157–165.

Lamiri, Mehdi, Xiaolan Xie, Alexandre Dolgui, Frédéric Grimaud. 2008a. A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research* **185**(3) 1026–1037.

Lamiri, Mehdi, Xiaolan Xie, Shuguang Zhang. 2008b. Column generation approach to operating theater planning with elective and emergency patients. *IIE Transactions* **40**(9) 838–852.

Laporte, Gilbert, Francois Louveaux, Hélène Mercure. 1992. The vehicle routing problem with stochastic travel times. *Transportation Science* **26**(3) 161–170.

Latorre-Núñez, Guillermo, Armin Lüer-Villagra, Vladimir Marianov, Carlos Obreque, Francisco Ramis, Liliana Neriz. 2016. Scheduling operating rooms with consideration of all resources, post anesthesia beds and emergency surgeries. *Computers & Industrial Engineering* **97** 248–257.

Lee, Changhyeok, Cong Liu, Sanjay Mehrotra, Zhaohong Bie. 2015. Robust distribution network reconfiguration. *IEEE Transactions on Smart Grid* **6**(2) 836–842.

Lee, Changhyeok, Cong Liu, Sanjay Mehrotra, Mohammad Shahidehpour. 2014. Modeling transmission line constraints in two-stage robust unit commitment problem. *IEEE Transactions on Power Systems* **29**(3) 1221–1231.

Lee, Chungmok, Kyungsik Lee, Sungsoo Park. 2012. Robust vehicle routing problem with deadlines and travel time/demand uncertainty. *Journal of the Operational Research Society* **63**(9) 1294–1306.

Leong, G, J Wilson, A Charlett. 2006. Duration of operation as a risk factor for surgical site infection : comparison of english and us data. *Journal of Hospital Infection* **63**(3) 255–262.

Li, Xiangyong, Peng Tian, Stephen CH Leung. 2010. Vehicle routing problems with time windows and stochastic travel and service times : Models and algorithm. *International Journal of Production Economics* **125**(1) 137–145.

Li, Yanzhi, Andrew Lim, Brian Rodrigues. 2005. Manpower allocation with time windows and job-teaming constraints. *Naval Research Logistics* **52**(4) 302–311.

Li, Zhigang, Wenchuan Wu, Mohammad Shahidehpour, Boming Zhang. 2015. Adaptive robust tie-line scheduling considering wind power uncertainty for interconnected power systems. *IEEE Transactions on Power Systems* **31**(4) 2701–2713.

Li, Zhigang, Wenchuan Wu, Bo Zeng, Mohammad Shahidehpour, Boming Zhang. 2017. Decentralized contingency-constrained tie-line scheduling for multi-area power grids. *IEEE Transactions on Power Systems* **32**(1) 354–367.

Lim, Andrew, Brian Rodrigues, Lei Song. 2004. Manpower allocation with time windows. *Journal of the Operational Research Society* **55**(11) 1178–1186.

Liu, Ya, Chengbin Chu, Kanliang Wang. 2011. A new heuristic algorithm for the operating room scheduling problem. *Computers & Industrial Engineering* **61**(3) 865–871.

Lü, Zhipeng, Jin-Kao Hao. 2012. Adaptive neighborhood search for nurse rostering. *European Journal of Operational Research* **218**(3) 865–876.

Lysgaard, Jens, Adam N Letchford, Richard W Eglese. 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming* **100**(2) 423–445.

Maenhout, Broos, Mario Vanhoucke. 2010. Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem. *Journal of Scheduling* **13**(1) 77–93.

Maenhout, Broos, Mario Vanhoucke. 2011. An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research* **38**(10) 1400–1411.

Malandraki, Chryssi, Mark S Daskin. 1992. Time dependent vehicle routing problems : Formulations, properties and heuristic algorithms. *Transportation Science* **26**(3) 185–200.

Malandraki, Chryssi, Robert B Dial. 1996. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research* **90**(1) 45–55.

Mancilla, Camilo, Robert Storer. 2012. A sample average approximation approach to stochastic appointment sequencing and scheduling. *IIE Transactions* **44**(8) 655–670.

Mankowska, Dorota Slawa, Frank Meisel, Christian Bierwirth. 2014. The home health care routing and scheduling problem with interdependent services. *Health Care Management Science* **17**(1) 15–30.

Marcon, Eric, Franklin Dexter. 2006. Impact of surgical sequencing on post anesthesia care unit staffing. *Health Care Management Science* **9**(1) 87–98.

Marques, Inês, M Eugénia Captivo. 2017. Different stakeholders' perspectives for a surgical case assignment problem : Deterministic and robust approaches. *European Journal of Operational Research* **261**(1) 260–278.

Marques, Inês, M Eugénia Captivo, Margarida Vaz Pato. 2012. An integer programming approach to elective surgery scheduling. *OR Spectrum* **34**(2) 407–427.

Martin, Simon, Djamila Ouelhadj, Pieter Smet, Greet Vanden Berghe, Ender ÖZcan. 2013. Cooperative search for fair nurse rosters. *Expert Systems with Applications* **40**(16) 6674–6683.

May, Jerrold H, William E Spangler, David P Strum, Luis G Vargas. 2011. The surgical scheduling problem : Current research and future opportunities. *Production and Operations Management* **20**(3) 392–405.

M'Hallah, Rym, AH Al-Roomi. 2014. The planning and scheduling of operating rooms : A simulation approach. *Computers & Industrial Engineering* **78** 235–248.

Miranda, Douglas Moura, Samuel Vieira Conceição. 2016. The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications* **64** 104–116.

Mısır, Mustafa, Pieter Smet, Greet Vanden Berghe. 2015. An analysis of generalised heuristics for vehicle routing and personnel rostering problems. *Journal of the Operational Research Society* **66**(5) 858–870.

Molina-Pariente, Jose M, Victor Fernandez-Viagas, Jose M Framinan. 2015a. Integrated operating room planning and scheduling problem with assistant surgeon dependent surgery durations. *Computers & Industrial Engineering* **82** 8–20.

Molina-Pariente, Jose M, Erwin W Hans, Jose M Framinan, Tomas Gomez-Cia. 2015b. New heuristics for planning operating rooms. *Computers & Industrial Engineering* **90** 429–443.

Monteiro, Thibaud, Nadine Meskens, Tao Wang. 2015. Surgical scheduling with antagonistic human resource objectives. *International Journal of Production Research* **53**(24) 7434–7449.

Moz, Margarida, Margarida Vaz Pato. 2007. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research* **34**(3) 667–691.

Nguyen, Tri-Dung, Andrew W Lo. 2012. Robust ranking and portfolio optimization. *European Journal of Operational Research* **221**(2) 407–416.

Nickel, Stefan, Michael Schröder, Jörg Steeg. 2012. Mid-term and short-term planning support for home health care services. *European Journal of Operational Research* **219**(3) 574–587.

Ogulata, S Noyan, Rızvan Erol. 2003. A hierarchical multiple criteria mathematical programming approach for scheduling general surgery operations in large hospitals. *Journal of Medical Systems* **27**(3) 259–270.

Ouorou, Adam. 2013. Tractable approximations to a robust capacity assignment model in telecommunications under demand uncertainty. *Computers & Operations Research* **40**(1) 318–327.

Oyola, Jorge, Halvard Arntzen, David L Woodruff. 2016. The stochastic vehicle routing problem, a literature review, part ii : solution methods. *EURO Journal on Transportation and Logistics* 1–40.

Ozkarahan, Irem. 2000. Allocation of surgeries to operating rooms by goal programing. *Journal of Medical Systems* **24**(6) 339–378.

Padberg, Manfred, Giovanni Rinaldi. 1990. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming* **47**(1) 19–36.

Paraskevopoulos, Dimitris C, Gilbert Laporte, Panagiotis P Repoussis, Christos D Tarantilis. 2016. Resource constrained routing and scheduling : Review and research prospects. *CIRRELT report* .

Perdomo, Viviana, Vincent Augusto, Xiaolan Xie. 2006. Operating theatre scheduling using lagrangian relaxation. *International Conference on Service Systems and Service Management*, vol. 2. IEEE, 1234–1239.

Persson, Marie, Jan A Persson. 2009. Health economic modeling to support surgery management at a swedish hospital. *Omega* **37**(4) 853–863.

Poss, Michael, Christian Raack. 2013. Affine recourse for the robust network design problem : Between static and dynamic routing. *Networks* **61**(2) 180–198.

Postek, Krzysztof, Dick Den Hertog. 2014. Multi-stage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *CentER Discussion Paper Series* .

Prestwich, Steven, J Christopher Beck. 2004. Exploiting dominance in three symmetric problems. *4th International Workshop on Symmetry and Constraint Satisfaction Problems*. Citeseer, 63–70.

Proll, Les, Barbara Smith. 1998. Integer linear programming and constraint programming approaches to a template design problem. *INFORMS Journal on Computing* **10**(3) 265–275.

Punnakitikashem, Prattana, Jay M Rosenberger, Deborah Buckley Behan. 2008. Stochastic programming for nurse assignment. *Computational Optimization and Applications* **40**(3) 321–349.

Purnomo, Hadi W, Jonathan F Bard. 2007. Cyclic preference scheduling for nurses using branch and price. *Naval Research Logistics* **54**(2) 200–220.

Rahimian, Erfan, Kerem Akartunalı, John Levine. 2017a. A hybrid integer and constraint programming approach to solve nurse rostering problems. *Computers & Operations Research* **82** 83–94.

Rahimian, Erfan, Kerem Akartunalı, John Levine. 2017b. A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research* **258**(2) 411–423.

Rasmussen, Matias Sevel, Tor Justesen, Anders Dohn, Jesper Larsen. 2012. The home care crew scheduling problem : Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research* **219**(3) 598–610.

Remli, Nabila, Monia Rekik. 2013. A robust winner determination problem for combinatorial transportation auctions under uncertain shipment volumes. *Transportation Research Part C : Emerging Technologies* **35** 204–217.

Roland, Benoît, Chr Di Martinelly, Fouad Riane, Yves Pochet. 2010. Scheduling an operating theatre under human resource constraints. *Computers & Industrial Engineering* **58**(2) 212–220.

Roland, Benoit, Christine Di Martinelly, Fouad Riane. 2006. Operating theatre optimization : A resource-constrained based solving approach. *International Conference on Service Systems and Service Management*, vol. 1. IEEE, 443–448.

Römer, Michael, Taïeb Mellouli. 2016. Future demand uncertainty in personnel scheduling : investigating deterministic lookahead policies using optimization and simulation. *30th European Conference on Modelling and Simulation*. ECMS.

Roshanaei, Vahid, Curtiss Luong, Dionne M Aleman, David Urbach. 2017. Propagating logic-based benders' decomposition approaches for distributed operating room scheduling. *European Journal of Operational Research* **257**(2) 439–455.

Ryan, David M, Brian A Foster. 1981. An integer programming approach to scheduling. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* 269–280.

Saadouli, Hadhemi, Badreddine Jerbi, Abdelaziz Dammak, Lotfi Masmoudi, Abir Bouaziz. 2015. A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery department. *Computers & Industrial Engineering* **80** 72–79.

Samudra, Michael, Carla Van Riet, Erik Demeulemeester, Brecht Cardoen, Nancy Vansteenkiste, Frank E Rademakers. 2016. Scheduling operating rooms : achievements, challenges and pitfalls. *Journal of Scheduling* **19**(5) 493–525.

Saremi, Alireza, Payman Jula, Tarek ElMekkawy, G Gary Wang. 2013. Appointment scheduling of outpatient surgical services in a multistage operating room department. *International Journal of Production Economics* **141**(2) 646–658.

Shao, Yufen, Jonathan F Bard, Ahmad I Jarrah. 2012. The therapist routing and scheduling problem. *IIE Transactions* **44**(10) 868–893.

Shen, Yu, Jean-Yves Potvin, Jean-Marc Rousseau, Serge Roy. 1995. A computer assistant for vehicle dispatching with learning capabilities. *Annals of Operations Research* **61**(1) 189–211.

Siddiq, Auyon. 2013. Robust facility location under demand location uncertainty. Master's thesis, University of Toronto.

Sier, D, P Tobin, C McGurk. 1997. Scheduling surgical procedures. *Journal of the Operational Research Society* **48**(9) 884–891.

Sniedovich, Moshe. 1981. Technical note – analysis of a preference order traveling salesman problem. *Operations Research* **29**(6) 1234–1237.

Soler, David, José Albiach, Eulalia MartíNez. 2009. A way to optimally solve a time-dependent vehicle routing problem with time windows. *Operations Research Letters* **37**(1) 37–42.

Souyris, Sebastián, Cristián E Cortés, Fernando Ordóñez, Andres Weintraub. 2013. A robust optimization approach to dispatching technicians under stochastic service times. *Optimization Letters* **7**(7) 1549–1568.

Soyster, Allen L. 1973. Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research* **21**(5) 1154–1157.

Taş, D, Michel Gendreau, N Dellaert, Tom Van Woensel, AG De Kok. 2014b. Vehicle routing with soft time windows and stochastic travel times : A column generation and branch-and-price solution approach. *European Journal of Operational Research* **236**(3) 789–799.

Taş, Duygu, Nico Dellaert, Tom Van Woensel, Ton De Kok. 2013. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research* **40**(1) 214–224.

Taş, Duygu, Nico Dellaert, Tom van Woensel, Ton de Kok. 2014a. The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C : Emerging Technologies* **48** 66–83.

Tassopoulos, Ioannis X, Ioannis P Solos, Grigorios N Beligiannis. 2015. A two-phase adaptive variable neighborhood approach for nurse rostering. *Computers & Operations Research* **60** 150–169.

Trautsamwieser, Andrea, Manfred Gronalt, Patrick Hirsch. 2011. Securing home health care in times of natural disasters. *OR Spectrum* **33**(3) 787–813.

Trautsamwieser, Andrea, Patrick Hirsch. 2014. A branch-price-and-cut approach for solving the medium-term home health care planning problem. *Networks* **64**(3) 143–159.

Tsai, Chang-Chun, Sherman HA Li. 2009. A two-stage modeling with genetic algorithms for the nurse scheduling problem. *Expert Systems with Applications* **36**(5) 9506–9512.

Types of Home Health Care Services. n.d. In www.hopkinsmedicine.org. retrieved december 2016, from http ://goo.gl/oa2up1. .

Valouxis, Christos, Christos Gogos, George Goulas, Panayiotis Alefragis, Efthymios Housos. 2012. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research* **219**(2) 425–433.

Van den Bergh, Jorne, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, Liesje De Boeck. 2013. Personnel scheduling : A literature review. *European Journal of Operational Research* **226**(3) 367–385.

Vayanos, Phebe, Daniel Kuhn, Berç Rustem. 2011. Decision rules for information discovery in multi-stage stochastic programming. *50th IEEE Conference on Decision and Control and European Control*. IEEE, 7368–7373.

Vijayakumar, Bharathwaj, Pratik J Parikh, Rosalyn Scott, April Barnes, Jennie Gallimore. 2013. A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research* **224**(3) 583–591.

Wang, Chengshan, Bingqi Jiao, Li Guo, Zhe Tian, Jide Niu, Siwei Li. 2016a. Robust scheduling of building energy system under uncertainty. *Applied Energy* **167** 366–376.

Wang, Shuo, Vahid Roshanaei, Dionne Aleman, David Urbach. 2016b. A discrete event simulation evaluation of distributed operating room scheduling. *IIE Transactions on Healthcare Systems Engineering* **6**(4) 236–245.

Wang, Tao, Nadine Meskens, David Duvivier. 2015. Scheduling operating theatres : Mixed integer programming vs. constraint programming. *European Journal of Operational Research* **247**(2) 401–413.

Wang, Yu, Jiafu Tang, Richard YK Fung. 2014a. A column-generation-based heuristic algorithm for solving operating theater planning problem under stochastic demand and surgery cancellation risk. *International Journal of Production Economics* **158** 28–36.

Wang, Zhaoyu, Bokan Chen, Jianhui Wang, Jinho Kim, Miroslav M. Begovic. 2014b. Robust optimization based optimal dg placement in microgrids. *IEEE Transactions on Smart Grid* **5**(5) 2173–2182.

Wu, Tai-Hsi, Jinn-Yi Yeh, Yueh-Min Lee. 2015. A particle swarm optimization approach with refinement procedure for nurse rostering problem. *Computers & Operations Research* **54** 52–63.

Xiao, Guanlian, Willem van Jaarsveld, Ming Dong, Joris van de Klundert. 2016. Stochastic programming analysis and solutions to schedule overcrowded operating rooms in china. *Computers & Operations Research* **74** 78–91.

Yan, Shangyao, Chih-Kang Lin, Sheng-Yu Chen. 2014. Logistical support scheduling under stochastic travel times given an emergency repair work schedule. *Computers & Industrial Engineering* **67** 20–35.

Yuan, Biao, Ran Liu, Zhibin Jiang. 2015. A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. *International Journal of Production Research* **53**(24) 7450–7464.

Zeng, Bo, Long Zhao. 2013. Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters* **41**(5) 457–461.

Zhang, Bo, Tao Yao, Terry L. Friesz, Yuqi Sun. 2015. A tractable two-stage robust winner determination model for truckload service procurement via combinatorial auctions. *Transportation Research Part B : Methodological* **78** 16–31.

Zhang, Tao, W Art Chaovalitwongse, Yuejie Zhang. 2012. Scatter search for the stochastic travel-time vehicle routing problem with simultaneous pick-ups and deliveries. *Computers & Operations Research* **39**(10) 2277–2290.

Zhang, Yu, Roberto Baldacci, Melvyn Sim, Jiafu Tang. 2016. Routing optimization with time windows under uncertainty. *Optimization Online* .

Zhao, Chaoyue, Yongpei Guan. 2013. Unified stochastic and robust unit commitment. *IEEE Transactions on Power Systems* **28**(3) 3353–3361.

Zhao, Long, Bo Zeng. 2012. Robust unit commitment problem with demand response and wind energy. *Power and Energy Society General Meeting, San Diego, CA*. IEEE, 1–8.

Zheng, T., J. Zhao, E. Litvinov, F. Zhao. 2012. Robust optimization and its application to power system operation. *Proceedings of CIGRE SC C2 Session, Paris, France*. CIGRE.

Zhong, Liwei, Shoucheng Luo, Lidong Wu, Lin Xu, Jinghui Yang, Guochun Tang. 2014. A two-stage approach for surgery scheduling. *Journal of Combinatorial Optimization* **27**(3) 545–556.

Zhou, Bing-hai, Meng Yin, Zhi-qiang Lu. 2016. An improved lagrangian relaxation heuristic for the scheduling problem of operating theatres. *Computers & Industrial Engineering* **101** 490–503.

APPENDIX A SUPPLEMENTS OF ARTICLE 1

A.1 The model for the case of nonidentical operating rooms

In this appendix, we discuss how the CG algorithm proposed in Section 4.3 can take into account the case of nonidentical operating rooms. We consider two cases :

Case 1) Different availabilities : In his case, instead of considering a separate subproblem for each operating room, the presented constraint programming model can be modified by adding dummy mandatory surgeries to represent the unavailable early and late time periods. We add the following constraints to the subproblems :

$$\text{If}\,(W_1 = i)\ \text{then}\,(V_1 = 0), \qquad i \in \mathit{dummy}_{early}, \tag{A.1}$$

$$\text{If}\,(W_p = j)\ \text{then}\,(V_p + t_{[W_p]} = |\mathcal{T}_d|), \qquad j \in \mathit{dummy}_{late}, p \in \{2, ..., n\}. \tag{A.2}$$

We assume that the early dummy surgeries must be scheduled in the first position and the others can be scheduled in any other positions. We specify appropriate deadlines to ensure that the dummies are assigned to the correct days. We must schedule appropriate pairs of dummy surgeries together ; for this we require the following constraint :

$$\text{If}\,(W_1 = i)\ \text{then}\,(C_{comp(i)} == 1)), \qquad i \in \mathit{dummy}_{early}, \tag{A.3}$$

This constraint states that if a dummy surgery is scheduled in the morning, its complementary dummy surgery, denoted *comp*(i), must appear in the same schedule. Constraint (A.2) forces *comp*(i) to be the last surgery in the operating room.

Case 2) Different equipment : For this case we need a different subproblem for each type of operating room. In the master problem, we modify constraint (A.4) as follows :

$$\sum_{j \in J_d} x_j \leq |\mathcal{K}_{r,d}|, \qquad d \in \mathcal{D}, r \in \mathit{Room}_d, \tag{A.4}$$

where Room_d is the set of operating room types on day d, and $|\mathcal{K}_{r,d}|$ is the number of rooms of type r on day d. The model still breaks the symmetry of identical rooms. Note that rooms of the same type have the same equipment but may have different availabilities.

A.2 An integer programming model adapted from (Marques et al. 2012)

The problem defined in Section 4.2 can be formulated as an integer programming model using the four-index binary variables x_{ikdt} (Roland et al. 2006, 2010, Marques et al. 2012) and two other sets of binary variables y_{ikd} and $z_{ii'kd}$. We define the following sets and variables.

Sets :

$\mathcal{T}_{di}$: A subset of $\mathcal{T}_d$ such that if surgery i starts at any time slot in this subset it finishes before the end of regular time on day d.

$Pairs_d$: The set of all pairs of surgeries i and i' with deadlines on or after day d such that cleaning is required if surgery i' is performed immediately after surgery i.

Variables :

x_{ikdt} : 1 if surgery i starts at time t on day d in operating room k ; 0 otherwise.

y_{ikd} : 1 if surgery i is scheduled in operating room k on day d ; 0 otherwise.

$z_{ii'kd}$: 1 if the pair $(i, i') \in Pairs_d$ is scheduled in operating room k on day d and surgery i starts at any time before surgery i' (not necessarily successively).

The integer programming model is as follows :

$$\max \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}_d} \sum_{i \in \mathcal{I}: dd_i \geq d} \sum_{t' \in \mathcal{T}_{di}} (t_i x_{ikdt'}) \tag{A.5}$$

subject to :

$$\sum_{d \in \mathcal{D}: d \leq dd_i} \sum_{k \in \mathcal{K}_d} \sum_{t \in \mathcal{T}_{di}} x_{ikdt} = 1, \qquad i \in \mathcal{I}_1, \tag{A.6}$$

$$\sum_{d \in \mathcal{D}: d \leq dd_i} \sum_{k \in \mathcal{K}_d} \sum_{t \in \mathcal{T}_{di}} x_{ikdt} \leq 1, \qquad i \in \mathcal{I}_2, \tag{A.7}$$

$$\sum_{i \in \mathcal{I}: dd_i \geq d} \sum_{t' \in \mathcal{T}_{di}: max[t - t_i, 0] < t' \leq t} x_{ikdt'} \leq 1, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, t \in \mathcal{T}_d, \tag{A.8}$$

$$\sum_{i \in \mathcal{I}'_l: dd_i \geq d} \sum_{k \in \mathcal{K}_d} \sum_{t' \in \mathcal{T}_{di}: max[t - t_i, 0] < t' \leq t} x_{ikdt'} \leq 1, \qquad d \in \mathcal{D}, l \in \mathcal{L}, t \in \mathcal{T}_d, \tag{A.9}$$

$$\sum_{i \in \mathcal{I}'_l: dd_i \geq d} \sum_{k \in \mathcal{K}_d} \sum_{t' \in \mathcal{T}_{di}} t_i x_{ikdt'} \leq A_{ld}, \qquad l \in \mathcal{L}, d \in \mathcal{D}, \tag{A.10}$$

$$y_{ikd} = \sum_{t \in \mathcal{T}_{di}} x_{ikdt}, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, i \in \mathcal{I} : dd_i \geq d, \tag{A.11}$$

$$z_{ii'kd} \leq y_{ikd}, z_{i'ikd} \leq y_{i'kd}, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, (i,i') \in Pairs_d, \tag{A.12}$$

$$z_{ii'kd} + z_{i'ikd} \leq 1, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, (i,i') \in Pairs_d (i < i'), \tag{A.13}$$

$$z_{ii'kd} + z_{i'ikd} \geq y_{ikd} + y_{i'kd} - 1, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, (i,i') \in Pairs_d (i < i'), \tag{A.14}$$

$$\sum_{t \in \mathcal{T}_{di}} (tx_{ikdt}) + t_i + CL_{ii'} \leq \sum_{t \in \mathcal{T}_{di'}} (tx_{i'kdt}) + M(1 - z_{ii'kd}), \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, (i,i') \in Pairs_d, \tag{A.15}$$

$$x_{ikdt} \in \{0,1\}, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, t \in \mathcal{T}_d, i \in \mathcal{I} : dd_i \geq d, \tag{A.16}$$

$$z_{ii'kd} \in \{0,1\}, \qquad d \in \mathcal{D}, k \in \mathcal{K}_d, (i,i') \in Pairs_d. \tag{A.17}$$

The objective function (A.5) maximizes the total duration of the scheduled surgeries. Constraints (A.6) and (A.7) ensure that the mandatory surgeries are performed and allow the optional surgeries to be postponed. Constraint (A.8) ensures that each operating room on each day is occupied by at most one surgery in each time slot. Constraint (A.9) prevents overlapping surgeries for the same surgeon. Constraint (A.10) enforces the maximum working hours for each surgeon. Constraint (A.11) links the intermediate variables y_{ikd} to x_{ikdt} ; these variables are included to make the model more readable. Constraints (A.12) force $z_{ii'kd}$ to be 0 if surgery i or surgery i' is not scheduled in operating room k on day d. Constraints (A.13) and (A.14) together force either $z_{ii'kd}$ or $z_{i'ikd}$ to be 1 if surgeries i and i' are both scheduled in operating room k on day d. If two surgeries from the set $Pairs_d$ are scheduled in the same room on the same day, they must be sequenced. Constraint (A.15) ensures that if $z_{ii'kd}$ is 1, the start time of surgery i' occurs after the finish time of surgery i in room k on day d plus $CL_{ii'}$.

A.3 A pure constraint-programming model

We now present a constraint programming model for the problem defined in Section 4.2. The constraint programming variables are as follows :

y_{id} : It is an interval variable for surgery i on day d. This variable is present in the solution if surgery i is scheduled on day d.

z_{id} : This variable is the index of the surgery scheduled immediately before surgery i in the same operating room on day d. If surgery i is the first surgery in the room this variable is 0 and if surgery i is not scheduled on day d it is -1.

The following model is implemented using the CP optimizer in IBM ILOG CPLEX Optimization Studio V12.4.

$$\max\{\sum_{d\in\mathcal{D}}\sum_{i\in\mathcal{I}:dd_i\geq d} Length(y_{id})\} \tag{A.18}$$

subject to :

$$z_{id} \in \{i' \in \mathcal{I} | i' \neq i, dd_{i'} \geq d\} \cup \{0, -1\}, \qquad d \in \mathcal{D}, i \in \mathcal{I} : dd_i \geq d, \tag{A.19}$$

$$End(y_{id}) \leq |\mathcal{T}_d|, \qquad d \in \mathcal{D}, i \in \mathcal{I} : dd_i \geq d, \tag{A.20}$$

$$\text{Count}([z_{id}]_{i\in\mathcal{I}:dd_i\geq d}, 0) \leq |\mathcal{K}_d|, \qquad d \in \mathcal{D}, \tag{A.21}$$

$$\text{Count}([z_{id}]_{i\in\mathcal{I}\ \{i'\}:dd_i\geq d}, i') \leq Presence(y_{i'd}), \qquad d \in \mathcal{D}, i' \in \mathcal{I} : dd_{i'} \geq d, \tag{A.22}$$

$$\text{NoOverlap}[y_{id}]_{i\in\mathcal{I}'_l:dd_i\geq d}, \qquad d \in \mathcal{D}, l \in \mathcal{L}, \tag{A.23}$$

$$\begin{aligned}&\text{If } (z_{id} = i') \text{ then } [(Presence(y_{i'd}) = 1)\ \& & d \in \mathcal{D}, i \in \mathcal{I} : dd_i \geq d,\\ &(Start(y_{id}) \geq End(y_{i'd}) + CL_{i'i})], & i' \in \mathcal{I}\backslash\{i\} : dd_{i'} \geq d,\end{aligned} \tag{A.24}$$

$$\text{If } (z_{id} = -1) \text{ then } (Presence(y_{id}) = 0), \qquad d \in \mathcal{D}, i \in \mathcal{I} : dd_i \geq d, \tag{A.25}$$

$$\sum_{d\in\mathcal{D}:dd_i\geq d} Presence(y_{id}) = 1, \qquad i \in \mathcal{I}_1 \tag{A.26}$$

$$\sum_{d\in\mathcal{D}:dd_i\geq d} Presence(y_{id}) \leq 1, \qquad i \in \mathcal{I}_2 \tag{A.27}$$

$$\sum_{i\in\mathcal{I}'_l:dd_i\geq d} Length(y_{id}) \leq A_{ld}, \qquad d \in \mathcal{D}, l \in \mathcal{L} \tag{A.28}$$

The objective function (A.18) maximizes the total scheduled surgery time. Here $Length(y_{id})$ is equal to the duration of surgery i if variable y_{id} is present in the solution. (A.19) specifies the domains of the z_{id} variables. Constraint (A.20) ensures that if surgery i is scheduled on day d it is completed before the end of the day. In constraint (A.21), $\text{Count}([z_{id}]_{i\in\mathcal{I}:dd_i\geq d}, 0)$ counts the number of times that the value 0 appears in the vector $[z_{id}]_{i\in\mathcal{I}:dd_i\geq d}$. This constraint ensures that the number of surgeries that are the first surgeries in their operating rooms on day d does not exceed the number of available operating rooms on this day. The left-hand side of constraint (A.22) counts the number of times that surgery i' is the immediate predecessor of the other surgeries on day d. Constraint (A.22) indicates that if surgery i' is not scheduled on day d, it cannot be the immediate predecessor of any surgery on that day. Constraint (A.23) prevents overlapping surgeries for the same surgeon. Constraint (A.24) states that if surgery i' is the immediate predecessor of surgery i on day d then surgery i' must be scheduled on the same day and the start time of surgery i must be after the finish time of surgery i' plus $CL_{i'i}$. Constraint (A.25) ensures that surgery i is not scheduled on day d if z_{id} is -1. Constraint (A.26) ensures that the mandatory surgeries are performed, and constraint (A.27) allows the optional surgeries to be postponed. Constraint (A.28) enforces the maximum working hours for each surgeon.

We apply dominance rule 1 by adding the following constraint :

$$Start(y_{id}) = 0 \textit{ or } Start(y_{id}) \geq SD_d, \qquad\qquad d \in \mathcal{D}, i \in \mathcal{I} : dd_i \geq d$$

where as defined in Dominance rule 1, SD_d is the smallest surgery duration among all surgeries with deadlines on or after day d. (i.e., $SD_d = \min_{i \in \mathcal{I}; dd_i \geq d} t_i$) We order the y_{id} variables by surgery deadline. For a fixed $i \in \mathcal{I}$, they are ordered lexicographically by d. Each z_{id} variable is evaluated immediately after its corresponding y_{id}.

A.4 Proof of λ values for different cases

In this section, We validate the λ values in Table 4.1. Let Δ_1 be the cleaning time required between the surgeries in positions $p^* - 1$ and p^* if surgery i in position p^* is replaced by surgery i'. Similarly, let Δ_2 be the cleaning time required between the surgeries in positions p^* and $p^* + 1$ if surgery i in position p^* is replaced by surgery i'. We show that regardless of the infection status of the surgery in position $p^* + 1$, $\Delta_1 + \Delta_2 \leq \lambda$ holds for the λ values in Table 4.1. Let f'_{p^*+1} be the infection status of the surgery in position $p^* + 1$ if it is infectious. In tables presented in this appendix, different infection statuses for the surgery in position $p^* + 1$ are presented under Column "$p^* + 1$".

Case 1 : Since surgeries i and i' are both noninfectious, replacing surgery i by surgery i' does not change the possible pre/post-cleaning time : $\lambda = 0$.

Case 2 : Two subcases are possible.

Case 2-1 : The surgery in position $p^* - 1$ is noninfectious. Table A.1 computes Δ_1 and Δ_2 given the infection status of the surgery in position $p^* + 1$. We have $\Delta_1 + \Delta_2 \leq OCT$ and $\lambda = OCT$.

Table A.1 – Validity of λ for case 2-1.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	0	OCT	OCT
Infectious with $f'_{p^*+1} = f_{i'}$	0	0	0
Infectious with $f'_{p^*+1} \neq f_{i'}$	0	OCT	OCT
$\lambda = \max(\Delta_1 + \Delta_2)$			OCT

Case 2-2 : The surgery in position $p^* - 1$ is infectious with $f'_{p^*+1} \neq f_{i'}$. Table A.2 shows that $\lambda = OCT$.

Table A.2 – Validity of λ for case 2-2.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	0	OCT	OCT
Infectious with $f'_{p^*+1} = f_i$	0	0	0
Infectious with $f'_{p^*+1} \neq f_i$ and $f'_{p^*+1} \neq f_{i'}$	0	OCT	OCT
$\lambda = \max(\Delta_1 + \Delta_2)$			OCT

Case 3 : Table A.3 shows that $\lambda = 0$.

Table A.3 – Validity of λ for case 3.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	-OCT	OCT	0
Infectious with $f'_{p^*+1} = f_{i'}$	-OCT	0	-OCT
Infectious with $f'_{p^*+1} \neq f_{i'}$	-OCT	OCT	0
$\lambda = \max(\Delta_1 + \Delta_2)$			0

Case 4 : Table A.4 shows that $\lambda = 0$.

Table A.4 – Validity of λ for case 4.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	0	-OCT	-OCT
Infectious with $f'_{p^*+1} = f_i$	0	0	0
Infectious with $f'_{p^*+1} \neq f_i$	0	-OCT	-OCT
$\lambda = \max(\Delta_1 + \Delta_2)$			0

Case 5 : Table A.5 shows that $\lambda = OCT$.

Case 6 : Since surgeries i and i' have the same type of infection, replacing surgery i by surgery i' does not change the possible pre/post-cleaning time : $\lambda = 0$.

Case 7 : Table A.6 shows that $\lambda = OCT$.

Table A.5 – Validity of λ for case 5.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	OCT	-OCT	0
Infectious with $f'_{p^*+1} = f_i$	OCT	0	OCT
Infectious with $f'_{p^*+1} \neq f_i$	OCT	-OCT	0
$\lambda = \max(\Delta_1 + \Delta_2)$			OCT

Table A.6 – Validity of λ for case 7.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	0	0	0
Infectious with $f'_{p^*+1} = f_i$	0	OCT	OCT
Infectious with $f'_{p^*+1} = f_{i'}$	0	-OCT	-OCT
Infectious with $f'_{p^*+1} \neq f_i$ and $f'_{p^*+1} \neq f_{i'}$	0	0	0
$\lambda = \max(\Delta_1 + \Delta_2)$			OCT

Case 8 : Table A.7 shows that $\lambda = 2OCT$.

Table A.7 – Validity of λ for case 8.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	OCT	0	OCT
Infectious with $f'_{p^*+1} = f_i$	OCT	OCT	2OCT
Infectious with $f'_{p^*+1} = f_{i'}$	OCT	-OCT	0
Infectious with $f'_{p^*+1} \neq f_i$ and $f'_{p^*+1} \neq f_{i'}$	OCT	0	OCT
$\lambda = \max(\Delta_1 + \Delta_2)$			2OCT

Case 9 : Table A.8 shows that $\lambda = 0$.

Table A.8 – Validity of λ for case 9.

$p^* + 1$	Δ_1	Δ_2	$\Delta_1 + \Delta_2$
Noninfectious	-OCT	0	-OCT
Infectious with $f'_{p^*+1} = f_i$	-OCT	OCT	0
Infectious with $f'_{p^*+1} = f_{i'}$	-OCT	-OCT	-2OCT
Infectious with $f'_{p^*+1} \neq f_i$ and $f'_{p^*+1} \neq f_{i'}$	-OCT	0	-OCT
$\lambda = \max(\Delta_1 + \Delta_2)$			0

A.5 Comparison of branches (1) and (2) with branch (3)

Table A.9 gives results for the three branching rules presented in Section 4.5.2. We compare two branch-and-price-and-cut algorithms : in Algorithm A.1 we use branches 1 and 2, and in Algorithm A.2 we use branch 3. The nodes are pruned whenever no branching is possible. Algorithm A.1 may terminate early because no branch can be found. For a fair comparison, we first run Algorithm A.1 and we then run Algorithm A.2 for the same period of time ; this value is given in the Time column. The rightmost columns give the differences in the upper and lower bounds found by the two algorithms. These values indicate that Algorithm A.1 generally provides better upper bounds. Algorithm A.2 generally provides better lower bounds, mainly because it generates more columns.

Table A.9 – Evaluation of the branching procedures.

			Algorithm A.1				Algorithm A.2					
Instances sets	No. of surgeries	Time	No. of columns	UB	LB	Gap (%)	No. of columns	UB	LB	Gap (%)	$UB_2 - UB_1$	$LB_2 - LB_1$
A	40	41	485	1403	1402	0.07	855	1411	1403	0.57	8	1
	60	181	1214	1971	1910	3.09	1973	1975	1958	0.86	4	48
	80	1341	2384	2203	2154	2.01	3391	2211	2170	1.61	8	16
	100	6030	3871	2297	2247	2.14	5491	2297	2251	1.99	0	4
	120	3231	3343	2304	2252	2.26	4123	2306	2254	2.24	2	2
B	40	147	626	1362	1361	0.06	838	1367	1359	0.61	5	-2
	60	1054	1629	1854	1834	1.09	1989	1859	1839	1.09	5	5
	80	2683	2497	2162	2119	2.01	3270	2173	2124	2.27	11	5
	100	9049	3799	2403	2327	3.12	5477	2403	2330	3.01	0	3
	120	8902	4407	2457	2342	4.67	5562	2457	2343	4.61	0	1
C	60	7379	3468	576	561	2.50	3472	576	571	0.66	0	10
	80	7429	1889	792	702	10.68	4216	792	713	9.34	0	11
	100	6823	3593	978	853	12.24	4271	978	860	11.51	0	7
Average :						3.53				3.11	3.31	8.54

A.6 A heuristic for setting the number of operating rooms

We now present the heuristic that determines the number of operating rooms for instance sets B and C.

Step 1 : For each surgeon, we divide the sum of all the surgery durations corresponding to that surgeon over the five days of the week proportional to the surgeon's maximum daily hours (being careful not to exceed the maximum value). This gives an indication of the surgeon's daily working time. This might give the result presented in Table A.10 for an example with eight surgeons.

Table A.10 – The output of Step 1 in the heuristic algorithm proposed for setting the number of operating rooms.

Day	Average daily workloads							
	Surgeon 1	Surgeon 2	Surgeon 3	Surgeon 4	Surgeon 5	Surgeon 6	Surgeon 7	Surgeon 8
Mon.	50.8	90.9	40.7	73.4	29.6	92.1	120.0	80.5
Tue.	93.8	68.4	72.8	60.5	48.4	69.1	36.6	20.3
Wed.	49.3	49.2	18.9	58.8	83.0	37.5	24.1	52.5
Thu.	58.5	24.4	112.8	59.2	63.2	30.0	72.5	69.4
Fri.	23.1	34.3	25.7	22.5	37.8	34.4	43.4	70.6

Step 2 : For each day, we sum the expected daily hours to determine the total operating time. For the example, this gives the result presented in Table A.11.

Table A.11 – The output of Step 2 in the heuristic algorithm proposed for setting the number of operating rooms.

Day	Mon.	Tue.	Wed.	Thu.	Fri.
Total operating time	578.0	469.8	373.4	490.1	291.8

Step 3 : We divide the total times by eight hours (i.e., 96 five-minute intervals) to determine the expected number of operating rooms for each day.

Table A.12 – The output of Step 3 in the heuristic algorithm proposed for setting the number of operating rooms.

Day	Mon.	Tue.	Wed.	Thu.	Fri.
Number of operating rooms	6	5	4	5	3

APPENDIX B SUPPLEMENTS OF ARTICLE 2

B.1 Proof of Lemma 5.1

We prove this lemma by induction. After the initialization in Line 1 of Algorithm 5.1, we have $d_i = 0$ and $d_j = \infty$ $j \in \mathcal{I}\backslash\{i\}$ that is the base of induction. Consider a moment before updating d_k in Line 5 of the algorithm. By the inductive assumption, we know that if $d_j < \infty$ $(j \neq i)$ it is equal to the reach time of customer j through a path originating from customer i at time t, plus the service time of customer j. Therefore, if d_k is updated by $d_k = max\{d_j + t_{rjk\omega}, e_k - t\} + s_{rk\omega}$ in Line 5 it is equal to a reach time of customer k through a path from the customer j because $d_j + t_{rjk\omega}$ presents a reach time of customer k by following a path that originates from customer i. In this path, the vehicle reaches and serves customer j, then travels from customer j to customer k. In $d_k = max\{d_j + t_{rjk\omega}, e_k - t\} + s_{rk\omega}$ by choosing the maximum of $e_k - t$ and $d_j + t_{rjk\omega}$, we ensure that if it is required the vehicle waits until the time window of customer k opens. Also $s_{rk\omega}$ at the end of the recent formula ensures that we consider the service time of customer k in the computation of d_k. Therefore, we proved that Lemma 5.1 holds.

B.2 Proof of Lemma 5.2

We prove this lemma by induction. The base case of the induction at iteration 0 (i.e. just before executing the *for loop* starting in Line 2 for the first t ime) h olds s ince $d\ _i = 0$ and $d_j = \infty$ for $j \in \mathcal{I}\backslash\{i\}$. As the inductive case, we must show that at the end of iteration n, d_k is equal to the shortest reach time for customer $k \in \mathcal{I}$ with at most n arcs. We prove that the inductive case holds by noting the following points.

1. At lines 3 and 4 of Algorithm 5.1, the algorithm tries to improve the reach time d_k by setting $d_k = max\{d_j + t_{rjk\omega}, e_k - t\} + s_{rk\omega}$ if $d_k > max\{d_j + t_{rjk\omega}, e_k - t\} + s_{rk\omega}$ is satisfied for any arc connecting customer j to customer k.
2. By the inductive assumption we know that at the end of iteration $n - 1$, d_j is the shortest reach time for node $i \in \mathcal{I}$ with at most $n - 1$ arcs.

B.3 Proof of Theorem 5.1

After $|\mathcal{I}|-1$ repetitions of the *for loop* starting in Line 2 of Algorithm 5.1, based on Lemma 5.2 we know that d_k is equal to the shortest required time to reach and serve customer k through a path with at most $|\mathcal{I}|-1$ arcs that originates from customer i at time t. Moreover, we know that the paths resulting in the shortest reach times are simple paths with no repetitive nodes. Therefore, since every simple path has at most $|\mathcal{I}|-1$ arcs, we have proved that after $|\mathcal{I}|-1$ repetitions of the *for loop*, d_j is equal to the sum of the shortest reach times $SRT_{rijt\omega}$ and the service time $s_{rj\omega}$ for $j \in \mathcal{I}$. We note that the service time $s_{rj\omega}$ is a part of d_j because in Line 5 of the algorithm, we consider the service time $s_{rk\omega}$. Therefore, by deducting service times from d_j in Line 9, we obtain the correct values of $SRT_{rijt\omega}$.

B.4 Proof of Theorem 5.3

We obtain the following constraint by summing over $i \in (\mathcal{I}_r \cup \{0\}) : (i,j) \in \mathcal{A}_r$ on constraint (5.11).

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{rij\omega}} u_{rij\omega} = \sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} x_{rij} \qquad r\in\mathcal{R}, j\in\mathcal{I}_r \tag{B.1}$$

Constraint (5.3) shows that the right-hand side of constraint (B.1) is equal to 1. Therefore, we can rewrite constraint (B.1) as the following constraints (B.2) and (B.3).

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{rij\omega}} u_{rij\omega} = 1 \qquad r\in\mathcal{R}, j\in\mathcal{I}_r \tag{B.2}$$

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\t'+s_{ri\omega}+t_{rij\omega}>t}} u_{rijt'\omega} + \sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\t'+s_{ri\omega}+t_{rij\omega}\le t}} u_{rijt'\omega} = 1 \qquad r\in\mathcal{R}, j\in\mathcal{I}_r \tag{B.3}$$

By replacing the left-hand side of constraint (5.26) using constraint (B.3), we obtain constraint (5.28).

Also, to obtain constraint (5.29) from constraint (5.27), we first rewrite constraint (B.2) as the following constraints (B.4) and (B.5).

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\t'+s_{ri\omega}+t_{rij\omega}=t}} u_{rijt'\omega} + \sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \sum_{\substack{t'\in\mathcal{T}_{rij\omega}:\\t'+s_{ri\omega}+t_{rij\omega}\neq t}} u_{rijt'\omega} = 1 \qquad r\in\mathcal{R}, j\in\mathcal{I}_r \tag{B.4}$$

$$\sum_{\substack{i\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r\,\&\,(t-s_{ri\omega}-t_{rij\omega})\in\mathcal{T}_{rij\omega}}} u_{rij(t-s_{ri\omega}-t_{rij\omega})\omega} +$$

$$\sum_{\substack{i \in (\mathcal{I}_r \cup \{0\}): \\ (i,j) \in \mathcal{A}_r}} \sum_{\substack{t' \in \mathcal{T}_{rij\omega}: \\ t' + s_{ri\omega} + t_{rij\omega} \neq t}} u_{rijt'\omega} = 1 \qquad r \in \mathcal{R}, j \in \mathcal{I}_r \qquad \text{(B.5)}$$

Then, we obtain constraint (5.29) by replacing the left-hand side of constraint (5.27) using constraint (B.5).

B.5 Proof of Theorem 5.4

To prove this theorem, we show that for any fixed first-stage solution without any subtour, if the corresponding second-stage solution is fractional we can turn it into an integer solution with an objective value that is at least as good as that of the fractional solution. In the following, Algorithm B1 shows how we can modify a fractional second-stage solution in scenario ω to obtain a second-stage integer solution. In this algorithm, a triplet (r, i, j) stands for an arc traversed by a type r vehicle from customer i to customer j. In this algorithm, L_1 maintains the list of all triplets (r, i, j) with $x_{rij} = 1$ for which there is a single index t such that $u_{rijt\omega} = 1$. Also, L_2 maintains the list of triplets (r, i, j) with $x_{rij} = 1$ for which the following two conditions hold :

1. triplet (r, i, j) is not in L_1.
2. $i = 0$ holds or for each $r' \in \mathcal{R}_i$, there is one index $k \in \mathcal{I}_{r'} \cup \{0\}$ satisfying $(r', k, i) \in L_1$.

The goal of Algorithm B1 is to modify a second-stage fractional solution for scenario ω to an integer solution with an objective value that is at least as good as the objective value of the original fractional solution. To prove that Algorithm B1 fulfills this goal, we must show that the following statements hold :

Statement 1 : At the end of Algorithm B1, all triplets (r, i, j) with $x_{rij} = 1$ are in list L_1.

Statement 2 : When modifying the solution in Step 1, all constraints remain satisfied.

Statement 3 : When modifying the solution in Step 1, the objective function does not deteriorate.

Part 1- Proof of Statement 1

We prove the validity of *Statement 1* by contradiction. Let's assume that this statement does not hold. It means that in one iteration of the algorithm, when we are in Step 1, L_2 is empty and there is one or more triplets (r, i, j) with $x_{rij} = 1$ that are neither in L_1 nor in L_2. We consider the following two cases :

Case 1) There is only one triplet (r, i, j) with $x_{rij} = 1$ that is neither in L_1 nor in L_2. With respect to the definition of L_2, since triplet $(r, i, j) \notin L_2$, we must have $i \neq 0$ and also there

Algorithm B1. Modification of second-stage fractional solutions

```
1: Set $L_1 = \{(r,i,j) \in \mathcal{R} \times \mathcal{I}^2 | \exists t \in \mathcal{T}_{rij\omega} : u_{ijt\omega} = 1\}$ and $L_2 = \{(r,i,j) \in (\mathcal{R} \times \mathcal{I}^2) \backslash L_1 | (i = 0)$ or $\forall r' \in \mathcal{R}_i \; \exists k \in (\mathcal{I}_{r'} \cup \{0\}) : (r',k,i) \in L_1\}$.
2: **Step 1**- Choose a triplet $(r,i,j) \in L_2$ and modify $u_{rijt\omega}$ for $t \in \mathcal{T}_{rij\omega}$ by setting $u_{rij(t_i^*)\omega} = v_{i(t_i^*)\omega} = 1$ and $u_{ijt\omega} = v_{it\omega} = 0$ for $t \neq t_i^*$ where $t_i^* = argmin\{t \in \mathcal{T}_{i\omega} | u_{rijt\omega} > 0\}$.
3: **Step 2**- Remove $(r,i,j)$ from $L_2$ and add it to $L_1$. If $L_1$ includes all triplets $(r,i,j)$ for which $x_{rij} = 1$ then the algorithm is converged and an integer solution with an objective value that is at least as good as the objective value of the original fractional solution is obtained.
```

must be at least one vehicle type $r_i' \in \mathcal{R}_i$, for which $(r_i', k, i) \notin L_1$ for all $k \in \mathcal{I}_{r_i'} \cup \{0\}$. Also, with respect to constraint (5.3), we know that there is an index $k_i \in \mathcal{I}_{r_i'} \cup \{0\}$ such that $x_{r_i' k_i i} = 1$ holds. Here two cases are possible :

Case 1-1) triplet (r_i', k_i, i) is in L_2 that is a contradiction because we assumed that L_2 is empty.

Case 1-2) triplet (r_i', k_i, i) is not L_2. As explained in Case 1, this triplet cannot be a member of L_1 too. Therefore, triplet (r,i,j) is not the only triplet with $x_{rij} = 1$ that is neither in L_1 nor in L_2 that is a contradiction with the assumption of Case 1.

Case 2) There are two or more triplets (r,i,j) with $x_{rij} = 1$ that are neither in L_1 nor in L_2. With respect to the definition of L_2, since triplets (r,i,j) are not in L_2, for each triplet (r,i,j) we must have $i \neq 0$ and also there must be at least one vehicle type $r_i' \in \mathcal{R}_i$, for which $(r_i', k, i) \notin L_1$ for all $k \in \mathcal{I}_{r_i'} \cup \{0\}$. Also, with respect to constraint (5.3), we know that for each triplet (r,i,j) there is an index $k_i \in \mathcal{I}_{r_i'} \cup \{0\}$ such that $x_{r_i' k_i i} = 1$. This means that for any triplet (r,i,j) that is not in L_1, there is at least one ingoing triplet (r_i', k_i, i) that is not in L_1 either. Therefore, this shows that there is at least one tour formed by triplets (r,i,j) that are not in L_1. We claim that this tour is a subtour. The reason is that indices k_i in ingoing triplets (r_i', k_i, i) cannot be equal to 0. This is because if $k_i = 0$ holds, then with respect to the definition of L_2, triplet (r_i', k_i, i) would have been either in L_1 or in L_2, while both of these cases are contradictions. Therefore, we conclude that there is at least one subtour formed by triplets (r,i,j) with $x_{rij} = 1$ that are not in L_1. The existence of such a subtour is in contradiction with the fact that the first-stage solution is subtour-free.

Part 2- Proof of Statement 2

To prove that *Statement 2* holds, we should show that constraints (5.11)-(5.15) are satisfied by the modified solution. Regarding constraint (5.14), we suppose that this constraint is already removed through replacing $w_{r\omega}$ in objective function (5.10). Constraint (5.11) remains satisfied because exactly one of the variables $u_{rijt\omega}$ is set to 1 if $x_{rij} = 1$. To demonstrate that constraint (5.12) remains satisfied we consider two following cases :

Case 1) Variables $u_{rijt\omega}$ $t \in \mathcal{T}_{i\omega}$ modified in Step 1 of Algorithm B1 are on the left-hand side of constraint (5.12). In this case, it is clear that by setting $u_{rij(t_i^*)\omega} = 1$ in Step 1 of Algorithm B1 this constraint remains satisfied.

Case 2) In this case, we assume modified variables $u_{rijt\omega}$ $t \in \mathcal{T}_{i\omega}$ are on the right-hand side of constraint (5.12). For the sake of consistency with indices of variables on the right-hand side of constraint (5.12), we assume that variables $u_{rjkt\omega}$ $t \in \mathcal{T}_{j\omega}$ with $x_{rjk} = 1$ are modified in Step 1 of Algorithm B1. First, note that in Algorithm B1, with respect to the precedence relations induced by non-zero x variables, we are sure that before modifying variables $u_{rjkt\omega}$ $t \in \mathcal{T}_{j\omega}$ that are on the right-hand side of constraint (5.12), variables $u_{rijt\omega}$ $t \in \mathcal{T}_{i\omega}$ with $x_{rij} = 1$ that are on the left-hand side of constraint (5.12) are already modified by the algorithm. Therefore, to demonstrate that constraint (5.12) remains satisfied in this case, we only need to show that $t_i^* + s_{ri\omega} + t_{rij\omega} \leq t_j^*$. To show the validity of this inequality we note that t_i^* and t_j^* denote the smallest indices $t \in \mathcal{T}_{i\omega}$ and $t \in \mathcal{T}_{j\omega}$ for which $u_{rijt\omega} > 0$ and $u_{rjkt\omega} > 0$ before modifying the variables. Therefore, if $t_i^* + s_{ri\omega} + t_{rij\omega} \leq t_j^*$ does not hold it means that constraint (5.12) was not satisfied before applying Algorithm B1 that is a contradiction.

Regarding constraint (5.13), it is clear that, by modifying the solution in Step 1 of Algorithm B1, this constraint remains satisfied. Also for constraint (5.15), we note that all $u_{r0it\omega}$ variables remain unchanged as their values before applying Algorithm B1 are either 0 or 1. So, this constraint remains satisfied by the modified second-stage solution.

Part 3- Proof of Statement 3

After replacing variables $w_{r\omega}$ using constraint (5.14), we rewrite objective function (5.10) as follows :

$$Q(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega)) = \min_{u,v} \sum_{r \in \mathcal{R}} \sum_{\substack{i \in \mathcal{I}_r: \\ (i,0) \in \mathcal{A}_r}} \sum_{t \in \mathcal{T}_{ri0t\omega}} (c^o_{rit\omega} + c^w_r f_{rit\omega}) u_{ri0t\omega} + \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}_{i\omega}} c^d_{it} v_{it\omega}$$
$$- \sum_{r \in \mathcal{R}} c^w_r \left[\sum_{\substack{i,j \in (\mathcal{I}_r \cup \{0\}): \\ (i,j) \in \mathcal{A}_r}} g_{rijt\omega} x_{rij} + \sum_{i \in \mathcal{I}_r} \sum_{t \in \mathcal{T}^{dep}_{ri}} t y_{rit} \right] \tag{B.6}$$

Apparently, for fixed indices $r \in \mathcal{R}$ and $i \in \mathcal{I}_r$, cost coefficients of $u_{ri0t\omega}$ variables are higher for larger values of $t \in \mathcal{T}_{ri0\omega}$. Similarly, for a fixed index $i \in \mathcal{I}$, cost coefficients of $v_{it\omega}$ variables are higher for larger values of $\mathcal{T}_{i\omega}$. Therefore, it is clear that by modifying the second-stage solution as proposed in Step 1 of Algorithm B1, the objective value does not deteriorate.

B.6 Proof of Lemma 5.3

As shown in objective function (B.6), the part of the second-stage objective function related to second-stage variables is equal to the weighted sum of tours completion times and service start times. Also, as discussed in the last paragraph in the proof of Theorem 5.4, coefficients of $u_{ri0t\omega}$ and $v_{it\omega}$ are non-decreasing in terms of t. Therefore, for a fixed first-stage solution, we obtain the best second-stage objective value when customers are served as soon as all required vehicles are available.

B.7 Proof of Lemma 5.4

Let's assume that $\kappa_j(\omega)$ denotes the finish time of service to customer j in scenario ω. In the following, we suppose that customer i is a predecessor of customer j if there is $r \in \mathcal{R}_j$ such that $x_{rij} = 1$. To prove the convexity of $\kappa_j(\omega)$ for all customers j we define two lists L_1 and L_2 as follows.

- L_1 maintains the list of customers j for which the convexity of $\kappa_j(\omega)$ is already proven.
- L_2 maintains the list of customers j for which there is not any customer i as the predecessor (i.e., for all $r \in \mathcal{R}_j, i \in \mathcal{I}_r : (i,j) \in \mathcal{A}_r$ we have $x_{rij} = 0$ or all customers i that are predecessors of customer j are in L_1).

To prove the convexity of $\kappa_j(\omega)$ for all customers j, we sequentially choose a customer j from L_2, prove the convexity of $\kappa_j(\omega)$ and add customer j to list L_1. Since there is not any subtour in routes formed by the first-stage solution x, after $|\mathcal{I}|$ iterations we are sure that the convexity $\kappa_j(\omega)$ for all customers j is proven.

For customer j selected from L_2, we compute $\kappa_j(\omega)$ by $\kappa_j(\omega) = max\{\underset{r \in \mathcal{R}_j}{max}\{\kappa_{i_r}(\omega) + t_{ri_rj\omega}\}, e_j\} + \underset{r \in \mathcal{R}_j}{max}\{s_{rj\omega}\}$ where i_r denotes the customer that a type r vehicle has served immediately before customer j. $s_{rj\omega}$ and $t_{ri_rj\omega}$ are parameters in the vector $\xi(\omega)$ and therefore they are convex in terms of $\xi(\omega)$ because assuming that $s_{rj(\alpha\omega_1+(1-\alpha)\omega_2)}$ and $t_{ri_rj(\alpha\omega_1+(1-\alpha)\omega_2)}$ denote the corresponding service and travel times in scenario $\alpha\xi(\omega_1) + (1-\alpha)\xi(\omega_2)$ we have $s_{rj(\alpha\omega_1+(1-\alpha)\omega_2)} = \alpha s_{rj\omega_1} + (1-\alpha)s_{rj\omega_2}$ and $t_{ri_rj(\alpha\omega_1+(1-\alpha)\omega_2)} = \alpha t_{ri_rj\omega_1} + (1-\alpha)t_{ri_rj\omega_2}$. Also e_j is a constant. Moreover, the convexity of $\kappa_{i_r}(\omega), r \in \mathcal{R}_j$ are proven in previous iterations of the algorithm. We know that convex functions are closed for two operations of summation and maximization, i.e., the sum and the maximum of two convex functions is a convex function too. Therefore, it is clear that $\kappa_j(\omega) = max\{\underset{r\in\mathcal{R}_j}{max}\{\kappa_{i_r}(\omega) + t_{ri_rj\omega}\}, e_j\} + \underset{r\in\mathcal{R}_j}{max}\{s_{rj\omega}\}$ is convex in $\xi(\omega)$. To complete our proof, we should show that in initial steps, when we evaluate the convexity of $\kappa_j(\omega)$ for customer j without any predecessor (i.e., $i_r = 0$), $\kappa_j(\omega)$ is convex. We note that in this case we have $\kappa_{i_r}(\omega) = \sum_{t\in\mathcal{T}} t\hat{y}_{rjt}$ that shows $\kappa_{i_r}(\omega)$ is a constant and the convexity of $\kappa_j(\omega)$ is proven in the same way explained before.

B.8 Proof of Lemma 5.5

In the general case, when service and travel times are not multiples of the time slots length, we define the second-stage cost function $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ as follows.

$$Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega)) = \sum_{i\in\mathcal{I}} c'_{delay} d_{i\omega} + \sum_{i\in\mathcal{I}}\sum_{r\in\mathcal{R}_i} c'_{overtime} o_{ri\omega} + \sum_{r\in\mathcal{R}} c^w_r w_{r\omega} \tag{B.7}$$

In $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ defined by (B.7), $d_{i\omega} = max\{\phi_i(\omega) - l_i, 0\}$ denotes the delay in serving customer i where $\phi_i(\omega)$ stands for the time that service to customer i starts. Also, $o_{ri\omega}$ denotes the overtime of the type r vehicle returning to the depot after serving customer i. We compute it by

$$o_{ri\omega} = \begin{cases} (\kappa_i(\omega) + t_{ri0\omega} - L), & \text{if } x_{ri0} = 1 \\ 0 \qquad\qquad\qquad\qquad , & \text{if } x_{ri0} = 0 \end{cases}$$

where $\kappa_i(\omega)$ is the finish time of service to customer i. Furthermore, $w_{r\omega}$ indicates the total waiting time of all vehicles of type r in scenario ω that is computed by

$$w_{r\omega} = \sum_{i\in\mathcal{I}_r} C_{ri\omega} - \sum_{\substack{i,j\in(\mathcal{I}_r\cup\{0\}):\\ (i,j)\in\mathcal{A}_r}} g_{rijt\omega} x_{rij} - \sum_{i\in\mathcal{I}_r}\sum_{t\in\mathcal{T}^{dep}_{ri}} t y_{rit}$$

where $C_{ri\omega}$ denotes the completion time of the type r vehicle returning to the depot after serving customer i. We compute $C_{ri\omega}$ by the following formula :

$$C_{ri\omega} = \begin{cases} (\kappa_i(\omega) + t_{ri0\omega}), & \text{if } x_{ri0} = 1 \\ 0 \qquad\qquad\qquad , & \text{if } x_{ri0} = 0 \end{cases}$$

In the following, we prove that $d_{i\omega}$, $o_{ri\omega}$ and $w_{r\omega}$ are convex in terms of $\xi(\omega)$, and therefore $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ is a convex function.

First, we note that $d_{i\omega} = max\{\phi_i(\omega) - l_i, 0\}$ where $\phi_i(\omega) = max\{\underset{r\in\mathcal{R}_j}{max}\{\kappa_{i_r}(\omega) + t_{ri_rj\omega}\}, e_j\}$

and $\kappa_i(\omega)$ and i_r are defined as in the proof of Lemma 5.4. Considering that we proved the convexity of $\kappa_i(\omega)$ using Lemmas 5.3 and 5.4 and also the fact that the sum and the maximum of two convex functions is also a convex function, we conclude that $\phi_i(\omega)$ is also convex in terms of $\xi(\omega)$. Similarly, based on the convexity of $\phi_i(\omega)$ we conclude that $d_{i\omega}$ is a convex in terms of $\xi(\omega)$ too. Also, based on the convexity of $\kappa_i(\omega)$ in terms of $\xi(\omega)$, it is clear that for a fixed first-stage solution $(\boldsymbol{x}, \boldsymbol{y})$, $o_{ri\omega}$, $C_{ri\omega}$ and $w_{r\omega}$ are convex. Therefore $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ is convex in terms of $\xi(\omega)$.

B.9 Proof of Lemma 5.6

First, we note that $Q'(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\omega)\rfloor) = Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\omega)\rfloor)$ holds because based on the definition of $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$, this function computes the second-stage cost regardless of whether or not travel and service times in $\xi(\omega)$ are multiples of the time slots length. Therefore, to prove that $Q'(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\omega)\rfloor) \leq Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ is correct we just need to show that $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\omega)\rfloor) \leq Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ holds. Based on Lemma 5.3, we know that service to a customer starts as soon as all required vehicles are available at the customer's location. Therefore, it is clear that $\phi_i(\omega)$ and $\kappa_i(\omega)$ which respectively stand for the start time and finish time of the service to customer i, do not increase when we replace $\xi(\omega)$ by $\lfloor\xi(\omega)\rfloor$. Considering relation (B.7) in the proof of Lemma 5.5, which represents $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$, we notice that $d_{i\omega}$, $o_{ri\omega}$, $C_{ri\omega}$ and $w_{r\omega}$ are non-decreasing functions in terms of $\kappa_i(\omega)$ and $\phi_i(\omega)$, and therefore we conclude that by replacing $\xi(\omega)$ with $\lfloor\xi(\omega)\rfloor$ the second-stage cost does not increase and $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \lfloor\xi(\omega)\rfloor) \leq Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ holds.

B.10 Proof of Theorem 5.5

As a result of Lemma 5.5, we can write Jensen's Inequality (Jensen 1906) for $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$ as follows.

$$Q^{general}(\boldsymbol{x}, \boldsymbol{y}) = \underset{\omega\in\Omega}{E}[Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))] \geq Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \bar{\xi}(\omega)) \tag{B.8}$$

Also, based on the definition of $Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))$, the following relation holds.

$$Q'(\boldsymbol{x}, \boldsymbol{y}) = \underset{\omega\in\Omega}{E}[Q'(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))] = \underset{\omega\in\Omega}{E}[Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \xi(\omega))] \tag{B.9}$$

Relations (B.8) and (B.9) show that the following relation holds.

$$Q'(\boldsymbol{x}, \boldsymbol{y}) \geq Q^{general}(\boldsymbol{x}, \boldsymbol{y}, \bar{\xi}(\omega)) \tag{B.10}$$

Also, Lemma 5.6 and relation (B.10) demonstrate the following inequality.

$$Q'(\boldsymbol{x}, \boldsymbol{y}) \geq Q'\left(\boldsymbol{x}, \boldsymbol{y}, \left\lfloor \bar{\xi}(\omega) \right\rfloor\right) \tag{B.11}$$

In master problem (MP), θ stands for the second-stage cost without negative parts of the waiting cost and we have $Q'(\boldsymbol{x}, \boldsymbol{y})$. Therefore, with respect to relation (B.11), $\theta \geq Q'\left(\boldsymbol{x}, \boldsymbol{y}, \left\lfloor \bar{\xi}(\omega) \right\rfloor\right)$ is a valid inequality for the master problem of the proposed L-shaped algorithm.

B.11 Proof of Theorem 5.6

We use a contradiction approach to prove that the proposed lower bounding functional prevents solutions with subtours. Let's assume that a subtour $(i_1, i_2, ..., i_n, i_1)$ exists for the vector of resources $(r_1, r_2, ..., r_n, r_1)$ that means $x_{r_c i_c i_{c+1}} = 1$ for $c = 1, 2, ..., n$ ($i_{n+1} = i_1$ and $r_{n+1} = r_1$). Assume that the start time of service to customer i_c is t^*_c, i.e., $v_{i(t^*_c)\omega} = 1$. Considering constraint (5.13) for $i = i_{n+1}$, $r = r_n$ and $t = t^*_{n+1}$, we have $\sum\limits_{\substack{j \in (\mathcal{I}_{r_n} \cup \{0\}): \\ (i,j) \in \mathcal{A}_{r_n}}} u_{r_n i_{n+1} j t^*_{n+1} \omega} = 1$ (note that $i \in \mathcal{I}_{r_n}$ because $x_{r_n i_n i_{n+1}} = 1$). Also, the right-hand side of constraint (5.12) for $r = r_n$, $j = i_{n+1}$ and $t = t^*_{n+1}$ is equal to $\sum\limits_{\substack{j \in (\mathcal{I}_{r_n} \cup \{0\}): \\ (i,j) \in \mathcal{A}_{r_n}}} u_{r_n i_{n+1} j t^*_{n+1} \omega} = 1$. Therefore, the left-hand side of this constraint must be equal to 1 (note that it cannot be greater than 1 because of constraint (5.11) and constraint (5.3)). Also since $x_{r_n i_n i_{n+1}} = 1$, with respect to constraint (5.11), we know that index i on the left-hand side of constraint (5.12) must be equal to i_n. Thus, there exists an index t' satisfying $t' + s_{r_n i_n \omega} + t_{r_n i_n i_{n+1} \omega} \leq t^*_{n+1}$ for which $u_{r_n i_n i_{n+1} t' \omega} = 1$. This also shows that, with respect to constraint (5.13), the start time of service to customer i_n, that is denoted by t^*_n, satisfies $t^*_n < t^*_{n+1}$. By repeating the above procedure n times, it is clear that we obtain $t^*_1 < t^*_2 < ... < t^*_n < t^*_{n+1} = t^*_1$ that is a contradiction.

B.12 Proof of Lemma 5.7

To prove this lemma, we show that by replacing the values of dual variables using (5.49)-(5.58), all constraints (5.42)-(5.46) in Model (D$_\omega$) are satisfied. In the following, we check constraints one by one for the proposed solution.

Evaluation of constraint (5.42)

Case 1) Assume that we have $x_{rij} \neq 1$. In this case, with respect to (5.55), constraint (5.42) is satisfied.

Case 2) In this case, we assume that $\hat{x}_{rij} = 1$ and $t \geq t^*_{i\omega} = dep_{rj}$ holds. With respect to (5.58), we have $\pi^{(2)}_{rjt'\omega} = \pi^{(1)}_{rij\omega} = \pi^{(3)}_{rjt'\omega} = 0$. Therefore, constraint (5.42) is satisfied.

Case 3) In this case, we assume that $\hat{x}_{rij} = 1$ and $t < t^*_{i\omega} = dep_{rj}$ holds. With respect to (5.58), we have $\pi^{(1)}_{r0j\omega} = 0$. After substitution of $\pi^{(2)}_{rjt\omega}$ using (5.51), we can rewrite constraint (5.42) as follows :

$$\sum_{\substack{t' \in \mathcal{T}_{j\omega}: t+t_{r0j\omega} \leq t' \\ t' \geq dep_{rj}+t_{r0j\omega}}} \pi^{(3)}_{rjt'\omega} \leq 0 \qquad r \in \mathcal{R}, j \in \mathcal{I}_r : (0,j) \in \mathcal{A}_r, t \in \mathcal{T}_{r0j\omega} \tag{B.12}$$

In the following, we show that summation on the left-hand side of (B.12) is equal to 0. Two cases are possible : either $r \in \mathcal{NR}_{j\omega}$ or $r \in \mathcal{CR}_{j\omega}$ holds. In the first case, with respect (5.58) all $\pi^{(3)}_{rjt'\omega}$ in (B.12) are equal to 0. In the second case, we note that $t' \geq dep_{rj} + t_{r0j\omega}$ implies $t' \geq t^*_{j\omega}$ because of $r \in \mathcal{CR}_{j\omega}$. Therefore, again (5.58) indicates that all $\pi^{(3)}_{rjt'\omega}$ in relation (B.12) are equal to 0.

Evaluation of constraint (5.43)

Case 1) Assume that we have $\hat{x}_{rij} \neq 1$. In this case, with respect to (5.56), constraint (5.43) is satisfied.

In all of the following cases for constraint (5.43), we assume $\hat{x}_{rij} = 1$.

Case 2) In this case, we assume that customer i is a non-critical customer, i.e., $i \in \mathcal{NT}_\omega$. In this case, with respect to relation (5.58) we have $\pi^{(1)}_{rij\omega} = \pi^{(3)}_{rit'\omega} = \pi^{(4)}_{rit\omega} = 0$. So, we can rewrite constraint (5.43) as follows :

$$\sum_{\substack{t' \in \mathcal{T}_{j\omega}: \\ t+s_{ri\omega}+t_{rij\omega} \leq t'}} \pi^{(3)}_{rjt'\omega} \leq 0 \qquad r \in \mathcal{R}, i,j \in \mathcal{I}_r : (i,j) \in \mathcal{A}_r, t \in \mathcal{T}_{rij\omega} \tag{B.13}$$

By considering two cases of $r \in \mathcal{NR}_{j\omega}$ or $r \in \mathcal{CR}_{j\omega}$ we can easily show that the remaining terms on the left-hand side of constraint (5.43) are equal to 0. In the case of $r \in \mathcal{NR}_{j\omega}$, relation (5.58) shows that all remaining $\pi^{(3)}_{rjt'\omega}$ are equal to 0. Also, in the case of $r \in \mathcal{CR}_{j\omega}$, with respect to relation (5.58), we know that $\pi^{(3)}_{rjt'\omega} = 0$ for $t' \geq t^*_{j\omega}$. In the following, we show that in constraint (5.43) for all terms $\pi^{(3)}_{rjt'\omega}$, $t' \geq t^*_{j\omega}$ holds and thus all are equal to 0. In constraint (5.43), we note that the summation of $\pi^{(3)}_{rjt'\omega}$ is over $t' \in \mathcal{T}_{j\omega} : t' \geq t + s_{ri\omega} + t_{rij\omega}$ where $t \in \mathcal{T}_{rij\omega}$. We know that for all $t \in \mathcal{T}_{rij\omega}$ we have $t \geq e_i$. Therefore, in $\sum_{\substack{t' \in \mathcal{T}_{j\omega}: \\ t+s_{ri\omega}+t_{rij\omega} \leq t'}} \pi^{(3)}_{rjt'\omega}$, condition $t' \geq t + s_{ri\omega} + t_{rij\omega}$ shows $t' \geq t + s_{ri\omega} + t_{rij\omega} \geq e_i + s_{ri\omega} + t_{rij\omega} = t^*_{j\omega}$ holds. In the recent relation, $t + s_{ri\omega} + t_{rij\omega} \geq e_i + s_{ri\omega} + t_{rij\omega}$ holds because in Case 2 we have assumed $i \in \mathcal{NT}_\omega$. Also, $e_i + s_{ri\omega} + t_{rij\omega} = t^*_{j\omega}$ holds because we have assumed $r \in \mathcal{CR}_{j\omega}$. Thus, for $r \in \mathcal{CR}_{j\omega}$ we have $\sum\limits_{\substack{t' \in \mathcal{T}_{j\omega}: \\ t + s_{ri\omega} + t_{rij\omega} \leq t'}} \pi^{(3)}_{rjt'\omega} = \sum\limits_{\substack{t' \in \mathcal{T}_{j\omega}: \\ t^*_{rj\omega} \leq t'}} \pi^{(3)}_{rjt'\omega} = 0$

Case 3) In this case, we assume that customer i is a critical customer, i.e., $i \in \mathcal{CT}_\omega$ and also $r \in \mathcal{NR}_{i\omega}$. In constraint (5.43), with respect to (5.58), we have $\pi^{(1)}_{rij\omega} = \pi^{(3)}_{rit'\omega} = 0$. By substituting $\pi^{(4)}_{rit\omega}$ using relation (5.50), the constraint reduces to $0 \leq 0$. In replacing the value of $\pi^{(4)}_{rit\omega}$ using relation (5.50), we note that $k_{ri} = j \neq 0$.

Case 4) In this case, we assume that customer i is a critical customer, i.e., $i \in \mathcal{CT}_\omega$ and also $r \in \mathcal{CR}_{i\omega}$. We divide this case into four sub-cases as follows :

Case 4-1) In this case, we additionally assume that $t < t^*_{i\omega}$ holds. By substituting the values of $\pi^{(3)}_{rit'\omega}$ using (5.53), we can rewrite constraint (5.43) as follows.

$$\pi^{(1)}_{rij\omega} + \pi^{(4)}_{ri(t^*_{i\omega})\omega} \leq 0 \qquad r \in \mathcal{R}, i, j \in \mathcal{I}_r : (i,j) \in \mathcal{A}_r, t \in \mathcal{T}_{rij\omega} \quad \text{(B.14)}$$

We note that, in replacement of $\pi^{(3)}_{rit'\omega}$ using relation (5.53), we have $k_{ri} = j \neq 0$. Then by replacing $\pi^{(1)}_{rij\omega}$ and $\pi^{(4)}_{ri(t^*_{i\omega})\omega}$ using relations (5.52) and (5.49), we obtain the following relation.

$$\alpha_{ri\omega} \sum_{\substack{r' \in \mathcal{NR}_{i\omega} : k_{r'i} \neq 0 \\ t' \in \mathcal{T}_{k_{r'i}\omega} : t^*_{i\omega} + s_{r'i\omega} + t_{r'ik_{r'i}\omega} \leq t'}} \pi^{(3)}_{r'k_{r'i}t'\omega} \leq 0 \qquad r \in \mathcal{R}, i, j \in \mathcal{I}_r : (i,j) \in \mathcal{A}_r, t \in \mathcal{T}_{rij\omega} \quad \text{(B.15)}$$

By considering two cases of $r' \in \mathcal{NR}_{k_{r'i}\omega}$ and $r' \in \mathcal{CR}_{k_{r'i}\omega}$ we can show that all $\pi^{(3)}_{r'k_{r'i}t'\omega}$ on the left-hand side of (B.15) are equal to 0 with respect to relation (5.58). For the case of $r' \in \mathcal{CR}_{k_{r'i}\omega}$, we note that $t' \geq t^*_{i\omega} + s_{r'i\omega} + t_{r'ik_{r'i}\omega}$ implies $t' \geq t^*_{k_{r'i}}$ and therefore relation (5.58) is applicable.

Case 4-2) In this case, we assume that $t \geq t^*_{i\omega}$ holds. We note that, with respect to (5.58) all $\pi^{(3)}_{rit'\omega}$ in constraint (5.43) are equal to 0. By substituting $\pi^{(4)}_{rit\omega}$ and $\pi^{(1)}_{rij\omega}$ using relations (5.49) and (5.52), we obtain the following relation.

$$\sum_{\substack{t' \in \mathcal{T}_{j\omega}: \\ t + s_{ri\omega} + t_{rij\omega} \leq t'}} \pi^{(3)}_{rjt'\omega} + \alpha_{ri\omega} \sum_{\substack{r' \in \mathcal{NR}_{i\omega} : k_{r'i} \neq 0 \\ t' \in \mathcal{T}_{k_{r'i}\omega} : t + s_{r'i\omega} + t_{r'ik_{r'i}\omega} \leq t'}} \pi^{(3)}_{r'k_{r'i}t'\omega} \leq 0 \qquad r \in \mathcal{R}, i, j \in \mathcal{I}_r : (i,j) \in \mathcal{A}_r, t \in \mathcal{T}_{rij\omega} \quad \text{(B.16)}$$

Similar to previous Case 4-1, by considering two case of $r' \in \mathcal{NR}_{k_{r'i}\omega}$ and $r' \in \mathcal{CR}_{k_{r'i}\omega}$, we can show that second summation on the left-hand side of (B.16) is equal to 0. We note that, in the case of $r' \in \mathcal{CR}_{k_{r'i}\omega}$, we use the assumption $t \geq t^*_{i\omega}$ to show that $t' \geq t + s_{r'i\omega} + t_{r'ik_{r'i}\omega} \geq t^*_{i\omega} + s_{r'i\omega} + t_{r'ik_{r'i}\omega} = t^*_{k_{r'i}\omega}$. Regarding the first summation on the left-hand side of (B.16), we can similarly show that all $\pi^{(3)}_{rjt'\omega}$ are equal to 0 for both cases $r \in \mathcal{NR}_{j\omega}$ and $r \in \mathcal{CR}_{j\omega}$. Therefore, (B.16) reduces to $0 \leq 0$ that holds.

Evaluation of constraint (5.44)

Case 1) Assume that we have $\hat{x}_{ri0} \neq 1$. In this case, with respect to (5.57), constraint (5.44) is satisfied.

Case 2) We assume that $\hat{x}_{ri0} = 1$ holds and also customer i is a non-critical customer, i.e., $i \in \mathcal{NT}_\omega$. In this case, in constraint (5.44), we have $\pi^{(4)}_{rit\omega} = \pi^{(3)}_{rit'\omega} = 0$ with respect to constraint (5.58). Also by substituting $\pi^{(1)}_{rij\omega}$, using relations (5.54), constraint (5.44) reduces to $\lambda_{rit^*_{i\omega}\omega} \leq \lambda_{rit\omega}$ which holds because for $t \in \mathcal{T}_{ri0\omega}$ we have $t \geq e_i = t^*_{i\omega}$. The recent equality is valid because we assumed $i \in \mathcal{NT}_\omega$. Relation $t \geq t^*_{i\omega}$ shows that $\lambda_{rit^*_{i\omega}\omega} \leq \lambda_{rit\omega}$ holds because $\lambda_{rit\omega}$ is non-decreasing in t.

Case 3) We assume that $\hat{x}_{ri0} = 1$ holds and also customer i is a critical customer, i.e., $i \in \mathcal{CT}_\omega$. We break case 3) to three following sub-cases :

Case 3-1) In this case, we assume that $r \in \mathcal{CR}_{i\omega}$ and $t \in \mathcal{T}_{ri0\omega} : t \geq t^*_{i\omega}$ hold. In this case, With respect to (5.58), all terms $\pi^{(3)}_{rit'\omega}$ in constraint (5.44) are equal to 0. By substituting relations $\pi^{(1)}_{rij\omega}$ and $\pi^{(4)}_{rit\omega}$ using relations (5.52) and (5.49), constraint (5.44) reduces to the following relation.

$$\lambda_{rit^*_{i\omega}\omega} + \alpha_{rj\omega} \sum_{\substack{r' \in \mathcal{NR}_{i\omega} : k_{r'i} \neq 0 \\ t' \in \mathcal{T}_{k_{r'i}\omega} : t + s_{r'i\omega} + t_{r'ik_{r'i}\omega} \leq t'}} \pi^{(3)}_{r'k_{r'i}t'\omega} \leq \lambda_{rit\omega} \qquad r \in \mathcal{R}, i \in \mathcal{I}_r : (i,0) \in \mathcal{A}_r,\ t \in \mathcal{T}_{ri0\omega} \quad \text{(B.17)}$$

Similar to Case 4-2 in the evaluation of constraint (5.43), all $\pi^{(3)}_{r'k_{r'i}t'\omega}$ in (B.17) are equal to 0. Thus, (B.17) comes down to $\lambda_{ri(t^*_{i\omega})\omega} \leq \lambda_{rit\omega}$ which holds because $\lambda_{rit\omega}$ is non-decreasing in terms of t and we have assumed $t \geq t^*_{i\omega}$.

Case 3-2) In this case, we assume that $r \in \mathcal{CR}_{i\omega}$ and $t \in \mathcal{T}_{rij\omega} : t < t^*_{i\omega}$ hold. In this case, constraint (5.44) reduces to the following relation by substituting $\pi^{(3)}_{rit'\omega}$ using relation (5.53).

$$\pi^{(1)}_{ri0\omega} + \pi^{(4)}_{ri(t^*_{i\omega})\omega} \leq \lambda_{rj(t^*_{i\omega})\omega} \qquad r \in \mathcal{R}, i \in \mathcal{I}_r : (i,0) \in \mathcal{A}_r, t \in \mathcal{T}_{ri0\omega} \quad \text{(B.18)}$$

In the replacement of $\pi^{(3)}_{rit'\omega}$ using relation (5.53), we note that $\hat{x}_{ri0} = 1$ holds. By replacing $\pi^{(1)}_{ri0\omega}$ and $\pi^{(3)}_{rit^*_{i\omega}\omega}$ using (5.52) and (5.49), (B.18) reduces the following relation.

$$\alpha_{ri\omega} \sum_{\substack{r' \in \mathcal{NR}_{i\omega}: k_{r'i} \neq 0 \\ t' \in \mathcal{T}_{k_{r'i}\omega}: t^*_{i\omega} + s_{r'i\omega} + t_{r'ik_{r'i}\omega} \leq t'}} \pi^{(3)}_{r'k_{r'i}t'\omega} \leq 0 \qquad r \in \mathcal{R}, i \in \mathcal{I}_r : (i,0) \in \mathcal{A}_r, t \in \mathcal{T}_{ri0\omega} \text{(B.19)}$$

The left-hand side of relation (B.19) is equal to 0, as discussed in Case 4-1 in the evaluation of constraint (5.43). Therefore, relation (B.19) comes down to $0 \leq 0$ that holds.

Case 3-3) In this case, we assume that $r \in \mathcal{NR}_{i\omega}$ holds. In this case, with respect to (5.58), all terms $\pi^{(1)}_{ri0\omega}$ and $\pi^{(3)}_{rit'\omega}$ in constraint (5.44) are equal to 0. By substituting $\pi^{(4)}_{rit\omega}$ using relation (5.50), constraint (5.44) reduces to $\lambda_{ri(min(t,t^*_{i\omega}))\omega} \leq \lambda_{rit\omega}$ which holds for both cases of $t < t^*_{i\omega}$ and $t \geq t^*_{i\omega}$ because $\lambda_{rit\omega}$ is non-decreasing in terms of t.

Evaluation of constraint (5.45)

Case 1) Let's assume customer i is a non-critical customer, i.e., $i \in \mathcal{NT}_\omega$. In this case, with respect to (5.58), all $\pi^{(4)}_{rit\omega}$ in constraint (5.45) are equal to 0 and therefore the constraint reduces to $0 \leq c^d_{it}$ which holds for all parameters c^d_{it}.

Case 2) Let's assume customer i is a critical customer, i.e., $i \in CT$ and also $t \geq t^*_{i\omega}$ holds. First, we rewrite constraint (5.45) as follows.

$$-\sum_{\substack{r \in \mathcal{NR}_{i\omega}: \\ \hat{x}_{ri0}=0}} \pi^{(4)}_{rit\omega} - \sum_{\substack{r \in \mathcal{NR}_{i\omega}: \\ \hat{x}_{ri0} \neq 0}} \pi^{(4)}_{rit\omega} - \sum_{r \in \mathcal{CR}_{i\omega}} \pi^{(4)}_{rit\omega} \leq c^d_{it} \qquad i \in \mathcal{CT}_\omega, t \in \mathcal{T}_{i\omega} : t \geq t^*_{i\omega} \qquad \text{(B.20)}$$

In the first summation of (B.20), using relation (5.50), we have $\pi^{(4)}_{rit\omega} = -\sum_{\substack{t' \in \mathcal{T}_{k_{rj}\omega}: \\ t + s_{rj\omega} + t_{rjk_{rj}\omega} \leq t'}} \pi^{(3)}_{rjk_{rj}t'\omega} = 0$. The last equality is valid with respect to discussion provided in Case 4-2 in the evaluation of constraint (5.43) for $t \geq t^*_{i\omega}$. By replacing $\pi^{(4)}_{rit\omega}$ in (B.20) using relations (5.49) and (5.50), we obtain the following relation.

$$c^d_{i(t^*_{i\omega})} - \sum_{r \in \mathcal{CR}_{i\omega}} \left(\alpha_{ri\omega} \sum_{\substack{r' \in \mathcal{NR}_{i\omega}: k_{r'i} \neq 0 \\ t' \in \mathcal{T}_{k_{r'i}\omega}: t + s_{r'i\omega} + t_{r'ik_{r'i}\omega} \leq t'}} \pi^{(3)}_{r'k_{r'i}t'\omega} \right) \leq c^d_{it} \quad r \in \mathcal{CT}_\omega, t \in \mathcal{T}_{i\omega} : t \geq t^*_{i\omega} \text{(B.21)}$$

In the recent replacement, we simplified the left-hand side of relation (B.21) using

$\sum_{r\in\mathcal{CR}_{i\omega}} \alpha_{ri\omega} = 1$. Also, we know that $\sum_{\substack{r'\in\mathcal{NR}_{i\omega}:k_{r'i}\neq 0 \\ t'\in\mathcal{T}_{k_{r'i}\omega}:t+s_{r'i\omega}+t_{r'ik_{r'i}\omega}\leq t'}} \pi^{(3)}_{r'k_{r'i}t'\omega}$ for $t \geq t^*_{i\omega}$ as discussed before in Case 4-2 in the evaluation of constraint (5.43). Therefore, relation (B.21) reduces to $c^d_{it^*_{i\omega}} \leq c^d_{it}$ which holds because c^d_{it} is non-decreasing in terms of t.

Case 3) Let's assume customer i is a critical customer, i.e., $i \in \mathcal{CT}_\omega$ and $t < t^*_{i\omega}$ holds. Similar to the previous case, we can rewrite constraint (5.45) as follows.

$$-\sum_{\substack{r\in\mathcal{NR}_{i\omega}: \\ \hat{x}_{ri0}=0}} \pi^{(4)}_{rit\omega} - \sum_{\substack{r\in\mathcal{NR}_{i\omega}: \\ \hat{x}_{ri0}\neq 0}} \pi^{(4)}_{rit\omega} - \sum_{r\in\mathcal{CR}_{i\omega}} \pi^{(4)}_{rit\omega} \leq c^d_{it} \qquad i \in \mathcal{CT}_\omega, t \in \mathcal{T}_{i\omega} : t < t^*_{i\omega} \tag{B.22}$$

By replacing $\pi^{(4)}_{rit\omega}$ using (5.49) and (5.50), we obtain the following relation.

$$c^d_{i(t^*_{i\omega})} + \sum_{r\in\mathcal{NR}_{i\omega}:k_{r'i}=0} \lambda_{r'i(t^*_{i\omega})\omega} \leq c^d_{it} + \sum_{r\in\mathcal{NR}_{i\omega}:k_{r'i}=0} \lambda_{rit\omega} \qquad i \in \mathcal{CT}_\omega, t \in \mathcal{T}_{i\omega} : t < t^*_{i\omega} \tag{B.23}$$

Relation (B.23) holds because $c^d_{it} + \sum_{r\in\mathcal{NR}_{i\omega}:k_{r'i}=0} \lambda_{rit\omega}$ is non-decreasing in terms of t and we assumed $t < t^*_{i\omega}$.

Evaluation of constraint (5.46)

For those $\pi^{(3)}_{rjt\omega}$ that are set to 0 by relation (5.58), constraint (5.46) is clearly satisfied. For those variables $\pi^{(3)}_{rjt\omega}$ evaluated by relation (5.53), $\left(\pi^{(4)}_{rjt\omega} - \pi^{(4)}_{rj(t+1)\omega}\right) + F_{(\hat{x}_{rj0}=1)}\left(\lambda_{rj(t+1)\omega} - \lambda_{rjt\omega}\right) \geq 0$ holds because $-\pi^{(4)}_{rjt\omega}$ and $\lambda_{rjt\omega}$ are non-decreasing in terms of t. As $\pi^{(3)}_{rjt\omega}$ in (5.53) depends on $\pi^{(3)}_{rk_{rj}(t+s_{rj\omega}+t_{rjk_{rj}\omega})\omega}$, by induction and starting from cases that $F_{[(\hat{x}_{rj0}\neq 1)\,\&\,(t+s_{rj\omega}+t_{rjk_{rj}\omega})\in\mathcal{T}_{k_{rj}\omega}]} = 0$, one can easily show that $\pi^{(3)}_{rjt\omega} \geq 0$ holds.

B.13 Proof of Lemma 5.8

First, we compute the objective value of subproblem (SP$_\omega$) and then we show that the objective value of Model (D$_\omega$) for the dual solution obtained by (5.49)-(5.58) is equal to the objective value of subproblem (SP$_\omega$).

With respect to Theorem 5.4, we know that all $u_{rijt\omega}$ and $v_{it\omega}$ variables take binary values. We denote the optimal values of $u_{rijt\omega}$ and $v_{it\omega}$ by $u^*_{rijt\omega}$ and $v^*_{it\omega}$ respectively. Also, let $t^*_{i\omega}$ denote the start time of serving customer i in scenario ω. Thus, we have $u^*_{rijt^*_{i\omega}\omega} = 1$, $u^*_{rijt\omega} = 0$ for $t \neq t^*_{i\omega}$ and $v^*_{i(t^*_{i\omega})\omega} = 1$, $v^*_{it\omega} = 0$ for $t \neq t^*_{i\omega}$.

$$\textit{Optimal objective value of subproblem } (SP_\omega) = \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\(i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0\omega}} c^o_{rit\omega} u^*_{ri0t\omega} + \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}_{i\omega}} c^d_{it} v^*_{it\omega} + \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\(i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0\omega}} c^w_r f_{rit\omega} u^*_{ri0t\omega}$$

Using $\lambda_{rit\omega} = c^o_{rit\omega} + c^w_r f_{rit\omega}$, we can simplify the above relation as follows.

$$\textit{Optimal objective value of subproblem } (SP_\omega) = \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\(i,0)\in\mathcal{A}_r}} \sum_{t\in\mathcal{T}_{ri0\omega}} \lambda_{rit\omega} u^*_{ri0t\omega} + \sum_{i\in\mathcal{I}} \sum_{t\in\mathcal{T}_{i\omega}} c^d_{it} v^*_{it\omega} = \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega} + \sum_{i\in\mathcal{I}} c^d_{i(t^*_{i\omega})} = \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega} + \sum_{i\in\mathcal{CT}_\omega} c^d_{i(t^*_{i\omega})}$$

Now, we compute the objective value of the Model (D$_\omega$) for the dual solution obtained by (5.49)-(5.58).

$$\textit{The objective value of Model } (D_\omega) = \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r}} \hat{x}_{rij}\pi^{(1)}_{rij\omega} + \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \hat{x}_{r0i}=1}} \sum_{t\in\mathcal{T}^{dep}_{ir}} \hat{y}_{rit}\pi^{(2)}_{rit\omega}$$

$$= \sum_{r\in\mathcal{R}} \sum_{\substack{i,j\in(\mathcal{I}_r\cup\{0\}):\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega} + \sum_{r\in\mathcal{R}} \sum_{\substack{i\in\mathcal{I}_r:\\ \hat{x}_{r0i}=1}} \sum_{\substack{t\in\mathcal{T}^{dep}_{ir}:\\ \hat{y}_{rit}=1}} \pi^{(2)}_{rit\omega} = \sum_{i,j\in(\mathcal{I}\cup\{0\}):} \sum_{\substack{r\in\mathcal{R}_i:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega}$$

$$= \sum_{\substack{i,j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{CT}_\omega}} \sum_{\substack{r\in\mathcal{R}_i:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega} + \sum_{\substack{i,j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{NT}_\omega}} \sum_{\substack{r\in\mathcal{R}_i:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega}$$

$$= \sum_{\substack{i,j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{CT}_\omega}} \sum_{\substack{r\in\mathcal{CR}_{i\omega}:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega} + \sum_{\substack{i,j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{CT}_\omega}} \sum_{\substack{r\in\mathcal{NR}_{i\omega}:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega} + \sum_{\substack{i,j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{NT}_\omega}} \sum_{\substack{r\in\mathcal{R}_i:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \pi^{(1)}_{rij\omega}$$

$$= \sum_{\substack{i\in\mathcal{I},j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{CT}_\omega}} \sum_{\substack{r\in\mathcal{CR}_{i\omega}:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \left(\alpha_{ri\omega}\left(c^d_{i(t^*_{i\omega})} + \sum_{r'\in\mathcal{NR}_{i\omega}:k_{r'i}=0} \lambda_{r'i(t^*_{i\omega})\omega}\right) + \lambda_{ri(t^*_{i\omega})\omega}F_{(\hat{x}_{ri0}=1)}\right)$$
$$+ \sum_{i\in\mathcal{I}\cap\mathcal{NT}_\omega} \sum_{\substack{r\in\mathcal{R}_i:\\ \hat{x}_{ri0}=1}} \lambda_{rit^*_{i\omega}\omega}$$

$$= \sum_{\substack{i\in\mathcal{I},j\in(\mathcal{I}\cup\{0\}):\\ \&\,i\in\mathcal{CT}_\omega}} \sum_{\substack{r\in\mathcal{CR}_{i\omega}:\\(i,j)\in\mathcal{A}_r\,\&\,\hat{x}_{rij}=1}} \left(\alpha_{ri\omega}\left(c^d_{i(t^*_{i\omega})} + \sum_{r'\in\mathcal{NR}_{i\omega}:k_{r'i}=0} \lambda_{r'i(t^*_{i\omega})\omega}\right)\right)$$

$$+ \sum_{\substack{i\in\mathcal{I}, j\in(\mathcal{I}\cup\{0\}): \\ \& \, i\in\mathcal{CT}_\omega}} \sum_{\substack{r\in\mathcal{CR}_{i\omega}: \\ (i,j)\in\mathcal{A}_r \,\&\, \hat{x}_{rij}=1}} \left(\lambda_{ri(t^*_{i\omega})\omega} F_{(\hat{x}_{ri0}=1)}\right) + \sum_{i\in\mathcal{I}\cap\mathcal{NT}_\omega} \sum_{\substack{r\in\mathcal{R}_i: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega}$$

$$= \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} \sum_{r\in\mathcal{CR}_{i\omega}} \left(\alpha_{ri\omega}\left(c^d_{i(t^*_{i\omega})} + \sum_{r'\in\mathcal{NR}_{i\omega}: k_{r'i}=0} \lambda_{r'i(t^*_{i\omega})\omega}\right)\right) + \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} \sum_{\substack{r\in\mathcal{CR}_{i\omega}: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega}$$

$$+ \sum_{i\in\mathcal{I}\cap\mathcal{NT}_\omega} \sum_{\substack{r\in\mathcal{R}_i: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega}$$

$$= \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} \left(c^d_{i(t^*_{i\omega})} + \sum_{r'\in\mathcal{NR}_{i\omega}: k_{r'i}=0} \lambda_{r'i(t^*_{i\omega})\omega}\right) + \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} \sum_{\substack{r\in\mathcal{CR}_{i\omega}: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega} + \sum_{i\in\mathcal{I}\cap\mathcal{NT}_\omega} \sum_{\substack{r\in\mathcal{R}_i: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega}$$

$$= \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} c^d_{i(t^*_{i\omega})} + \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} \sum_{\substack{r\in\mathcal{NR}_{i\omega}: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega} + \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} \sum_{\substack{r\in\mathcal{CR}_{i\omega}: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega} + \sum_{i\in\mathcal{I}\cap\mathcal{NT}_\omega} \sum_{\substack{r\in\mathcal{R}_i: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega}$$

$$= \sum_{i\in\mathcal{I}\cap\mathcal{CT}_\omega} c^d_{i(t^*_{i\omega})} + \sum_{i\in\mathcal{I}} \sum_{\substack{r\in\mathcal{R}_i: \\ \hat{x}_{ri0}=1}} \lambda_{ri(t^*_{i\omega})\omega} = Optimal\ objective\ value\ of\ subproblem\,(SP_\omega)$$

APPENDIX C SUPPLEMENTS OF ARTICLE 3

C.1 Proof of Lemma 6.1

We use the following notation.

$(\mathcal{U},\mathcal{W})$: The uncertainty set that is defined as $(\mathcal{U},\mathcal{W}) = \{(u,w)|$ constraints (6.25)-(6.28) are satisfied$\}$.

$\mathcal{J}$: The index set of $(\mathcal{U},\mathcal{W})$ that is defined as $\mathcal{J} = \{1,2,...,|(\mathcal{U},\mathcal{W})|\}$ where $|(\mathcal{U},\mathcal{W})|$ represents the cardinality of $(\mathcal{U},\mathcal{W})$.

(u^j, w^j) : The j-th member of $(\mathcal{U},\mathcal{W})$.

We also define $f_k(x,w)$ and $g_k(x,e_{ks})$ as follows.

$$f_k(x,w) = \min_{y_k} c_{2k}^{\intercal} y_k \tag{C.1}$$

Subject to :

$$A_k x + \sum_{k\in\mathcal{K}} e_{ks} w_{ks} + C_k y_k \le b_k \qquad k\in\mathcal{K} \tag{C.2}$$

$$y_k \in \mathcal{Y}_k \qquad k\in\mathcal{K} \tag{C.3}$$

$$g_k(x,e_{ks}) = \min_{y'_{ks}} c_{2k}^{\intercal} y'_{ks} \tag{C.4}$$

Subject to :

$$A_k x + e_{ks} + C_k y'_{ks} \le b_k \qquad k\in\mathcal{K} \tag{C.5}$$

$$y'_{ks} \in \mathcal{Y}_k \qquad k\in\mathcal{K} \tag{C.6}$$

For each adversarial scenario $(y,w) \in (\mathcal{U},\mathcal{W})$ with index $j \in \mathcal{J}$, with respect to constraints (6.17) and (6.25), exactly one of the variables w^j_{ks} $s \in \mathcal{S}_k$ is equal to 1 for each $k\in\mathcal{K}$. Let s_j denote the index in $\mathcal{S}_k$ for which $w^j_{ks_j}$ is equal to 1. Therefore we have the following relations.

$$w^j_{ks_j} = 1 \qquad j\in\mathcal{J} \tag{C.7}$$

$$w^j_{ks} = 0 \qquad j\in\mathcal{J}, s\ne s_j \tag{C.8}$$

In the following we prove the *if*-statement of Lemma 1. The *only if*-statement of this lemma can be proven in a reverse direction. Assume that $\hat{x}$ is a first-stage feasible solution of Model (P2). In the following we separately prove that

- $\hat{x}$ is also a first-stage feasible solution of Model (P3).
- The objective values of (P2) and (P3) for this fixed first-stage solution are the same if $\max\limits_{u,w}$ and $\min\limits_{y'}$ are solved optimally.

Proof of Part 1 : Since Model (P2) is feasible, there is at least a feasible second-stage policy $\langle \alpha_{kj} \rangle_{(k \in \mathcal{K})}$ for each $j \in \mathcal{J}$ such that

$$A_k \hat{x} + \sum_{s \in \mathcal{S}_k} e_{ks} w^j_{ks} + C_k \alpha_{kj} \leq b_k \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{C.9}$$

$$\alpha_{kj} \in \mathcal{Y}_k \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{C.10}$$

Using (C.7) and (C.8), we can rewrite relations (C.9)-(C.10) as follows.

$$A_k \hat{x} + e_{ks_j} + C_k \alpha_{kj} \leq b_k \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{C.11}$$

$$\alpha_{kj} \in \mathcal{Y}_k \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{C.12}$$

Relations (C.11)-(C.12) demonstrate that for each $k \in \mathcal{K}$ and $s \in \mathcal{S}_k$ there is at least one $j \in \mathcal{J}$ such that for $y'_{ks} = \alpha_{kj}$ constraints $A_k x + e_{ks} + C_k y'_{ks} \leq b_k$ and $y'_{ks} \in \mathcal{Y}_k$ are satisfied. Therefore, $\hat{x}$ is also a first-stage feasible solution of Model (P3).

Proof of Part 2 : To prove that the objective values of (P2) and (P3) for the fixed first-stage solution $\hat{x}$ are the same, it is enough to prove that relation (C.13) or its equivalent, relation (C.14), holds.

$$c_1^\intercal \hat{x} + \max_{(u,w) \in (\mathcal{U},\mathcal{W})} \left(\sum_{k \in \mathcal{K}} f_k(\hat{x}, w) \right) = c_1^\intercal \hat{x} + \max_{(u,w) \in (\mathcal{U},\mathcal{W})} \left(\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} g_k(\hat{x}, e_{ks}) w_{ks} \right) \tag{C.13}$$

$$\max_{(u,w) \in (\mathcal{U},\mathcal{W})} \left(\sum_{k \in \mathcal{K}} f_k(\hat{x}, w) \right) = \max_{(u,w) \in (\mathcal{U},\mathcal{W})} \left(\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} g_k(\hat{x}, e_{ks}) w_{ks} \right) \tag{C.14}$$

Moreover, regarding (C.7) and (C.8), in constraint (C.2) of $f_k(\hat{x}, w^j)$ we can substitute $\sum\limits_{s \in \mathcal{S}_k} e_{ks} w^j_{ks}$ by e_{ks_j}. It is then clear that mathematical programs corresponding to $g_k(\hat{x}, e_{ks_j})$ and $f_k(\hat{x}, w^j)$ have the same structure and following relations hold.

$$g_k(\hat{x}, e_{ks_j}) = f_k(\hat{x}, w^j) \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{C.15}$$

$$\operatorname*{argmin}_{y'_{ks}} \left(g_k(\hat{x}, e_{ks_j}) \right) = \operatorname*{argmin}_{y_k} \left(f_k(\hat{x}, w^j) \right) \qquad k \in \mathcal{K}, j \in \mathcal{J} \tag{C.16}$$

The following stream of equalities proves the validity of (C.14). In the following relations the second equality is obtained using (C.15). The third equality is valid because of (C.7)-(C.8).

$$\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}} f_k(\hat{x},w)\right) = \max_{j\in\mathcal{J}}\left(\sum_{k\in\mathcal{K}} f_k(\hat{x},w^j)\right) = \max_{j\in\mathcal{J}}\left(\sum_{k\in\mathcal{K}} g_k(\hat{x},e_{ks_j})\right) =$$
$$\max_{j\in\mathcal{J}}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w^j_{ks}\right) = \max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w_{ks}\right)$$

In addition, relation (C.16) shows that we can obtain the second-stage optimal policies for variables y_k in Model (P2) from the optimal values of variables y'_{ks}.

C.2 Proof of Theorem 6.2

As discussed in Appendix C.1, we can present the inner max problem in Model (P3) by

$$\max_{(u,w)\in(\mathcal{U},\mathcal{W})}\left(\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k} g_k(\hat{x},e_{ks})w_{ks}\right) \tag{C.17}$$

where $g_k(\hat{x},e_{ks})$ is defined as follows.

$$g_k(x,e_{ks}) = \min_{y'_{ks}} c_{2k}^{\intercal} y'_{ks} \tag{C.18}$$
$$A_k x + e_{ks} + C_k y'_{ks} \le b_k \qquad k\in\mathcal{K} \tag{C.19}$$
$$y'_{ks} \in \mathcal{Y}_k \qquad k\in\mathcal{K} \tag{C.20}$$

It is clear that the optimal values of vectors y'_{ks} for $k\in\mathcal{K}, s\in\mathcal{S}_k$ are independent of $(u,w)\in(\mathcal{U},\mathcal{W})$ and are defined by

$$y'^{*}_{ks} = \arg\min_{y'_{ks}\in\mathcal{G}_{ks}} (c_{2k}^{\intercal} y'_{ks}) \qquad k\in\mathcal{K}, s\in\mathcal{S}_k \tag{C.21}$$

where $\mathcal{G}_{ks} = \{y'_{ks}\in\mathcal{Y}_k | A_k\hat{x} + e_{ks} + C_k y'_{ks} \le b_k\}$. Therefore, because of the independence of $y'_{ks}, k\in\mathcal{K}, s\in\mathcal{S}_k$ from $(u,w)\in(\mathcal{U},W)$, we can swap $\max\limits_{(u,w)}$ and $\min\limits_{y'}$ in Model (P3) and Theorem 6.2 is proven.

C.3 Proof of Theorem 6.3

Consider the following problem.

$$(\mathrm{MP}') \qquad \min_{(x,y')\in(\mathcal{X},\mathcal{Y}')} \left(c_1^{\intercal} x + \max_{(y,w)\in(\mathcal{U},\mathcal{W})'} \left(\sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} c_{2k}^{\intercal} y'_{ks} w_{ks} \right)\right) \tag{C.22}$$

where $(\mathcal{U},\mathcal{W})' = \{(u^j, w^j), j = 1, 2, ..., m\}$. Since $(\mathcal{U},\mathcal{W})' \subseteq (\mathcal{U},\mathcal{W})$ the optimal objective value of Model (MP$'$) is a valid lower bound for the optimal objective value of the original robust problem (P5). In the following we demonstrate that (MP$'$) is equivalent to (MP). By writing the convex combination of m scenarios (u^j, w^j), Model (MP$'$) can be rewritten as follows.

$$(\mathrm{MP}'') \qquad \min_{(x,y')\in(\mathcal{X},\mathcal{Y}')} \left(c_1^{\intercal} x + \max_{\lambda} \left(\sum_{j=1}^{m} \lambda_j \left(\sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} c_{2k}^{\intercal} y'_{ks} \hat{w}_{ks}^{j} \right)\right)\right) \tag{C.23}$$

$$\sum_{j=1}^{m} \lambda_j = 1 \tag{C.24}$$

$$\lambda_j \geq 0 \qquad j = 1, 2, ..., m. \tag{C.25}$$

In Model (MP$''$), for a fixed value of (x, y'), the inner max problem is a linear programming model and one of its extreme points will be the optimal solution. Each extreme point of this model corresponds to one of the scenarios (u^j, w^j). Therefore, Model (MP$''$) is equivalent to Model(MP$'$). By dualizing the inner max problem in Model (MP$''$) and assuming θ as the dual variables of constraint (C.24) we obtain Model (MP) and Theorem 6.3 is proven.

C.4 Proof of Theorem 6.4

To prove that the Benders algorithm without stopping conditions converges in at most $|\mathcal{W}|+1$ iterations, it is enough to show that if the algorithm visits an adversarial scenario with a repeated vector w in the subproblem, then the optimal solution is found and the Benders algorithm is converged. Let's denote this adversarial scenario by $(\hat{u}, \hat{w})$. We also assume that the algorithm obtains solution $(\hat{x}, \hat{y}')$ by solving the master problem just before the subproblem in which scenario $(\hat{u}, \hat{w})$ is found. Since in scenario $(\hat{u}, \hat{w})$, vector w is repeated, the above master problem includes an instance of constraint (6.42) corresponding to vector $\hat{w}$. Furthermore, since $(\hat{x}, \hat{y}')$ is a feasible solution in the master problem we have

$$c_1^{\intercal} \hat{x} + \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} c_{2k}^{\intercal} \hat{y}'_{ks} \hat{w}_{ks} \leq \theta^* \leq Opt \tag{C.26}$$

where θ^* is the optimal solution of the master problem in this iteration and Opt is the optimal objective value of the robust problem. Besides, in the Benders algorithm without stopping conditions for the master problem and subproblem, the adversarial scenario $(\hat{u}, \hat{w})$ is visited in the subproblem if it is the optimal solution of the subproblem. Therefore, we have

$$Opt \leq c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} \tag{C.27}$$

Relation (C.27) is valid because the optimal objective value of the subproblem is a valid upper bound for the optimal objective value of the robust problem. Consequently, (C.26)-(C.27) results in relation (C.28).

$$Opt = c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} \tag{C.28}$$

This relation means that $(\hat{x}, \hat{y}')$ is the optimal solution of the robust problem and the Benders algorithm is converged in at most $|\mathcal{W}| + 1$ iterations. Since for each $w \in \mathcal{W}$ there is at least one $u \in \mathcal{U}$ satisfying $(u, w) \in (\mathcal{U}, \mathcal{W})$, $|\mathcal{W}| + 1$ is bounded above by $n + 1$.

Notation used in appendices C.5 to C.9

We use the following notation in the proofs of appendices C.5 to C.9.

$\mathcal{W}$:	The set of vectors w for which there is $u \in \mathcal{U}$ such that $(u, w) \in (\mathcal{U}, \mathcal{W})$.
n :	The number of adversarial scenarios in $(\mathcal{U}, \mathcal{W})$.
n' :	The number of unique vectors w that the algorithm visits in the subproblem before it converges.
n'' :	The number of times that the algorithm visits an already encountered vector w in the subproblem before it converges.
ε :	A positive constant used in stopping conditions of the master problem and subproblem.
$MP(i)$:	The master problem in iteration i.
$SP(i)$:	The subproblem in iteration i.
Opt :	The optimal objective value of the original robust problem.
U_i^{MP} :	The upper bound of the master problem in iteration i.
O_i^{MP} :	The optimal objective value of the master problem in iteration i.
L_i^{MP} :	The lower bound of the master problem in iteration i.
U_i^{SP} :	The upper bound of the subproblem in iteration i.
O_i^{SP} :	The optimal objective value of the subproblem in iteration i.
L_i^{SP} :	The lower bound of the subproblem in iteration i.
$f(j)$:	The iteration in which for the j-th times the algorithm generates an adversarial scenario with a new vector w in the subproblem.
$g(i)$:	The iteration in which for the i-th times the algorithm re-visits any of the generated vectors w in the subproblem.
I_i :	An indicator that is equal to 1 if in iteration i the algorithm generates an adversarial scenario with a repeated vector w, 0 otherwise.

C.5 Proof of Lemma 6.2

Let $(\hat{x}, \hat{y}')$ and $\hat{\theta}$ respectively denote the solution and the objective value of the master problem in iteration $i - 1$. Furthermore, let $(\hat{u}, \hat{w})$ denote the adversarial scenario with the repeated vector $w = \hat{w}$ found in the subproblem in iteration i. Since vector $w = \hat{w}$ is repeated, we have already included an instance of constraint (6.42) corresponding to this vector in the master problem in iteration $i-1$ and the following relation holds.

$$\hat{\theta} \geq c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} \tag{C.29}$$

The Benders algorithm applies solution $(\hat{x}, \hat{y}')$ to modify the objective function of the subproblem in iteration i. If $(\hat{u}, \hat{w})$ is not the optimal solution of subproblem then it means that in the subproblem the following stopping condition is satisfied.

$$c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} \geq \hat{\theta} + \varepsilon \tag{C.30}$$

Obviously relation (C.30) is in contrast with (C.29) and we conclude that if the algorithm visits an adversarial scenario with a repeated vector w in the subproblem, this scenario is the optimal solution of the subproblem. To prove that the optimal objective value of the subproblem is equal to the upper bound of the master problem in iteration $i-1$, we have to show that in the master problem, an instance of constraint (6.42) corresponding to the repeated vector $\hat{w}$ is binding. If for another adversarial scenario with a different repeated vector $w = w'$, constraint (6.42) is binding, then we must have $c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} \hat{w}_{ks} < c_1^\intercal \hat{x} + \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} c_{2k}^\intercal \hat{y}'_{ks} w'_{ks}$ which is a contradiction regarding the optimality of $(\hat{u}, \hat{w})$ in the subproblem in iteration i. Therefore, if the algorithm finds an adversarial scenario with a repeated vector $\hat{w}$ in the subproblem, the optimal objective value of the subproblem is equal to the upper bound of the recent master problem.

C.6 Proof of Lemma 6.3

Equivalently this lemma states that if in $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations after iteration i the algorithm does not find any adversarial scenario with a repeated vector w then $O_{i+k}^{SP} - Opt \leq \varepsilon$ holds. In iteration i, since the algorithm found an adversarial scenario with a repeated vector w in subproblem $SP(i)$, regarding Lemma 6.2 this adversarial scenario is the optimal solution of the subproblem and $L_i^{SP} = O_i^{SP}$ holds. Furthermore, in the master problem $MP(i)$ that is solved after subproblem $SP(i)$, two cases are possible.

Case 1) $O_i^{MP} > O_i^{SP} - \varepsilon$ holds. First note that $O_i^{MP} < Opt$ is a valid regarding Theorem 6.3. $O_i^{MP} > O_i^{SP} - \varepsilon$ together with $O_i^{MP} < Opt$ results in $Opt > O_i^{SP} - \varepsilon$. The latter relation contradicts with the initial assumption $O_i^{SP} - Opt > \varepsilon$. Therefore, this case does not happen.

Case 2) $O_i^{MP} \leq O_i^{SP} - \varepsilon$ holds. This relation is equivalent to $O_i^{MP} \leq L_i^{SP} - \varepsilon$ with respect to relation $L_i^{SP} = O_i^{SP}$. Regarding $O_i^{MP} \leq L_i^{SP} - \varepsilon$, the stopping condition in master problem $MP(i)$ is satisfied and the master problems stops when it finds a feasible solution with an upper bound U_i^{MP} satisfying the following relation.

$$U_i^{MP} \leq L_i^{SP} - \varepsilon = O_i^{SP} - \varepsilon \tag{C.31}$$

We have assumed that no adversarial scenario with a new vector w is generated in $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations after iteration i. Therefore, in iteration $i+1$ an adversarial scenario with a repeated vector w is generated and regarding Lemma 6.2 we have $O_{i+1}^{SP} = U_i^{MP}$. The recent relation together with (C.31) results in the following relation.

$$O_{i+1}^{SP} \leq O_i^{SP} - \varepsilon \tag{C.32}$$

Similarly we can show that for $k \leq \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ relation (C.33) holds. This is because it is supposed form iteration i to iteration $i + \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ all visited adversarial scenarios have repeated vectors w.

$$O_{i+h}^{SP} \leq O_{i+h-1}^{SP} - \varepsilon \qquad h \in \{1, 2, ..., k\} \tag{C.33}$$

Relation (C.33) is equivalent to (C.34).

$$\frac{O_{i+h}^{SP} - Opt}{\varepsilon} \leq \frac{O_{i+h-1}^{SP} - Opt}{\varepsilon} - 1 \qquad h \in \{1, 2, ..., k\} \tag{C.34}$$

From (C.34) we can simply obtain

$$\frac{O_{i+k}^{SP} - Opt}{\varepsilon} \leq \frac{O_{i+h-1}^{SP} - Opt}{\varepsilon} - k \tag{C.35}$$

and by setting $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ we will have

$$\frac{O_{(i+k)}^{SP} - Opt}{\varepsilon} \leq \frac{O_{(i+h-1)}^{SP} - Opt}{\varepsilon} - \left\lfloor \frac{O_i^{SP} - Opt}{\varepsilon} \right\rfloor \tag{C.36}$$

which is equivalent to

$$O_{(i+k)}^{SP} - Opt \leq \varepsilon \tag{C.37}$$

Therefore, we proved that if in $k = \lfloor (O_i^{SP} - Opt)/\varepsilon \rfloor$ iterations after iteration i the algorithm does not find any adversarial scenario with a repeated vector w then $O^{SP}_{i+k} - Opt \leq \varepsilon$ holds.

C.7 Proof of Lemma 6.4

Three cases are possible.

Case 1) $O^{SP}_i - Opt \leq \varepsilon$ and $O^{SP}_i - O^{MP}_i \geq \varepsilon$ hold. We show that in this case in the next iteration the algorithm generates an adversarial scenario with a new vector w. Because of $O^{SP}_i - O^{MP}_i \geq \varepsilon$, the stopping condition in the master problem in iteration i is satisfied and the following relation holds.

$$U^{MP}_i \leq O^{SP}_i - \varepsilon \leq Opt \tag{C.38}$$

If the algorithm visits an adversarial scenario with a repeated vector w in the subproblem in iteration $i+1$, we must have $U^{MP}_i = O^{SP}_{i+1}$ regarding Lemma 6.2. Then with respect to (C.38), $O^{SP}_{i+1} < Opt$ holds which is a contradiction because the optimal objective value of the subproblem is an upper bound of the optimal objective of the robust problem. Therefore, in this case in the next iteration an adversarial scenario with a new vector w will be generated.

Case 2) $O^{SP}_i - Opt \leq \varepsilon$, $O^{SP}_i - O^{MP}_i \leq \varepsilon$ and $O^{MP}_i < Opt$ hold. We show in the next iteration the algorithm generates an adversarial scenario with a new vector w. Because of $O^{SP}_i - O^{MP}_i \leq \varepsilon$, in the master problem in iteration i there is not any adversarial scenario satisfying the stopping condition. Thus, the master problem is solved optimally and we will have the following relation.

$$O^{MP}_i = U^{MP}_i \tag{C.39}$$

In the subproblem of next iteration, if the algorithm visits an adversarial scenario with a repeated vector w, then regarding Lemma 6.2 we must have relation (C.40).

$$U^{MP}_i = O^{SP}_{i+1} \tag{C.40}$$

Considering the primary assumption $O^{MP}_i < Opt$ and relations (C.39)-(C.40) we must have $O^{SP}_{i+1} < Opt$ which is a contradiction because the optimal objective value of the subproblem is an upper bound of the optimal objective value of the robust problem. Therefore, in this case in iteration $i+1$ the algorithm generates an adversarial scenario with a new vector w.

Case 3) $O^{SP}_i - Opt \leq \varepsilon$, $O^{SP}_i - O^{MP}_i \leq \varepsilon$ and $O^{MP}_i = Opt$ hold. We show that in this case in the next iteration either the Benders algorithm converges or it generates an adversarial scenario with a new vector w. Because of $O_i^{SP} - O_i^{MP} \leq \varepsilon$, in the master problem in iteration i there is not any adversarial scenario satisfying the stopping condition. Therefore, the master problem is solved optimally and relation (C.41) holds.

$$L_i^{MP} = O_i^{MP} = U_i^{MP} \tag{C.41}$$

In the subproblem of iteration $i+1$, the algorithm generates an adversarial scenario with either a new vector w or a repeated vector w. In the latter case regarding Lemma 6.2 we must have relation (C.40). Considering the primary assumption $O_i^{MP} = Opt$ and relations (C.40)-(C.41) we have $L_i^{MP} = Opt = O_{i+1}^{SP}$. This relation demonstrates that the optimal solution of the robust problem is obtained and the Benders algorithm is converged. Therefore, in this case, in the next iteration either the Benders algorithm converges or it generates an adversarial scenario with a repeated vector w.

C.8 Proof of Lemma 6.5

Regarding constraint (6.45) since the algorithm visits an adversarial scenario with a repeated vector w in the subproblem of iteration $g(i_1)$, in any iteration $j \geq g(i_1)$, the inequality $U_j^{MP} \leq O_{g(i_1)}^{SP}$ holds and by setting $j = g(i_2) - 1 \geq g(i_1)$ we obtain the following relation.

$$U_{g(i_2)-1}^{MP} \leq O_{g(i_1)}^{SP} \tag{C.42}$$

Note that $g(i_2) - 1 \geq g(i_1)$ holds because $i_1 < i_2$. Also regarding Lemma 6.2, in the subproblem of iteration $g(i_2)$ that the algorithm has visited an adversarial scenario with a repeated vector w, we have $U_{g(i_2)-1}^{MP} = O_{g(i_2)}^{SP}$. This relation together with relation (C.42) demonstrates the validity of $O_{g(i_2)}^{SP} \leq O_{g(i_1)}^{SP}$.

C.9 Proof of Theorem 6.5

To prove that the Benders algorithm converges in at most $\sum_{j=1}^{n'} (1 + (\lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor + 1) I_{f(j)+1})$ iterations it is enough to show it takes at most $1 + (\lfloor (O_{f(j)+1}^{SP} - Opt)/\varepsilon \rfloor + 1) I_{f(j)+1}$ iterations between visiting j-th and $(j+1)$-th new vector w in the subproblem. Let's assume $j < n'$. Two cases are possible.

Case 1) we have $I_{f(j)+1} = 0$ that means in the iteration $f(j)+1$ the algorithm finds an adversarial scenario with a new vector w. In this case the number of between visiting j-th and $(j+1)$-th new adversarial scenarios is 1.

Case 2) we have $I_{f(j)+1} = 1$ that means in iteration $f(j)+1$ the algorithm visits an adversarial scenario with a repeated vector w. In this case, after visiting the an adversarial scenario with a repeated vector w in iteration $f(j)+1$, with respect to Lemma 6.3 it takes at most $k = \lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor$ to find an adversarial scenario with a new vector w or to have $O^{SP}_{f(j)+1+k} - Opt \leq \varepsilon$. In the latter case, regarding Lemma 6.4, we know that in the next iteration $f(j)+k+2$ either the Benders algorithm converges or an adversarial scenario with a new vector w is found. Since it is assumed that $j < n'$, the Benders algorithm does not converge before finding the $(j+1)$-th adversarial scenario with a new vector w. Thus, we expect that the algorithm generates $(j+1)$-th new vector w by iteration $f(j)+k+2$. In other words, the number of iterations between visiting j-th and $(j+1)$-th new vector w is at most $\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 2$. Therefore, for $j < n'$ the number of iterations between visiting j-th and $(j+1)$-th new vector w is computed by relation (C.43).

$$(1 - I_{f(j)+1}) + \left(\left\lfloor \frac{O^{SP}_{f(j)+1} - Opt}{\varepsilon} \right\rfloor + 2 \right) I_{f(j)+1} \tag{C.43}$$

For $j = n'$ we can use a similar reasoning as presented above for $j < n'$. The difference is that only Case 2 is applicable because regarding the definition of n' no new vector w is visited after visiting the n'-th new vector w. Moreover, when we use Lemmas 6.3 and 6.4 in Case 2, the generation of an adversarial scenario with a new vector w is not an option and we are sure that after finding the n'-th new vector w, the Benders algorithm converges in at most $\left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 2 \right)$ iterations that is the same as (C.43) with respect to $I_{f(j)+1} = 1$ for $j = n'$. Therefore, by summing the number of iterations computed by relation (C.43) from $j = 1$ to $j = n'$ we obtain the following maximum number of iterations.

$$\begin{aligned}
\sum_{j=1}^{n'} \left(1 + \left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1 \right) I_{f(j)+1} \right) &= n' + \sum_{j=1}^{n'} \left(\left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1 \right) I_{f(j)+1} \right) \\
\leq n' + \sum_{j=1}^{n'} \left(\lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1 \right) &= n' \left(\lfloor (O^{SP}_{g(1)} - Opt)/\varepsilon \rfloor + 2 \right) \\
&\leq |\mathcal{W}| \left(\lfloor (O^{SP}_{g(1)} - Opt)/\varepsilon \rfloor + 2 \right)
\end{aligned}$$

Proof of the first inequality : We know that in n' iterations the algorithm visits at least one adversarial scenario with a repeated vector w. $g(1)$ denotes the iteration in which a repeated vector w is visited for the first time. To prove the first inequality it is enough to show the validity of the following relation (C.44).

$$\left\lfloor \frac{O^{SP}_{g(1)} - Opt}{\varepsilon} \right\rfloor + 1 \geq \left(\left\lfloor \frac{O^{SP}_{f(j)+1} - Opt}{\varepsilon} \right\rfloor + 1 \right) I_{f(j)+1} \qquad j \in \{1, 2, ..., n'\} \tag{C.44}$$

As $O^{SP}_{g(1)} \geq Opt$ is a valid relation, (C.44) holds when $I_{f(j)+1}$ equal 0. In the case that $I_{f(j)+1}$ is equal to 1, regarding the definition of g(1) and $I_{f(j)+1}$ we know that $g(1) \leq f(j)+1$. Thus, with respect to Lemma 6.5, we have $O^{SP}_{g(1)} \geq O^{SP}_{f(j)+1}$ that results in $\lfloor (O^{SP}_{g(1)} - Opt)/\varepsilon \rfloor + 1 \geq \lfloor (O^{SP}_{f(j)+1} - Opt)/\varepsilon \rfloor + 1$. Therefore, relation (C.44) is valid.

C.10 An example to show the local optimality of the dual algorithm

Consider the problem $\min_{(x_1,x_2)\in\mathcal{X}}(2x_1 + 1.5x_2 + \max_{(u_1,u_2)\in\mathcal{U}}(x_1u_1 + x_2u_2))$ where

$$\mathcal{U} = \{(u_1,u_2) \in N^2 | u_1 \leq 2,\ u_2 \geq 1,\ 0.99u_1 + 2u_2 \leq 5.98,\ 1.99u_1 + u_2 \geq 2.99\}$$

and

$$\mathcal{X} = \{(x_1,x_2) \in R^2 | x_1 + x_2 = 1,\ (x_1,x_2) \in \{0,1\}^2\}.$$

The solution space of the adversarial variables (u_1, u_2) are four points A, B, C, and D in Figure C.1. The optimal solution of this problem is $(x_1, x_2) = (0, 1)$. For this solution the objective line in $\max_{(u_1,u_2)\in\mathcal{U}}$ is Line L1. This objective line shows that scenarios A and B in the adversarial problem are optimal with a total objective value of 3.5. If we relax the integrality constraints on variables u_1 and u_2 the solution space in the adversarial problem extends to polytope E-B-C-D. In this case, for solution $(x_1, x_2) = (0, 1)$ the optimal adversarial scenario is Point E with an objective value of 4.49. However, for solution $(x_1, x_2) = (1, 0)$ the objective line L2 represents the objective function of the inner max problem. This objective line finds points B and C as the optimal adversarial scenarios with an objective value of 4. In this example, if we apply the dual algorithm to solve this problem the algorithm converges in the first iteration by finding the non-optimal solution $(x_1, x_2) = (1, 0)$.

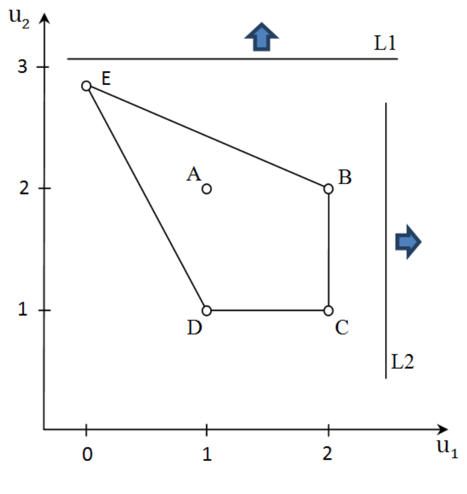

Figure C.1 The solution space of the adversarial variables in the example presented to show the non-optimality of the dual algorithm.

C.11 Non-adjustable nurse planning problem

In a non-adjustable robust problem, the decision maker is not allowed to take recourse actions in the second stage. Therefore, to formulate and solve the non-adjustable nurse planning problem, in Model (6.32)-(6.40) we should set the second-stage variables y'_{ds} to zero. In this case, Model (6.32)-(6.40) reduces to the following problem.

$$\min_{x}\left(\max_{u,w}\left(\sum_{d\in\mathcal{D}} c_1 x_d\right)\right) \tag{C.45}$$

$$\delta x_d \geq \rho \times s \qquad d\in\mathcal{D}, s\in\mathcal{S}_d \tag{C.46}$$

$$x_d \geq 0, \text{integer} \qquad d\in\mathcal{D} \tag{C.47}$$

$$\sum_{s\in\mathcal{S}_d} w_{ds} = 1 \qquad d\in\mathcal{D} \tag{C.48}$$

$$\sum_{t\in\mathcal{T}}\sum_{p\in\mathcal{P}_{td}} u_{tp} = \sum_{s\in\mathcal{S}_d} s w_{ds} \qquad d\in\mathcal{D} \tag{C.49}$$

$$\sum_{p\in\mathcal{P}_t} u_{tp} = 1 \qquad t\in\mathcal{T} \tag{C.50}$$

$$w_{ds}\in\{0,1\} \qquad d\in\mathcal{D}, s\in\mathcal{S}_d \tag{C.51}$$

$$u_{tp}\in\{0,1\} \qquad t\in\mathcal{T}, p\in\mathcal{P}_t \tag{C.52}$$

It is clear that we can remove adversarial variables u_{tp} and w_{ds} from the above model. Also we can consider constraint (C.46) only for the highest value $s\in\mathcal{S}_d$ for each $d\in\mathcal{D}$. Therefore, assuming that $s_{max,d}$ denotes the highest value $s\in\mathcal{S}_d$, the non-adjustable nurse planning problem reduces to the following model.

$$\min_{x}\left(\sum_{d\in\mathcal{D}} c_1 x_d\right) \tag{C.53}$$

$$\delta x_d \geq \rho \times s_{max,d} \qquad d\in\mathcal{D} \tag{C.54}$$

$$x_d \geq 0, \text{integer} \qquad d\in\mathcal{D} \tag{C.55}$$

The above model is trivial and its optimal solution is presented as follows.

$$x^*_d = max\left\{0, \left\lceil\frac{\rho\times s_{max,d}}{\delta}\right\rceil\right\} \qquad d\in\mathcal{D} \tag{C.56}$$

C.12 Details on the number of first-stage nurses in the best and non-adjustable solutions in Tables 6.2 to 6.5.

Table C.1 – Details of the number of first-stage nurses in the best and non-adjustable solutions for instances with a planning horizon of two weeks ($L = 2$).

Data Info.			*The number of first-stage nurses in the best solution*				*The number of first-stage nurses in the non-adjustable solution*			
IF	*OR*	*Sur.*	*Ave*	*Min*	*Max*	*STD*	*Ave*	*Min*	*Max*	*STD*
1.1	1	39	2.63	2.14	2.93	0.22	3.15	2.79	3.71	0.28
	2	79	5.00	4.29	5.86	0.46	6.20	5.50	7.36	0.50
	3	119	7.46	7.14	8.07	0.30	9.10	8.64	9.71	0.36
	4	157	9.26	8.21	10.86	0.80	12.07	11.07	13.57	0.79
	5	202	11.69	9.57	13.36	0.89	15.19	13.43	16.21	0.71
1.3	1	39	2.63	2.14	2.93	0.22	3.15	2.79	3.71	0.28
	2	79	5.00	4.29	5.86	0.46	6.20	5.50	7.36	0.50
	3	119	7.46	7.14	8.07	0.30	9.10	8.64	9.71	0.36
	4	157	9.51	8.00	10.36	0.72	12.07	11.07	13.57	0.79
	5	202	11.89	9.86	12.71	0.78	15.19	13.43	16.21	0.71
1.5	1	39	2.65	2.14	3.00	0.24	3.15	2.79	3.71	0.28
	2	79	5.02	4.50	5.86	0.43	6.20	5.50	7.36	0.50
	3	119	7.51	7.14	8.50	0.45	9.10	8.64	9.71	0.36
	4	157	9.58	8.07	10.50	0.72	12.07	11.07	13.57	0.79
	5	202	12.01	10.00	13.29	0.80	15.19	13.43	16.21	0.71
1.7	1	39	2.67	2.14	3.00	0.24	3.15	2.79	3.71	0.28
	2	79	5.06	4.50	5.86	0.43	6.20	5.50	7.36	0.50
	3	119	7.54	7.14	8.50	0.43	9.10	8.64	9.71	0.36
	4	157	9.76	8.36	10.86	0.70	12.07	11.07	13.57	0.79
	5	202	12.49	10.14	13.79	0.88	15.19	13.43	16.21	0.71
1.9	1	39	2.67	2.14	3.00	0.24	3.15	2.79	3.71	0.28
	2	79	5.06	4.50	5.86	0.43	6.20	5.50	7.36	0.50
	3	119	7.54	7.14	8.50	0.43	9.10	8.64	9.71	0.36
	4	157	9.75	8.43	10.71	0.66	12.07	11.07	13.57	0.79
	5	202	12.89	10.36	15.43	1.24	15.19	13.43	16.21	0.71
Average		119	7.39	6.38	8.31	0.54	9.14	8.29	10.11	0.53

Table C.2 – Details of the number of first-stage nurses in the best and non-adjustable solutions for instances with a planning horizon of three weeks ($L = 3$).

Data Info.			*The number of first-stage nurses in the best solution*				*The number of first-stage nurses in the non-adjustable solution*			
IF	*OR*	*Sur.*	*Ave*	*Min*	*Max*	*STD*	*Ave*	*Min*	*Max*	*STD*
1.1	1	59	3.36	3.00	3.76	0.20	4.31	3.90	4.71	0.23
	2	121	6.09	5.05	7.52	0.67	8.64	7.48	9.86	0.63
	3	182	8.71	7.76	9.52	0.53	12.61	11.57	13.19	0.48
	4	240	11.31	10.67	12.81	0.56	16.57	15.67	17.33	0.56
	5	300	14.17	13.43	15.10	0.54	20.71	19.43	21.95	0.70
1.3	1	59	3.36	3.00	3.76	0.20	4.31	3.90	4.71	0.23
	2	121	6.28	5.43	7.19	0.53	8.64	7.48	9.86	0.63
	3	182	8.95	7.95	9.76	0.52	12.61	11.57	13.19	0.48
	4	240	11.73	11.14	13.00	0.51	16.57	15.67	17.33	0.56
	5	300	14.66	13.90	15.38	0.52	20.71	19.43	21.95	0.70
1.5	1	59	3.48	3.14	3.90	0.23	4.31	3.90	4.71	0.23
	2	121	6.67	5.76	7.95	0.66	8.64	7.48	9.86	0.63
	3	182	9.25	8.48	9.90	0.42	12.61	11.57	13.19	0.48
	4	240	11.78	11.14	13.14	0.55	16.57	15.67	17.33	0.56
	5	300	14.90	14.05	15.52	0.52	20.71	19.43	21.95	0.70
1.7	1	59	3.50	3.14	3.9	0.24	4.31	3.90	4.71	0.23
	2	121	6.86	6.29	7.81	0.46	8.64	7.48	9.86	0.63
	3	182	9.90	9.14	10.57	0.44	12.61	11.57	13.19	0.48
	4	240	12.30	11.52	14.00	0.72	16.57	15.67	17.33	0.56
	5	300	15.90	14.24	20.86	1.77	20.71	19.43	21.95	0.7
1.9	1	59	3.50	3.14	3.90	0.24	4.31	3.90	4.71	0.23
	2	121	6.88	6.24	7.81	0.47	8.64	7.48	9.86	0.63
	3	182	10.83	9.14	12.67	1.16	12.61	11.57	13.19	0.48
	4	240	15.70	12.14	17.00	1.43	16.57	15.67	17.33	0.56
	5	300	20.40	19.19	21.52	0.66	20.71	19.43	21.95	0.70
Average		180	9.62	8.72	10.73	0.59	12.57	11.61	13.41	0.52

Table C.3 – Details of the number of first-stage nurses in the best and non-adjustable solutions for instances with a planning horizon of four weeks ($L = 4$).

Data Info.			*The number of first-stage nurses in the best solution*				*The number of first-stage nurses in the non-adjustable solution*			
IF	*OR*	*Sur.*	*Ave*	*Min*	*Max*	*STD*	*Ave*	*Min*	*Max*	*STD*
1.1	1	80	3.67	3.21	4.18	0.31	5.01	4.36	5.39	0.26
	2	163	6.79	5.93	7.54	0.44	9.88	8.86	10.71	0.56
	3	241	9.45	8.36	9.89	0.45	14.34	13.71	14.71	0.31
	4	318	12.13	11.43	12.93	0.46	18.74	18.18	19.64	0.46
	5	397	15.33	14.68	16.00	0.43	23.09	22.18	23.79	0.48
1.3	1	80	3.83	3.46	4.18	0.22	5.01	4.36	5.39	0.26
	2	163	7.13	6.21	7.96	0.48	9.88	8.86	10.71	0.56
	3	241	9.77	8.57	10.32	0.50	14.34	13.71	14.71	0.31
	4	318	12.74	12.00	13.79	0.52	18.74	18.18	19.64	0.46
	5	397	15.82	15.14	16.46	0.41	23.09	22.18	23.79	0.48
1.5	1	80	3.96	3.57	4.29	0.23	5.01	4.36	5.39	0.26
	2	163	7.47	6.50	7.86	0.38	9.88	8.86	10.71	0.56
	3	241	10.10	8.79	10.79	0.61	14.34	13.71	14.71	0.31
	4	318	13.33	12.29	14.57	0.70	18.74	18.18	19.64	0.46
	5	397	19.44	15.61	23.14	3.16	23.09	22.18	23.79	0.48
1.7	1	80	4.00	3.57	4.29	0.19	5.01	4.36	5.39	0.26
	2	163	7.78	7.21	8.64	0.43	9.88	8.86	10.71	0.56
	3	241	10.75	9.54	11.54	0.53	14.34	13.71	14.71	0.31
	4	318	16.10	12.64	19.39	2.78	18.74	18.18	19.64	0.46
	5	397	22.84	22.00	23.57	0.46	23.09	22.18	23.79	0.48
1.9	1	80	4.01	3.57	4.29	0.18	5.01	4.36	5.39	0.26
	2	163	8.52	7.32	10.50	1.25	9.88	8.86	10.71	0.56
	3	241	14.04	13.39	14.46	0.35	14.34	13.71	14.71	0.31
	4	318	18.44	17.71	19.39	0.50	18.74	18.18	19.64	0.46
	5	397	22.79	21.96	23.54	0.47	23.09	22.18	23.79	0.48
Average		240	11.21	10.19	12.14	0.66	14.21	13.46	14.85	0.42

Table C.4 – Details of the number of first-stage nurses in the best and non-adjustable solutions for instances with a planning horizon of five weeks ($L = 5$).

Data Info.			*The number of first-stage nurses in the best solution*				*The number of first-stage nurses in the non-adjustable solution*			
IF	*OR*	*Sur.*	*Ave*	*Min*	*Max*	*STD*	*Ave*	*Min*	*Max*	*STD*
1.1	1	101	3.76	3.51	3.94	0.17	5.29	5.11	5.49	0.11
	2	202	6.71	6.46	7.11	0.21	10.33	10.00	11.03	0.28
	3	302	10.04	9.46	11.00	0.44	15.31	14.74	16.06	0.41
	4	401	13.04	12.49	13.89	0.41	20.19	19.54	20.89	0.44
	5	503	17.06	16.49	17.86	0.40	25.30	24.49	26.31	0.55
1.3	1	101	4.03	3.71	4.26	0.16	5.29	5.11	5.49	0.11
	2	202	7.29	6.86	7.66	0.26	10.33	10.00	11.03	0.28
	3	302	10.44	9.80	11.26	0.45	15.31	14.74	16.06	0.41
	4	401	13.7	13.34	14.57	0.37	20.19	19.54	20.89	0.44
	5	503	17.51	16.89	18.14	0.40	25.30	24.49	26.31	0.55
1.5	1	101	4.30	3.89	4.66	0.22	5.29	5.11	5.49	0.11
	2	202	7.51	7.03	8.23	0.37	10.33	10.00	11.03	0.28
	3	302	11.59	9.91	15.69	2.05	15.31	14.74	16.06	0.41
	4	401	17.98	13.74	20.4	2.68	20.19	19.54	20.89	0.44
	5	503	25.03	24.37	26.17	0.55	25.30	24.49	26.31	0.55
1.7	1	101	4.23	3.97	4.54	0.16	5.29	5.11	5.49	0.11
	2	202	7.93	7.57	8.77	0.36	10.33	10.00	11.03	0.28
	3	302	11.88	10.8	15.63	1.36	15.31	14.74	16.06	0.41
	4	401	19.99	19.34	20.66	0.45	20.19	19.54	20.89	0.44
	5	503	25.02	24.31	26.00	0.52	25.30	24.49	26.31	0.55
1.9	1	101	4.26	4.06	4.46	0.13	5.29	5.11	5.49	0.11
	2	202	8.89	7.57	9.97	0.97	10.33	10.00	11.03	0.28
	3	302	15.03	14.40	15.63	0.39	15.31	14.74	16.06	0.41
	4	401	19.96	19.34	20.66	0.48	20.19	19.54	20.89	0.44
	5	503	25.01	24.34	26.00	0.52	25.30	24.49	26.31	0.55
Average		302	12.49	11.75	13.49	0.58	15.28	14.78	15.95	0.36