Université de Montréal

Solution Methods for Service Network Design with Resource Management Consideration

par Duc-Minh Vu

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences en vue de

l'obtention du grade de doctorat en Informatique et Recherche Opérationnelle

Juin, 2014

Résumé

La gestion des ressources, équipements, équipes de travail, et autres, devrait être prise en compte lors de la conception de tout plan réalisable pour le problème de conception de réseaux de services. Cependant, les travaux de recherche portant sur la gestion des ressources et la conception de réseaux de services restent limités. La présente thèse a pour objectif de combler cette lacune en faisant l'examen de problèmes de conception de réseaux de services prenant en compte la gestion des ressources. Pour ce faire, cette thèse se décline en trois études portant sur la conception de réseaux.

La première étude considère le problème de *capacitated multi-commodity fixed cost network design with design-balance constraints*(DBCMND). La structure multi-produits avec capacité sur les arcs du DBCMND, de même que ses contraintes *design-balance*, font qu'il apparaît comme sous-problème dans de nombreux problèmes reliés à la conception de réseaux de services, d'où l'intérêt d'étudier le DBCMND dans le contexte de cette thèse. Nous proposons une nouvelle approche pour résoudre ce problème combinant la recherche tabou, la recomposition de chemin, et une procédure d'intensification de la recherche dans une région particulière de l'espace de solutions. Dans un premier temps la recherche tabou identifie de bonnes solutions réalisables. Ensuite la recomposition de chemin est utilisée pour augmenter le nombre de solutions réalisables. Les solutions trouvées par ces deux méta-heuristiques permettent d'identifier un sous-ensemble d'arcs qui ont de bonnes chances d'avoir un statut ouvert ou fermé dans une solution optimale. Le statut de ces arcs est alors fixé selon la valeur qui prédomine dans les solutions trouvées préalablement. Enfin, nous utilisons la puissance d'un solveur de programmation mixte en nombres entiers pour intensifier la recherche sur le problème restreint par le statut fixé ouvert/fermé de certains arcs. Les tests montrent que cette approche est capable de trouver de bonnes solutions aux problèmes de grandes tailles dans des temps raisonnables. Cette recherche est publiée dans la revue scientifique *Journal of heuristics*.

La deuxième étude introduit la gestion des ressources au niveau de la conception de réseaux de services en prenant en compte explicitement le nombre fini de véhicules utilisés à chaque terminal pour le transport de produits. Une approche de solution faisant appel au slope-scaling, la génération de colonnes et des heuristiques basées sur une formulation en cycles est ainsi proposée. La génération de colonnes résout une relaxation linéaire du problème de conception de réseaux, générant des colonnes qui sont ensuite utilisées par le slope-scaling. Le slope-scaling résout une approximation linéaire du problème de conception de réseaux, d'où l'utilisation

d'une heuristique pour convertir les solutions obtenues par le slope-scaling en solutions réalisables pour le problème original. L'algorithme se termine avec une procédure de perturbation qui améliore les solutions réalisables. Les tests montrent que l'algorithme proposé est capable de trouver de bonnes solutions au problème de conception de réseaux de services avec un nombre fixe des ressources à chaque terminal. Les résultats de cette recherche seront publiés dans la revue scientifique *Transportation Science*.

La troisième étude élargie nos considérations sur la gestion des ressources en prenant en compte l'achat ou la location de nouvelles ressources de même que le repositionnement de ressources existantes. Nous faisons les hypothèses suivantes: une unité de ressource est nécessaire pour faire fonctionner un service, chaque ressource doit retourner à son terminal d'origine, il existe un nombre fixe de ressources à chaque terminal, et la longueur du circuit des ressources est limitée. Nous considérons les alternatives suivantes dans la gestion des ressources: 1) repositionnement de ressources entre les terminaux pour tenir compte des changements de la demande, 2) achat et/ou location de nouvelles ressources et leur distribution à différents terminaux, 3) externalisation de certains services. Nous présentons une formulation intégrée combinant les décisions reliées à la gestion des ressources avec les décisions reliées à la conception des réseaux de services. Nous présentons également une méthode de résolution matheuristique combinant le slope-scaling et la génération de colonnes. Nous discutons des performances de cette méthode de résolution, et nous faisons une analyse de l'impact de différentes décisions de gestion des ressources dans le contexte de la conception de réseaux de services. Cette étude sera présentée au XII International Symposium On Locational Decision, en conjonction avec XXI Meeting of EURO Working Group on Locational Analysis, Naples/Capri (Italy), 2014.

En résumé, trois études différentes sont considérées dans la présente thèse. La première porte sur une nouvelle méthode de solution pour le "capacitated multi-commodity fixed cost network design with design-balance constraints". Nous y proposons une matheuristique comprenant la recherche tabou, la recomposition de chemin, et l'optimisation exacte. Dans la deuxième étude, nous présentons un nouveau modèle de conception de réseaux de services prenant en compte un nombre fini de ressources à chaque terminal. Nous y proposons une matheuristique avancée basée sur la formulation en cycles comprenant le slope-scaling, la génération de colonnes, des heuristiques et l'optimisation exacte. Enfin, nous étudions l'allocation des ressources dans la conception de réseaux de services en introduisant des formulations qui modèlent le repositionnement, l'acquisition et la location de ressources, et l'externalisation de certains services. À cet égard, un cadre de solution slope-scaling développé à partir d'une formulation en cycles

est proposé. Ce dernier comporte la génération de colonnes et une heuristique. Les méthodes proposées dans ces trois études ont montré leur capacité à trouver de bonnes solutions.

Mots clés: *Transport de marchandises, conception de réseaux de services, gestion des ressources, méta-heuristiques, algorithme exact, formulation en cycles, la recherche tabou, la composition de chemins, slope-scaling, la génération de colonnes.*

Abstract

Resource management in freight transportation service network design is an important issue that has been studied extensively in recent years. Resources such as vehicles, crews, etc. are factors that can not be ignored when designing a feasible plan for any service network design problem. However, contributions related to resource management issues and service network design are still limited. The goal of the thesis is to fill this gap by taking into account service network design problems with resource management issues. In this thesis, we propose and address three service network design problems that consider resource management.

In the first study, we consider the capacitated multi-commodity fixed cost network design with design-balance constraints which is a basic sub-problem for many service design problems because of the capacitated multi-commodity structure as well as its design-balance property. We propose a three-phase matheuristic that combines tabu-search, path-relinking and an exact-based intensification procedure to find high quality solutions. Tabu-search identifies feasible solutions while path-relinking extends the set of feasible solutions. The solutions found by these two meta-heuristics are used to fix arcs as open or close. An exact solver intensifies the search on a restricted problem derived from fixing arcs. The experiments on benchmark instances show that the solution approach finds good solutions to large-scale problems in a reasonable amount of time. The contribution with regard to this study has been accepted in the *Journal of Heuristics*.

In the second study, together with the consideration of the design of routes to transport a set of commodities by vehicles, we extend resources management by explicitly taking account of the number of available vehicles at each terminal. We introduce a matheuristic solution framework based on a cycle-based formulation that includes column generation, slope-scaling, heuristic and exact optimization techniques. As far as we know, this is the first matheuristic procedure developed for a cycle-based formulation. The column generation solves the linear relaxation model and provides a set of cycles to define the approximation model used in slope-scaling loop. A heuristic is used to convert each solution to the approximation problem into a feasible solution. Memory-based perturbation procedure is used to enhance the performance of the algorithm. Experiments show that the proposed algorithm is able to find good feasible solutions for the problem. The contribution with regard to this study has been accepted for publication in *Transportation Science*.

In the third study, we examine resources allocation issues in service network design. We aim to address a number of fleet utilization issues which usually appear at the beginning of the season because of the change of demand patterns: 1) reposition resources among terminals to account for shifts in demand patterns; 2) acquire (buy or long-term rent) new resources and assign them to terminals; 3) outsource particular services. We present an integrated formulation combining these selection-location and scheduled service design decisions. The mixed-integer formulation is defined over a time-space network, the initial period modeling the location decisions on resource acquisition and positioning, while the decisions on service selection and scheduling, resource assignment and cycling routing, and demand satisfaction being modeled on the rest of the network. We also present a matheuristic solution method combining slope scaling and column generation, discuss its algorithmic performance, and explore the impact of combining the location and design decisions in the context of consolidation carrier service design. This study will be presented at XII International Symposium On Locational Decision, in conjunction with the XXI Meeting of EURO Working Group on Locational Analysis, Naples/Capri (Italy), 2014.

In summary, three studies are considered in this thesis. The first one considers the capaciated multi-commodity fixed cost network design with design-balance constraints, a basic problem in many service network design problems with design-balance constraints. We propose an efficient three-phase matheuristic solution method that includes tabu search, path relinking and exact optimization. In the second study, we propose a new service network design model that takes into account resources limitations at each terminal. We also propose an advanced matheuristic framework solution method based on a cycle-based formulation which includes slope-scaling, column generation, heuristics and exact optimization for this problem. The last study addresses resources allocation issues in service network design. We introduce formulations that model the reposition, acquisition/renting of resources and outsourcing of services. A solution framework based on the slope-scaling approach on cycle-based formulations is proposed. Tests indicate that these proposed algorithms are able to find good feasible solutions for each of these problems.

Keywords: *Freight transportation, service network design, asset management, cycle-based formulations, meta-heuristics, exact algorithm, tabu search, path-relinking, slope-scaling, column generation.*

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Professor Teodor Gabriel Crainic with my sincere and deepest gratitude. Without him, I would not have this very important opportunity to become a PhD student at the Université de Montréal and the Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT). I would like to express my thankfulness towards his valuable and dedicated support throughout my research. His deep instructions and suggestions for my research problems not only help me to understand the significances of my research problems and improve the overall research quality but also show me the way on how I can improve my approach methodology for future research.

I would like to thank my co-supervisor Professor Michel Toulouse for the valuable chance that I could become a PhD student at CIRRELT. His thorough instructions during my study here help me not only understand how to present my research problems; he also shows his kindness during the time doing detail correction for my working documents and thesis. I also always remember the beautiful summers in Montréal which are always nice memories in my life. In addition, I could never forget the days with a lot of fun when I did a scholar visiting position in Oklahoma, US and worked with him during 2013.

It has always been pleasure to collaborate with Professor Mike Hewitt, who currently works at Loyola University Chicago, US. Working with him, I had a good chance to broaden my views and improve my research quality after reading and understanding his deep review and analysis of my research problems. I highly appreciate his time and effort to review my research work as well as his comments and suggestions to improve these works. I also learn a lot about how to write research papers from him, and this would be so valuable for me for my future research.

I sincerely thank the directors and secretaries of CIRRELT and DIRO who are always so willing to guide and help me whenever I need. I would also thank to the technical staffs of

CIRRELT and Calcul Québec for the technical support.

The research could not have been done without the financial support from the Université de Montréal. I would like to thank for their valuable support that allowed me to follow the study at Université de Montréal.

During the time here, I have a chance to meet research colleagues and friends from around the world and from Vietnam. Their help, encouragement, entertainment has made my time spent in Montréal a wonderful experience. I also want to thank Thomas Baker for the beautiful days and nice memories I spent in Oklahoma, US.

I would like to express the gratitude towards the Faculty of Mathematics, Mechanics and Informatics, Hanoi University of Science, and my colleagues for the support during the time I worked at the faculty and I am here.

I cannot express enough gratitude toward my parents and my family who always take care of me and encourage me to study and work in this research field. Their sharing is the basement that makes me feel strong while living away from home.

1. INTRODUCTION

Transportation systems are key actors in the economy. Industries use transportation systems to transport commodities such as raw materials from production sites to factories or finished goods from factories to stores; people use transportation systems for all kind of activities including getting to work, enjoying vacations, shopping or simply visiting friends and family. Efficient transportation systems help to sustain all these activities and the economy while poor transportation systems have the opposite effect. The design of an efficient transportation system means that good decisions have to be made about several aspects of the system such as where to place terminal stations between which commodities are transported, how to transfer commodities, what is the best transportation schedule, where transportation vehicles should be placed at the beginning of a transportation schedule, and many others. Research in transportation, particularly network design and service network design helps to articulate the way these questions can be answered efficiently.

Network design refers to mathematical models and algorithmic solutions that articulate the decision process for the design of physical transportation infrastructures which are needed to transport commodities between different points of origin and destination. *Service network design* refers to a set of decisions concerning the types of services that are used to carry the transportation activities. Service network design is based on solutions to the network design problem, service network design actualizes the transportation activities through decisions regarding the type of service to use, the departure time of each service, the capacity of the service, etc. A flight between two cities or a train trip between two stations at a specific moment are examples of services. A service network consists of all potential services for a specific transportation problem. A service network design provides the selection, routing, scheduling of services, as well as the consolidation of activities in terminals, and the routing of vehicles for each particular demand. The goal is cost efficient operations together with timely and reliable delivery of commodities according to customer specifications and the targets of the carrier.

More recently, service network design has addressed the integration of resource management activities. Resources are generally scarce. Power units (e.g., tractors and locomotives), carrying units (rail cars, trailers, trucks, ships), and crews are examples of resources. Resources are an important component in a service network design as resources are needed to operate services and to route demands for customers. Therefore, service network design problems should consider resources management issues explicitly in the problem description as well as in their solution methods. The lack of research in this field as well as the importance of the resource management in service network design is the motivation of this thesis.

1.1 Motivation and Research Problems

Extensive studies on different aspects and issues in service network design have been carried out so far. However, there are few studies on resource management in service network design. Independent researches have been conducted in regard of particular applications such as ferry transportation (Lai & Lo, 2004), deferred items and ground vehicle transportation (Smilowitz *et al.*, 2003), etc. Recently, Pedersen *et al.* (2009), Andersen & Christiansen (2009); Andersen *et al.* (2009a, 2011) have introduced generic formulations, definitions and notations related to resource management. An example of generic formulations is the capacitated multi-commodity fixed cost service network design problem, mentioned in Pedersen *et al.* (2009). This problem appears in many service network papers considering resource management issues. Because of the broad and important applications of generic formulations, the thesis focuses on this type of formulations.

This thesis proposes three studies. These studies tackle optimization problems that belong to the tactical level of planning in consolidation-based transportation carriers. We first describe the research problem and motivations for each of three studies. Then, we describe our contributions, challenges, and how we solve these challenges in our work.

First study. In this first research work, we consider the capacitated multi-commodity fixed cost network design with design-balance constraints, named DBCMND, as introduced in Pedersen *et al.* (2009). The DBCMND is a basic sub-problem of several service design problems because of the capacitated multi-commodity structure as well as the design-balance property. Indeed, the design-balance property, which requires that the number of resources entering and leaving a terminal must be equal, appears in many problems dealing with resource management. According to Pedersen *et al.* (2009), it is "far from trivial" even to find feasible solutions for the DBCMND because of the design-balance constraints. So far, algorithms to solve design-balance constraints and to obtain feasible solutions were either adhoc techniques such as fixing and rounding variables in a branch-and-bound procedure (e.g. Smilowitz *et al.* (2003)) or neighborhood-based exploration using a tabu search procedure (Pedersen *et al.* (2009)). These methods require a lot of computation to handle design-balance constraints in order to obtain feasible solutions, there are not efficient for large problem instances. In this thesis, we propose a new and efficient solution method to address the DBCMND. This research has been published in the *Journal of Heuristics* with the following reference information: D. M. Vu, T. G. Crainic, M. Toulouse: A three-phase matheuristic for capacitated multi-commodity fixed-cost

network design with design-balance constraints, *J. Heuristics* 19(5): 757-795 (2013)

Second study. In this second research work, we consider limitations on resources at terminals in a problem setting named SNDRC. Existing service network design problems assume usually limitations on resources at the global level, i.e. a maximum number of resources that can be used in the network. However, in practice, a terminal can only have a limited number of resources; further, the number of resources at each terminal usually does not change during the execution of a planning horizon. In this study, we propose a cycle-based mathematical formulation which explicitly accounts for the limited number of resources available at each terminal and which takes advantage of the structure introduced into the model by the resource management constraints. By considering resources bound information, we extends resource management aspects into tactical planning models. Our solution approach combines column generation, a slope-scaling procedure and heuristics. This research has been accepted for publication in *Transportation Science* with following reference information: T. G. Crainic, M. Hewitt, M. Toulouse, D. M. Vu: Service Network Design with Resource Constraints, *Transportation Science* (2013).

Third study. In this work we propose a new service network design model which considers issues related to the allocation of resources such as the reposition of existing resources among terminals to account for shifts in demand patterns, the acquisition/renting of new resources and their assignment to terminals, and the outsourcing of particular services in a problem setting named LWSND. These three fleet utilization issues must be addressed at the beginning of each season. We present an integrated formulation combining the selection-location and scheduled service design decisions. A mixed-integer formulation is defined over a time-space network, the initial period modeling the location decisions on resource acquisition and positioning, while the rest of the network models decisions on service selection and scheduling, resource assignment and cycling routing, and demand satisfaction. Based on our second study, we present a matheuristic solution method combining slope scaling and column generation, discuss its performance, and explore the impact of combining the location and design decisions in the context of consolidation carrier service design. The study will be presented at XII International Symposium On Locational Decision with following reference information: T. G. Crainic, M. Hewitt, M. Toulouse, D. M. Vu: Integration Location - Service Network Design, XIII ISOLDE Symposium - Naples/Capri (Italy) 2014.

1.2 Challenges and Contributions

We had to face several challenges while addressing issues in the three studies. In the first study we had to design a cycle-based neighborhood structure for tabu search that obtains a good compromise between computational speed and the quality of neighborhood moves. We also had to identify an efficient procedure that can handle the design-balance constraints. The state-of-art solution method for the DBCMND still requires a lot of computation effort (many iterations of a add/remove paths procedure) to handle these constraints. We have proposed a minimum cost maximum flow model to quickly obtain feasible solutions that satisfy the design-balance constraints from a solution satisfying the flow constraints. This approach allows to obtain feasible solutions directly from solutions that satisfy the flow constraints in one step. Our contribution in this study is a three-phase matheuristic that combines tabu search, path-relinking and an intensification procedure based on a mixed integer programming solver. Tabu search identifies feasible solutions, path-relinking extends the set of feasible solutions. The solutions found by these two meta-heuristics are used to fix arcs as open or close. An exact solver intensifies the search on a restricted DBCMND problems derived from fixing arcs. The experiments on benchmark instances show that the solution approach finds good solutions to large-scale problems in a reasonable amount of time.

The second study proposes a model named SNDRC which, together with the consideration of the design of the routes to transport a set of commodities by vehicles, takes into account limitations on the number of vehicles at each terminal. A matheuristic solution framework based on cycle-based formulation has been introduced. This matheuristic is composed of a column generation procedure, a slope-scaling procedure and exact optimization techniques. For a large network, it is impossible however to generate all cycles and create the corresponding mathematical model. The first challenge of the this study was to find a way to identify a promising subset of cycles to develop the approximation problem used in the slope-scaling procedure. We found such set of cycles during the solution of a linear relaxation of the model using a column generation procedure. Another issue we had to address in this second study was to develop the approximation problem given that the design-cycle variables do not relate directly to the flow variables. We have decomposed the flows on the arcs into flows on cycles and then develop the approximation model based on new flow variables. The solution to the approximation model, which gives a set of cycles that can deliver flow, provides a (possibly) infeasible solution to the original problem. Unlike other problems from which we can obtain a feasible solution by

rounding variables, in our case, the obtained cycles may violate some set of constraints. A third issue in this study was the development of a heuristic that is able to produce feasible solutions from cycles obtained from the solution to the approximation problem. The heuristic we propose includes two minimum cost maximum flow models and a cycle-extraction procedure; the output is a set of cycles without constraint violations (not including flow constraints). Note, the minimum cost maximum flow model that we have proposed in the first study of this thesis is used in this phase to handle the violations of design-balance constraints and resources bound constraints. An exact solver solves a flow problem to check whether we can obtain feasible solutions. The performance of the algorithm is further improved by the integration of perturbation procedures.

Finally, the third study investigates issues related to resources location in service network design. A number of questions related to the utilization of resources must be addressed at the beginning of each season because of changes in demand patterns compared to the planning of the previous season. We present an integrated formulation, named LWSND, combining the selection-location and schedule service design decisions. A mixed-integer formulation is defined over a time-space network. The proposed model supports the reposition of resources among terminals to account for shifts in demand pattern, the acquisition/renting of resources from external suppliers and the outsourcing of services. Based on our works in the second study, we again develop an approach that includes column generation, slope-scaling and heuristic ideas to solve this new problem. The experiment shows that the algorithm is able to provide good solutions for this new problem in which many new constraints are added. In addition to quantitative comparison, we explore the impact of combining the location and design decisions in the context of consolidation carrier service design.

In terms of contributions, our studies enrich the current research literature on service network design with resource management by, first, proposing an efficient solution method for the capacitated multi-commodity fixed cost network design with design balance constraints (DBCMND), and second, by proposing two new problem settings, named SNRDC and LWSND, with corresponding mathematical models and solution methods. SNDRC addresses the limitations of resources at each terminal while LWSND addresses the reposition of resources, acquisition of capacity and the outsourcing of services. The solution method for DBCMND is efficient not only for the problem itself but also for other problems with design-balance constraints. In particular, the method demonstrates its applications in our studies to SNDRC and LWSND. Regarding SNDRC and LWSND, we are the first to introduce this combination of

slope-scaling, column generation, and heuristics to obtain feasible solutions for cycle-based formulations with complex constraint sets. Our solution methods are efficient for our research problems. In addition, these methods present new techniques to find feasible solutions regarding cycle-based formulations. This will extend the application of our research to other problems of this class.

1.3 Outline of the thesis

The remaining content of the thesis is organized as follow. In Chapter 2, we provide a background on network design problems, particularly, the capacitated multi-commodity fixed cost network design problem, which is the core problem for many service network design problems as well as our research problems. Research on this problem provides us with the basic tools to develop solution methods for our research problems. Chapter 3 provides a literature review on service network design in general and service network design with resource management in particular. In Chapter 4, we describe our first research problem which is the capacitated multicommodity fixed costs service network design with design balance constraints and our contribution to this problem. In Chapter 5, we present a new service network design model with the consideration of resources at terminal level and the corresponding cycle-based formulation. We develop a solution approach which uses slope-scaling to tackle this problem. Chapter 6 considers resource location issues related to repositioning of resources and the acquisition of capacity. We propose a model that allows the consideration of the reposition of resources between terminals, acquisition and renting of resources from external suppliers, and, finally, outsourcing of services. Finally, we provide concluding remarks in Chapter 7, emphasizing again the contributions of the thesis and highlight some items for future research.

Summary of the first part of the thesis

The first part of the thesis provides a background to the area of our research, a literature review pertinent to the problems addressed in the thesis. In Chapter 2, we present key elements of a service network and then we present a generic network design formulation. We focus on a popular setting of the generic formulation, named *capacitated multi-commodity fixed cost network design formulation* which is the sub-structure problem for many problems in service network design. We summarize research works for this problem and highlight which of these works contribute to our research. Next, in Chapter 3, we review papers in service network design, particularly, papers dealing with resources management. We present a classification based on static and dynamic formulations. We discuss briefly about stochastic service network design. Next, we discuss the resources management issues in literature, we present papers dealing with these issues and review the contributions of each paper.

2. BACKGROUND

This thesis focuses on service network design problems with resource management issues. To understand service network design problems with resource management issues, we put these problems in the context of freight transportation carriers.

The first section of this chapter describes briefly freight transportation systems where we distinguish between customized-based transportation service and consolidation-based transportation service. An example of customized transportation services is the taxi services offered by taxi companies. Regular train trips, regular flights are examples of consolidation-based transportation services. Consolidation-based transportation services are usually offered by freight transportation carriers such as rail-based carriers, truck-based carriers, air-based carriers. In this thesis, we focus on consolidation-based transportation carriers.

Next, we present a classification of decision problems for freight transportation systems in terms of three levels of planning: strategic, tactical and operation level. This classification has been introduced in Crainic & Laporte (1997). Service network design problems model network design problems at tactical level. We then describe the basic components of a network design model. From the basic components, we describe a generic network design model, named as the *capacitated mutli-commodity network design*, and the corresponding mathematical formulations. It is noted that many well-known models could be derived from this generic model.

Finally, we focus on a special case of the generic model, namely the *capacitated multi-commodity fixed cost network design model*. This model is the basic structure of many service network design problems. The research on this model provides important contributions to the development of service network design models. We review solution methods developed for this particular problem and we indicate which of these researches are exploited to design solution approaches for our research problems.

2.1 Overview of a Freight Transportation System

Freight transportation, which focuses on the transportation of raw material and goods, supports extensively economic activities such as production, trade, and consumption by ensuring the efficient movement and timely availability of raw materials and finished goods. Freight transportation represents a significant part of the cost of a product, as well as of the national expenditures of any country. Consequently, freight transportation activities should be rational-

ized as much as possible to reduce the footprint of these activities on the cost of goods.

The rationalization of freight transportation takes the form of freight transportation models or systems that are used to determine transportation plans for enterprises that need transportation *services* to transport goods, identified in the subsequent as *demands*. Each service usually consists of one or several vehicles that carry *shipments*, which is a set of one or several demands, from origins to destinations. We can categorize each transportation service based on its nature; this lead to *customized transportation service* and *consolidation transportation service*.

Customized transportation service In customized transportation service, customers can ask for dedicated service. Full truck-load door-to-door services, taxi services are some basic examples. Customized services can be dynamic (like taxi services) or prescheduled according to a long-term contract (like school-bus services).

Consolidation transportation service Contrary to customized transportation, consolidation transportation provides a service for several customers simultaneously by using a same set of vehicles. Consolidation-based transportation services are usually provided by consolidation-based transportation carriers, for example, maritime carriers, railroad carriers, or flight-carriers. *Consolidation* refers to the process of grouping demands from different customers who use the services provided by a consolidation-based transportation carrier. In order to do this, different cargoes and vehicles are grouped, or simply moved from one service to another at the consolidation terminal (e.g. rail classification yard, less-than-truckload breakbulk) in the transportation network. With consolidation-based services, a shipment may pass one or several consolidation terminals before reaching its destination.

Freight transportation company carriers such as rail carriers, long-haul truck-based carriers usually provide consolidation-based transportation services. Consolidation-based transportation services account for a considerable portion of carriers business operating on a large transportation network. The operations of consolidation-based transportation services are complicated by regulatory factors such as rules, policies of carriers and physical factors such as resources, terminals.

The large structure of a consolidation-based transportation carrier creates challenges to decision problems about the structure of its transportation system. Decision problems may involve

many factors such as services, resources, terminals, and many side constraints. If we have a decision problem about the construction of new terminals or the offering of new services, this problem may belong to long-term planning. Whereas, a decision problem that assigns crews to a set of resources (vehicles), will belong to short-term planning. To facilitate the classification of decision problems, we use a classification of planning in three levels.

1. **Strategic** (long-term): decisions at this level determine general development policies and broadly shape the operating strategies of the system including: the design and evolution of the physical network; the acquisition of major resources (e.g. locomotive power units); the definition of broad service and tariff policies.

2. **Tactical** (medium-term): planning here aims to ensure, over a medium term horizon, an efficient and rational allocation of existing resources to improve the performance of the whole system. Typically, tactical decisions concern the design of the service network, i.e. route selection, type of service to operate, operating rules for each terminal and work allocation among terminals, traffic routing using the available services and terminals, repositioning of resources (e.g., empty vehicles) for use in the next planning period (e.g. next week, next month).

3. **Operational** (short-term): planning at this level is performed by local management in a highly dynamic environment where the time factor plays an important role and where detailed representations of vehicles, facilities and activities are essential. Scheduling of services, maintenance activities, routing and dispatching of vehicles and crews, resource allocation are examples of operational decisions.

The classification of decision problems following this three levels of planning allows us to identify to which level an optimization problem belongs. In order to understand the role of the tactical planning level in this classification, we will explain how a decision problem belonging to a level affects problems belonging at adjacent levels. A solution to a strategic problem may define how many terminals we should open and where we should open the terminals, and which services we should offer between opened terminals, etc. Then, based on the network defined by a solution to the strategic problem, a solution to a tactical problem will define resources, services selected for a specific planning horizon such as one month, or one semester. From the solution to the tactical problem, we will assign crews to operate resources so that working rules are enforced for crews to define solutions to operational problems.

In general, the strategic level sets general policies and guidelines for tactical decisions. The tactical level determine goals, rules, and limits for the operational decision level regulating the transportation system. From the outputs, each level supplies information essential for decision-making processes at higher levels. This hierarchical relationships emphasizes the need for formulations that address specific problems at particular levels of decision making. The information from tactical planning activities affects the decisions of both strategic and operational level. It stresses the importance of tactical planning activities and their associated service network design models.

Because of the importance of the tactical level, the thesis focuses on problems in this level. According to Crainic (2000), decisions at the tactical planning address the following issues:

Service Selection Selection of the services to be offered, as well as determination of the characteristics of each service, including the origin and destination terminals, intermediate stops and physical routes. Priority (with frequency) or speed (with schedule) decisions are made in this process.

Traffic Distribution For each commodity, the traffic distribution is the routing specification for the shipments between each origin-destination pair, including the services used, terminals passed through, and operations performed in terminals.

Terminal Policies Terminal policies concern the specification of the activities to be performed in terminals. For example, in rail applications, they indicate how trains entering each yard should be inspected and disassembled, and how cars should be sorted and reassembled into blocks to form new outbound trains.

Empty Balancing Empty balancing refers to the problem of repositioning empty vehicles and other resources so that they can be re-used to satisfy the needs in the next planning period.

The issues considered in the tactical planning are interrelated, and present complex trade-offs. For example, traffic distribution must be honored by the service selection in order to answer the transportation demands; however, the consolidation process which prepares commodities is restricted by terminal equipments and working policies. Empty balancing also

contributes to the traffic flow and should be considered together with the terminal policies in order to avoid terminal congestion.

Service network design is used to model and solve the set of main tactical issues and related decisions for consolidation-based transportation carriers. These issues and decisions include the selection and scheduling of the services to operate, the specification of the terminal operations, and the routing of freight. Service network design models for these decision problems usually take the form of network design formulations; and they are formulated as mixed-integer network optimization problems which are typically NP-Hard problems. To have a general view of these problems, we present the basic components and characteristics of network design problems.

2.2 Basic Components of Network Problems

Generally, a network design problem is defined on a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ in which the set of nodes $\mathcal{N}$ refers to a set of cities or terminals in a transportation network. *Demands*, which are described through a set of *commodities*, are goods, products, or data to be moved between nodes in a network. Each commodity is usually associated with its origin node, its destination node and its volume. The set of links, $\mathcal{A}$, provides representations for the transportation activities between nodes such as the exchange of commodities. Depending on the application, links may be national roads for ground transportation, rail-road for rail transportation, or Internet cables for data communication. Links may have various characteristics such as length, capacity and cost. Length presents the distance of a link or the amount of time to travel on this link. Capacity presents the upper bound of the volume of commodities that can be routed over a link. Cost represents the price for using or adding resources on a link, which is called *fixed cost*, and price for routing commodities on a link, called *routing cost or variable cost or flow cost*.

The objective in network design problems is to select links in a network, along with capacities, eventually, in order to satisfy the demand for transportation at the lowest possible system cost computed as the total fixed cost of the selected links, plus the total variable cost of using the network.

In the following, components and characteristics of network design problems are described

in generic terms and are modeled using mathematical notations. These terms and notations will appear again in other chapters to describe research problems and solution methods.

Commodities The term *commodity* refers to entities that are carried in a transportation network. Example of commodities are passengers in a public transportation system or finished goods transported from factories to distribution centers. Different commodities might refer to different goods, or to a same good with different origins and destinations. For example, shipping coals from an origin to multiple destinations could be seen as sending several commodities from an origin to different destinations. In general, *demands* in a network are treated as multiple commodities, each with an origin-destination pair.

Let k be a commodity and $\mathcal{K}$ the set of all commodities that have to be routed simultaneously over a network. Each commodity $k \in \mathcal{K}$ is associated with an origin $o(k)$ and a destination $d(k)$ as well as its volume w_k. The origin $o(k)$ is where and when commodity k is available and the destination $o(k)$ is the final location of commodity k. A problem with several kinds of commodities is a *multi-commodity network flow problem,* otherwise the problem is classified as a *single commodity network flow problem*.

Links and services We generally use the term *link* to model the transportation activities between different nodes in a network. In a directed network, links are represented by arcs, and in an undirected network, links are represented by edges. In freight transportation, each link represents a *service* that is offered by the carrier. The *capacity* of a link (i, j), denoted as u_{ij}, characterizes the maximum volume of commodities that can be routed over the corresponding arc. If we want to set capacity rules on each individual commodity $k \in \mathcal{K}$, then for each arc (i, j), we will have $|\mathcal{K}|$ links connecting i and j with the corresponding capacity u_{ij}^k for each $k \in \mathcal{K}$.

A network design problem with capacitated constraints on links, either on all the links or a subset of them, is a *capacitated network design problem*. If demands cannot violate capacity constraints, the problem is an *uncapacitated network design problem.*

Costs Costs in the model represent expenditures for the execution of transportation activities. Associated with each unit of commodity k traveling on a link (i, j) is *flow cost*, denoted as

c_{ij}^k. The cost c_{ij}^k is linear in terms of the flow on links, this is the case in multi-commodity network flow problems. In addition to linear cost, we may use non-linear routing costs to depict congestion effects or to model diseconomies other than the fixed cost.

Associated with an installment of a link on a link (i, j) is a *fixed cost*, denoted as f_{ij}. In some situations, installment of a same link multiple times is allowed, e.g., in network communication. If each link is allowed to be installed at most one time, the problem is then classified as a *fixed cost network design problem*.

Static/Dynamic Static network design assumes that demand does not vary during the planning period that is considered. These formulations focus on the selection of service and service frequency, which is the number of times for which a service is installed. Time information is presented implicitly through the definition of services and inter-service operations at terminals. These formulations are usually used to define strategic/tactical problems.

Dynamic or time-dependent network design formulations include an explicit representation of commodities movements in time. The target of a dynamic formulation is a plan of a specified schedule problem. This plan will support decisions related to when services depart. These formulations are used to define tactical/operational problems. To illustrate the movement in time of the system, we use a time-space network by representing the operations of the system over a certain number of time periods. In such structure, the representation of the physical network is replicated in each period.

In dynamic network design, services are defined between terminals in different periods. A service starts from an origin terminal in a given period and then arrives later at another terminal. A service in a static network is duplicated by multiple temporal services links in a time-space network. Links that connect a same terminal with different time-periods represent waiting activities for the flow and resources at terminals. Additional links may be used to capture penalties for arriving too early or too late or to permit the repositioning of vehicles for their next departure. The time-space network increases the size of the network considerably, but it allows an analysis of the dynamic aspects within the planning horizon.

Deterministic/Stochastic Deterministic formulations assume a complete knowledge of the relevant parameters over the planning horizon. To obtain this knowledge, information

about parameters is estimated through either historical data, operating rules (e.g. waiting and handling time in terminals, travel times, operating costs, etc) or forecasting methods.

To handle uncertainties in the change of demand or cost during the planning horizon, stochastic is introduced in network design problems. Stochastic network design problems are built on deterministic ones. These deterministic models are called scenarios; each of them is assigned a probability to reflect the relative importance in an uncertain environment.

This thesis focuses on the deterministic capacitated multi-commodity fixed cost service network design problem because it is a sub-problem in many transportation applications. For particular problems that can be obtained by adding restrictions or relaxations, we find, for example, papers dealing with single-commodity issues (Ortega & Wolsey (2000), Thapalia *et al.* (2012)); unsplittable capacitated network design (Bartolini & Mingozzi (2009), Atamtürk & Rajan (2002)), network loading problems (Avella *et al.* (2007), Berger *et al.* (2000)) or papers dealing with uncapacitated multi-commodity scenario such as Holmberg & Hellstrand (1998); Magnanti *et al.* (1986).

2.3 Network Design Models

In this section, we describe the capacitated multi-commodity network design problem which is an important sub-problem appearing in many problems in the scope of the thesis. Our research problems are also developed on top of this model. For completeness, detailed reviews on network design problems can be found in Magnanti & Wong (1984), Minoux (1989), Ahuja *et al.* (1994), Balakrishnan *et al.* (1997), and Crainic (2000).

We first provide a description of the generic network design problem and mathematical notations. The network design problem consists of finding a minimum cost design, i.e., a selection of arcs in the network to enable the flow of commodities while minimizing the total network cost: the sum of the fixed cost of including the arcs in the final design and the variable cost of routing the commodities.

Formulations for network design problems usually consist of two sets of decision variables. The first set are design variables, these variables indicating how the links are selected and installed. The second set are flow variables, these variables indicating how the commodities

are routed on installed links. For flow information, we can use arc-based flow variables to represent the volume of a commodity on an arc, or we can use path-based variables to represent the volume of a commodity on the path from the origin to the destination of this commodity. These two options for the flow variables express two different formulations for network design models - which are named *arc-design, arc-flow based formulation* and *arc-design, path-flow based formulation*.

Arc-design, arc-flow based model This section describes the generic network design model using arc-design variables and arc-flow variables. Integer design variables y_{ij} represent the number of times a service is installed on link (i, j). Flow variables x_{ij}^k represent the flow distribution within the network and stand for the volume of commodity k moved by service link (i, j). The capacitated multi-commodity fixed cost network design is then:

$$\min z(x, y) = \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k + \sum_{(i,j) \in \mathcal{A}} f_{ij} y_{ij}, \tag{2.1}$$

subject to

$$\sum_{j \in \mathcal{N}_i^+} x_{ij}^k - \sum_{j \in \mathcal{N}_i^-} x_{ji}^k = d_i^k, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2.2}$$

$$\sum_{k \in \mathcal{K}} x_{ij}^k \leq u_{ij} y_{ij}, \quad \forall (i, j) \in \mathcal{A}, \tag{2.3}$$

$$(y, x) \in \mathcal{C} \tag{2.4}$$

$$x_{ij}^k \geq 0, \quad \forall (i, j) \in \mathcal{K}, \forall k \in \mathcal{K}, \tag{2.5}$$

$$y \in \mathcal{Y} \tag{2.6}$$

In this model, $\mathcal{N}_i^+ = \{j \in \mathcal{N} | (i, j) \in \mathcal{A}\}$ and $\mathcal{N}_i^- = \{j \in \mathcal{N} | (j, i) \in \mathcal{A}\}$ define respectively the outward and inward neighbours of node $i \in \mathcal{N}$, while d_i^k stands for the flow value of commodity k at node i, which equals w_k for $i = o(k)$, $-w_k$ when $i = d(k)$, and 0 otherwise.

The objective function (2.1) indicates that the goal of the problem is to minimize the total network cost: the sum of the fixed cost of including the arcs in the final design and the variable cost of routing the commodities. Constraint set (2.2) ensures that each commodity is routed from its origin node to its destination node. Constraint set (2.3) ensures that enough capacity is installed on an arc to carry all flows routed on this arc. Constraint set (2.4) defines constraints

representing additional relations between design variables and flow variables. Constraint set (2.5) and (2.6) define the domain of the variables.

When $\mathcal{Y} = \{0,1\}^{|\mathcal{A}|}$ or $y_{ij} \in \{0,1\}\ \forall (i,j) \in \mathcal{A}$ then y_{ij} = 1 only if link (i,j) is open, or used in the final design; the link is closed when $y_{ij} = 0$. In this case we have a formulation for the *capacitated multi-commodity fixed cost network design problem*.

When $\mathcal{Y} = \mathbb{N}_+^{|\mathcal{A}|}$ or y_{ij} is an integer, then the y_{ij} variables usually represent the number of units of capacity installed, or the level of service offered, and f_{ij} is the cost pay for each unit of capacity installed or service offered on arc (i,j).

In this thesis, we address the case when $\mathcal{Y} = \{0,1\}^{|\mathcal{A}|}$, i.e. the *capacitated multi-commodity fixed cost network design problem* or CMND. Regarding CMND, the objective function (2.1) accounts for the total system cost which includes the fixed cost of selected arcs plus the routing cost of demanded commodities. Constraints (2.2) correspond to flow conversation for each node and each commodity. Constraints (2.3) are often identified as bundle or forcing constraints. These constraints state that the total flow on the link (i,j) can not exceed its capacity u_{ij} if the link is chosen in the final design of the network (i.e. $y_{ij} = 1$) and must be 0 if (i,j) is not part of the selected network (i.e. $y_{ij} = 0$). When the capacity is so large that it is never binding (i.e. $u_{ij} \geq \sum_{k \in \mathcal{K}} w_k$), we have the uncapacitated case, and we can simply ignore (2.3).

Constraints (2.4) may capture additional relations regarding decision variables and/or flow variables. For example, the following budget constraints express the limited nature of available financial resources.

$$\sum_{(i,j) \in \mathcal{A}} f_{ij} y_{ij} \leq B \tag{2.7}$$

These budget constraints illustrate a relatively general class of restrictions imposed upon resources shared by links.

Constraint (2.4) may include partial constraints

$$x_{ij}^k \leq u_{ij}^k, \quad \forall (i,j) \in \mathcal{A}, \quad k \in \mathcal{K} \tag{2.8}$$

which shows restrictions imposed on the use of some facilities by individual commodities, for example the maximum quantity of some goods moved by a vehicle. If $u_{ij}^k = u_{ij}$, these constraints are redundant because of the constraint set (2.3). Despite its redundancy, Gendron & Crainic (1996) indicated that these constraints yield a tighter relaxation formulation and improve the lower bound calculation of network design models.

Arc-design, path-flow based model The multi-commodity capacitated network design problem can be represented via path-based formulations. We denote by $\mathcal{L}^k$ the set of paths from the origin $o(k)$ to the destination $d(k)$ for commodity k. We define the parameter δ_{ij}^{lk}; $\delta_{ij}^{lk} = 1$ if arc (i,j) belongs to path $l \in \mathcal{L}^k$ (0, otherwise). We define c_l^k as the transportation cost of commodity k on path $l \in \mathcal{L}^k$, we have the following relation $c_l^k = \sum_{(i,j)\in\mathcal{A}} c_{ij}^k \delta_{ij}^{lk}$. Define variables h_l^k as the volume of commodity k on path l. We obtain the arc-design path-flow based model:

$$\min z(h,y) = \sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{L}^k} c_l^k h_l^k + \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij}, \tag{2.9}$$

subject to

$$\sum_{l\in\mathcal{L}^k} h_l^k = w_k, \quad \forall k \in \mathcal{K}, \tag{2.10}$$

$$\sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{L}^k} \delta_{ij}^{lk} h_l^k \leq u_{ij} y_{ij}, \quad \forall (i,j) \in \mathcal{A}, \tag{2.11}$$

$$h_l^k \geq 0, \quad \forall l \in \mathcal{L}^k,\ k \in \mathcal{K}, \tag{2.12}$$

$$(y,h) \in \mathcal{C}, \tag{2.13}$$

$$y \in \mathcal{Y} \tag{2.14}$$

Besides CMND, many well-known problems such as Traveling Salesman Problem, Spanning Tree Problem, Vehicle Routing Problem, etc. can be derived from the generic formulations by an appropriate definition of the network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and constraints set (2.4). See Magnanti & Wong (1984) for a summary of these problems and how we can derive these problems from the generic formulations. This illustrates the richness and importance of the network design models and indicates its wide range of real applications. CMND and other problems have been proved to be NP-Hard (Magnanti & Wong, 1984), which suggests that many other problems derived from this problem are also NP-Hard.

From a theoretical point of view, many network design problems are NP-Hard. Consequently, exact solutions are unlikely to solve problems of realistic sizes, leaving heuristic methods to be the best choice to provide solution methods for almost all these problems. Gendron & Crainic (1994) show that for capacitated models, the lower bound obtained via linear relaxation is a poor approximation of the optimal value. It is difficult to model exactly the interplay between link capacities and fixed costs with these relaxations. Moreover, the network flow subproblems are often highly degenerate, meaning the simplex method needs many iterations to solve these flow problems to optimality, when the number of commodities increases. Large problem dimensions are also a challenge. Only for some special cases, e.g. uncapacitated case, have important results been achieved. Contributions are needed for general cases, which usually appear in real applications. Because of the importance of researches on capacitated multi-commodity fixed cost network design problems to our researches, we present in the next section the contributions to this particular model and highlights the applications of these contributions to our research works.

2.4 Solution Approaches to Network Design

Understanding how network design problems are solved, the advantages and disadvantages of each proposed solution method, will contribute to the design of efficient solution methods for our research problems. We classify network design problems and their solution methods into two categories: uncapacitated network design problems and capacitated network design problems. Much research efforts have been done for these two categories; however, the capacitated case is more practical and more difficult than the uncapacitated case. We also show which of these researches contribute to the design of solution methods for our research problems.

2.4.1 Solution Methods for Uncapacitated Network Design Models

Several solution methods have been proposed to solve the uncapacitated fixed cost network design problem and significant results have been obtained. Balakrishnan *et al.* (1989) describe a dual ascent procedure which obtains solutions within 4% of optimality. The proposed algorithm iteratively includes additional arcs in the network by increasing selected arc variable costs. This method is similar to the Lagrangian concept of increasing the multiplier values of

violated constraints to direct the solution towards primal feasibility. Experimental results on problems with 20 to 60 arcs show that the lower bounds obtained are better than those achieved by linear programming relaxations.

Holmberg & Hellstrand (1998) proposed a branch-and-bound algorithm based on a Lagrangian relaxation. At each node of the branch-and-bound tree, they solve a knapsack problem (Gendron & Crainic, 1994) to obtain a lower bound. A feasible solution is generated from the solution of a relaxed problem. Lagrangian dual information obtained during a sub-gradient search is used to fix variables in the branch-and-bound tree. The experimentation is conducted on 115 instances grouped into 13 different classes, however few of these instances have more than 100 commodities. The experimentation shows that the Lagrangian heuristic with branch-and-bound is quite a powerful method, outperforming a state-of-the-art mixed-integer code, with respect to both problem size and solution time.

Magnanti & Wong (1984) applied Benders decomposition to the fixed-charge uncapacitated network design problem. The authors proposed a variable elimination pre-processing procedure to solve undirected uncapacitated network problems to optimality up to 30 nodes and 90 edges.

Sun *et al.* (1998) developed a tabu search heuristic using a simplex method on a graph as a local search. In addition to the uncapacitated assumption, this problem assumes there is a direct route between a pair of origin and destination for a commodity. The procedure is guided by several types of memories, including an immediate memory to find the local approximated optima, a short-term memory to intensify the search, and a long-term memory to diversify the search among the least visited arcs. The algorithm is able to solve problems with 50 origins and 100 destinations.

In general, for uncapacitated models, efficient solution methods exist partially because of the good characteristic of the linear programming relaxation of uncapacitated network models. However, in most problems, the total volume of demands shipped usually violates capacity of any link, yielding network design problems that belong to the capacitated case. This leads to the development of solution methods for the capacitated models.

2.4.2 Solution Methods for Capacitated Network Design Models

Much effort has been dedicated towards capacitated problems on general networks which are more difficult to solve and which pose considerable algorithmic challenges. The development of solution methods for capacitated problems requires that these methods can efficiently provide solutions with good bounds and can perform on large-scale problem instances with a large number of commodities.

We classify solution methods for capacitated models into two categories: exact methods and meta-heuristic methods. Exact methods use mathematical tools such as linear programming and integer programming to develop solution methods. One advantage of exact method is that optimality of solutions can be proved. Meta-heuristics are strategies to guide the search process which uses low-level heuristics to find solutions. Meta-heuristics do not guarantee to find global optimal solutions; however, they can often find good solutions with less computational effort than exact methods, or simple heuristics.

Exact methods Exact methods use mathematical results from graph theory, linear programming, and mixed integer programming to study and exploit the mathematical structure and characteristics of each problem. Wolsey (1998) is a good source on notations and theory related to linear programming and mixed integer programming; one can find definitions and discussions related to linear relaxations, lower bound computation, valid inequalities, cutting plane algorithm.

Gendron & Crainic (1994, 1996) examine the characteristics of lower bounds for the CMND through several relaxation approaches. The structure of the capacitated multi-commodity network flow problem allows to decompose the CMND into several easier sub-problems such as the shortest path problem and the knapsack problem (through Lagrangian relaxations), which can be solved efficiently by inspection or using classical algorithms. Crainic *et al.* (2001) exploit these Lagrangian relaxations to derive bundle-based relaxation solution methods to solve the CMND. The study offers insights into the behavior of each particular relaxation method and proposes strategies for their design and implementation. These experiments also demonstrate the advantages of bundle methods compared to sub-gradient approaches, e.g. their converge faster toward the optimal value of the Lagrangian dual. Frangioni, A. & Gendron, B. (2009) further contribute to lower bound computations by introducing 0-1 formulations of the prob-

lem; the authors develop a column-and-row generation algorithm to compute lower bounds efficiently. The authors show that the Lagrangian relaxation can be solved easily by solving a set of linear relaxations of knapsack problems and restricted master problem by using techniques such as those in bundle-type algorithms (e.g. Crainic *et al.* (2001)).

Holmberg & Yuan (2000) have proposed a branch-and-bound algorithm based on a Lagrangian relaxation for the CMND. Similar to the uncapacitated case, at each node of the branch-and-bound tree, they solve a knapsack relaxation problem (described in Gendron & Crainic (1994)) to obtain a lower bound. Feasible solutions are generated from the solutions of the relaxed problem. The Lagrangian dual information obtained during a sub-gradient search is used to fix variables in the branch-and-bound tree. The computational results suggest that the proposed method provides quite an efficient way to obtain near optimal solutions with a small computational effort, especially for large-scale problem. Their method is better and/or faster in 52/65 cases, compared to CPLEX.

We can find studies regarding mathematical characteristics such as valid inequalities and polyhedra structure of the CMND in Atamtürk & Rajan (2002); Barahona (1996); Atamtürk (2000) and others. Based on valid inequalities, Chouman *et al.* (2003, 2009) propose a cutting plane solution method to solve the CMND. The authors present separation and lifting procedures for each class of valid inequalities used in the algorithm. In order to solve the separation easily, the authors propose to use simpler versions of flow cover and flow pack inequalities instead of solving the original ones. The simpler versions involve only one arc for each inequality while the original ones involve multiple arcs which belong to a cut-set. Experimentations show that these valid inequalities are able to produce better lower bounds. These experimentations also show that the cutting plane algorithm can be included in a branch-and-cut algorithm because the cuts it generates are valid at all nodes of the enumeration tree.

Benders decomposition (Benders, 1962) is applied to solve fixed cost network design problems by exploiting the multi-commodity network structure. Costa (2005) presents a survey on Benders decomposition for network design problems. Magnanti *et al.* (1986) propose a methodology to improve the performance of Benders decomposition by using Pareto-optimal cuts. Sridhar & Park (2000) propose a Benders decomposition approach for the fixed cost capacitated network design problem. Rei *et al.* (2009) accelerate Benders decomposition by using a local branching technique that was introduced by Fischetti & Lodi (2003) and that was applied to a multicommodity capacitated fixed-charge network design problem. Numerical tests show that local branching techniques can improve the performance of standard Benders

algorithms for mixed integer programming problems.

Because of the importance of the capacitated multi-commodity network design model, the contributions of exact methods to this problem (e.g. valid inequalities such as flow covers) have been integrated into commercial solvers such as CPLEX to speed up the computation process and improve solution quality.

Heuristic methods Because the CMND problem is NP-Hard (Magnanti & Wong, 1984), almost all efficient solution methods for this problem are based on heuristic or meta-heuristic methods. Review on meta-heuristics can be found in Glover & Laguna (1997) for tabu search; Resende *et al.* (2010) for path-relinking and scatter search; and Crainic *et al.* (2005); Crainic & Nourredine (2005); Crainic & Toulouse (2003, 2010, 1998); Crainic (2005) for the parallel meta-heuristics.

Ghamlouche *et al.* (2003) proposed one of the most important tabu search solution method for the CMND. The key element of this heuristic is a neighborhood structure, named *cycle-based neighborhood*, which allows changing the flow pattern of several commodities simultaneously, as well as opening and closing several arcs. The experimentation demonstrates that this is the best approximate solution method for the CMND in terms of robust performance, solution quality, and computing efficiency. The idea of the cycle-based neighborhood structure has been reused to develop many other neighborhood structures (e.g. in (Pedersen *et al.*, 2009; Chouman & Crainic, 2010). This shows the importance of this neighborhood structure in the development of heuristics for the capacitated multi-commodity network design in general. The first study in this thesis also presents a variant of the cycle-based neighborhood structure with the goal of obtaining a good tradeoff between computation time and quality of the neighborhood move.

Extending the work in Ghamlouche *et al.* (2003), Ghamlouche *et al.* (2004) propose a new path-relinking solution method which uses the neighborhood structure of the tabu search in Ghamlouche *et al.* (2003). This paper proposed six ways of building reference sets and six ways of selecting initial and guiding solutions. This work reinforces the guidelines relative to how one can build the reference set and select the solutions to be linked. The authors also explain how to use path relinking to implement both the search intensification and the diversification phases. The experimentation shows that path-relinking provides better results than tabu search; and it highlights the application of path-relinking to network design problems. The first study

in our thesis also inherits from this contribution to develop a path-relinking procedure which is a component of the solution method framework in our first research problem.

Crainic & Gendreau (2007) propose a scatter search algorithm for the CMND. Scatter search obtains new solutions from a set of candidate solutions. Like path-relinking, scatter search creates new solutions by using existing solutions in a candidate set. The difference is that path-relinking creates new solutions based on a pair of solutions while scatter search creates new solutions from more than two solutions. A new design, which is a set of open links, is obtained by a linear weighted combination of existing designs in the candidate set. Then, flow information is obtained by solving a multi-commodity network flow problem defined on open links. The cycle-based tabu search algorithm of Ghamlouche *et al.* (2003) is used to get improved solutions. Extensive computational experiments performed on a fairly large set of benchmark problems have shown that, on average, the most effective variants of the scatter search heuristic do not perform better than the best existing method in Ghamlouche *et al.* (2004), but they give close results.

Crainic *et al.* (2006) propose the first multi-level cooperative search for the fixed cost network design. Multi-level cooperative search is a parallel meta-heuristics built upon independent search threads that exchange information between levels. Each level is defined by a restricted fixed cost network design problem. A restricted problem is a problem obtained from the fixed cost network design problem by fixing some links to be open or close. The multilevel cooperative method obtained consistently higher-quality solutions, for 93% of the problem instances tested, for an average improvement gap of 2.38% compared to sequential cycle-based tabu search. The objective function values of the solutions identified by the multi-level method are often equal or better than those obtained by the path relinking (Ghamlouche *et al.*, 2004), for an overall average gap improvement of 1%. In addition to a new algorithm contribution, the paper present guidelines regarding the definition of levels and the information exchanged between levels for network design problems.

Crainic *et al.* (2000) proposed a simplex-based tabu search method for the CMND based on a path-based model. They use ideas from the simplex algorithm, i.e. moves from one base to another base by adding/removing paths. A column generation method is used to generate paths. The results show that the simplex-based tabu search identifies good solutions within reasonable computing efforts and it outperforms relaxation-based heuristics in terms of solution quality. The relative optimality gaps are also low. This approach shows how one can exploit the generic and popular solution method for linear programming to solve mixed-integer

programming problems.

Crainic & Gendreau (2002) extend the works of Crainic *et al.* (2000) by embedding the sequential tabu search into a parallel cooperative framework. The cooperation aspect of the parallel scheme is achieved through asynchronous exchanges of information. Information is shared through a central memory or pool of solutions. They develop five selection strategies corresponding to different ways to access the pool of data and to extract the information requested by a search thread. They also develop six import criteria to determine when a search tread looks for, and eventually accepts, an external solution. These strategies could be used as a guideline for selection strategies and import criteria for the cooperation mechanism.

Hewitt *et al.* (2010) present a solution approach that combines mathematical programming algorithms with heuristic search techniques. The method generates together the upper bound and lower bound at each iteration while using arc-based and path-based formulations to guide the search. This work includes attractive ideas such as guidelines for arc selection and arc fixing with different criteria; and after fixing variables, it allows to use solvers such as CPLEX to find good feasible solutions as well as to obtain lower bounds more rapidly than existing approaches.

Kim & Pardalos (1999) first introduce the application of the dynamic slope-scaling method to solve fixed cost network design problems. Slope-scaling is a technique to approximate a mixed integer program by a linear program. The original problem is approximated by a capacitated multi-commodity flow problem characterized by its objective function. A solution to the approximation problem provides a solution to the original problem by opening all links that carry flows. The approximation problem is also updated by using its solution to adjust the objective function, and this process is repeated until some stopping conditions such as the number of iterations or when the solutions of two consecutive approximations are the same. This work defined a new approach for the design of solution methods for network design problems.

Crainic *et al.* (2004) have enhanced the slope-scaling approach for CMND by integrating Lagrangian relaxation and long-term memory. Lagrangian relaxation information is used to diversify the search. Long-term memory is used to gather useful information and help intensify the search into more promising regions. These results emphasize that not only the combination of slope-scaling, Lagrangian relaxation and long-term memory enhances the performance of the slope-scaling for CMND, but also that it might become the algorithm of choice for more complex problems if it is able to convert a solution to the approximation problem into a solution

to the original problem. Zhu *et al.* (2011) demonstrated an application of slope-scaling to solve a complex and large rail freight and operational planning problem. In our thesis, we show how we can apply slope-scaling to solve complex cycle-based service network design formulations. We describe a technique where we approximate a cycle-based formulation by a flow formulation and obtain active cycles from flow information. We also introduce heuristic techniques that convert infeasible solutions into feasible solutions of a complex service network design problem.

2.5 Conclusion

This chapter has introduced background on network design problems. We have provided some common notations used in mathematical formulations of such problems. We have described in detail the capacitated multi-commodity fixed cost network design problem because this problem serves as a basic problem for many other problems. We have described the solution methods proposed for this problem and we have highlighted the contributions of these researches to our work. Specifically, tabu search, cycle-based neighborhood structure, path-relinking, column generation and slope-scaling methodologies will contribute to the development of our solution methods.

3. LITERATURE REVIEW

This chapter reviews the literature on service network design problems, particularly, problems with resource management. We first present a overview of service network design and then introduce service network design with resource management. A review of papers with resources management constraints is presented to summarize the contributions of these papers.

3.1 Service Network Design

Service network design addresses issues related to planning, selection, and scheduling of services in transportation systems. Service network design formulations are used to build a transportation plan to ensure that the system operates efficiently, serves demand, and ensures the profitability of the firm. The physical infrastructure and the available resources are fixed for these problems. The goal is cost efficient operations together with timely and reliable delivery of demand according to customer specifications and the targets of the carrier. There is quite a significant body of literature on the subject: Christiansen *et al.* (2004), Christiansen *et al.* (2007) for maritime transportation, Crainic (2000) for service network design in freight transportation, Crainic (2003) for long-haul transportation, Cordeau *et al.* (1998) for rail transportation, Crainic & Kim (2007), Crainic & Bektas (2008) for inter-modal transportation.

Deterministic service network design formulations belong to one of the two following categories: *service network design with frequency* and *service network design with schedule*. We describe the formulations into these two categories. Note that all formulations in these two categories address deterministic, fixed cost, capacitated and multi-commodity network design problems.

Service network design with frequency refers to problems that determine the number of times each service is performed. Service network design with frequency may be furthered classified into *decision* or *output* problems according to how the formulation generates the frequency for each service. Output-frequency service network design include fixed cost network design formulations. In these formulations, 0-1 variables indicate whether a service is operated or not, and then the frequency is determined from flow information, i.e. by dividing the flow by the capacity of a vehicle. Powell & Sheffi (1983); Powell (1986) provide examples of output-frequency service network designs for a Less-Than-TruckLoad motor carrier. Solutions give the number of trucks needed to carry the flow on each selected service. Powell & Sheffi (1983) present a service network design model to solve the load plan problem for a motor

carrier. The problem is modeled as a large-scale mixed integer problem and heuristics are developed to determine how to route and combine flows of shipments over the network. Powell (1986) improved the solution methods by local improvement heuristics which add and drop services from the network and reroute the flow to determine the effect on the total system costs. Then, an interactive optimization system that solved service network design models using these heuristics was developed for a motor carrier (Powell & Sheffi, 1989). Crainic *et al.* (1984) addresses a decision-frequency service network design problem. In this work, the frequency of each service is an integer number which determines how many times a services is operated. Other examples of service network design with frequency are the works of Kim *et al.* (1999); Barnhart *et al.* (2002); Barnhart & Shen (2005) which focus on express shipment delivery. Kim *et al.* (1999) proposed a general modeling and solution approach for large-scale multimodal express package service network design problems. Barnhart *et al.* (2002) improved the model and solution approach presented in Kim *et al.* (1999) by introducing new valid inequalities and heuristics. Barnhart & Shen (2005) again improved the existing works by developing model that integrates next-day services and second-day services.

Regarding service network design with schedule or dynamic service network design, time-dimension is integrated into the formulations present planning schedules. The time information supports decisions related to issues such as whether a service or not and when the service departs if selected. Solutions to service network design with schedule define not only the services offer but also the schedule of the services. The following papers address service network design with schedule: Haghani (1989); Zhu *et al.* (2011); Andersen *et al.* (2011) for rail-based transportation, Favolden & Powell (1994) for less-than-truckload transportation, Lai & Lo (2004) for ferry transportation. In Haghani (1989), the authors proposed a nonlinear mixed-integer formulation over a time-space network to describe the railway operations. They proposed a solution approach in which the original problem is decomposed into many smaller subproblems. These subproblems could be solved more easily and then they combine the solutions to these subproblems to obtain solution to the original problem. Favolden & Powell (1994) presented a mixed integer programming formulation for the less-than-truckload transportation in which they minimize the total shipment routing and travel costs of tractors while taking into account the penalty cost for late/early delivery. The authors proposed a primal-partitioning algorithm where a column generation algorithm is used to solve the freight routing problem. In this algorithm, service configuration is explored using an add/delete heuristic. These papers are described in more details in the next section.

Lastly, we take a look at stochastic service network design. Deterministic service network design assumes full information about demand in the future. However, in most cases, one does not know what the demand will be once the plan is executed. Consequently, ad-hoc changes with high cost need to be done to adapt to the actual demands. Lium *et al.* (2007), Lium *et al.* (2009) investigate the importance of introducing demand stochasticity in the form of a scenario tree into service network design formulations. Based on experiments using small instances, they show how solutions based on uncertain demand differ from solutions based on deterministic demand and provide qualitative descriptions of the structural differences. They also point out some characteristics relating to paths/services that carry flows which are shared by scenarios. Hoff *et al.* (2010) extend the formulations in Lium *et al.* (2007), Lium *et al.* (2009) and propose a first meta-heuristic approach which includes exact and heuristic methods to solve real-life instances. Bai *et al.* (2014) extends the work of Lium *et al.* (2009) by introducing rerouting as a second meaning to increase flexibility in addition to outsourcing which was presented in Lium *et al.* (2009). The lack of work in this field reflects the needs for efforts to fill the gaps in stochastic service network design.

3.2 Resource Management in Service Network Design

Now, we focus on the resource management in service network problems which is the subject of the thesis. Resource is a generic term which may refer to trucks, vehicles or crews which are part of service network design problems. When resource management issues are not jointly addressed with the service network design problem, those issues are left to the operational level of planning on an already-decided service network schedule. With the increasing pressure on carriers to decrease costs and improve service quality, combined with the growth in demand for transportation, resource management issues must be solved together with the design and schedule of the service network. This trend appears explicitly in applications where resources are limited and are high-valued such as in freight transportation or in air transportation.

Despite its importance, resource management have not been widely considered in service network design problems. We only see these considerations in the most basic form in some specific applications, e.g., ground vehicle transportation, Smilowitz *et al.* (2003), ferry passengers application, Lai & Lo (2004), and express air courier transportation, Kim *et al.* (1999). Starting from basic issues such as the number of resources, more practical and complicated issues such as design-balance constraints, route-length constraints, service-frequency constraints, etc. are

integrated into service network design models.

The *full asset-utilization operation policy*, Crainic & Kim (2007), represents an usual situation in service network design models that relates to services and resources. A full-asset-utilization operation policy generally corresponds to the operation of regular and cyclic scheduled services with fixed composition. Crainic & Bektas (2008) describe an application of full-asset-utilization operation in rail-transportation: "Given a specific frequency (daily or every x days), each service occurrence operates a train of the same capacity (length, number of cars, tonnage) and the same number and definition, i.e., origin, destination, and length, of blocks (groups of cars traveling together as a unit from the origin to the destination of the block; blocks result from classification operations at yards). Assets, engines, rail cars, and even crews assigned to a system based on full asset-utilization operation policies can then continuously follow circular routes and schedules". This shows an example of the relationship between services and resources used to operate services.

To model the policy in service network design, many constraint sets are proposed. One of the most basic and important constraint is the design-balance requirement. This constraint ensures, for example, that the number of resources entering and leaving a terminal must be equal. This constraint appears in vehicle routing (Smilowitz *et al.*, 2003) or ferry transportation (Lai & Lo, 2004). This constraint enables "the circular routes and schedules" which is part of a full asset-utilization policy. Service network design models have been extended to capture more complicated resource management issues. The route-length limits on the duration of resource routes reflect practice in several transportation firms. For example, we may set an appropriate value to the route-length constraint to support hours-of-services rules (zHO, 2013) in long-haul transportation. Other resource management constraints such as the coordination of fleet in an inter-modal system (Andersen *et al.*, 2009b) or the service/resource frequency (Andersen *et al.*, 2009a, 2011) are considered. The coordination of multiple fleets refers to the management of many vehicles with different characteristics so that these vehicles could operate efficiently together to transport shipments from their sources to their destinations. The consideration of service/resource frequency aims to obtain a tradeoff between running with free space in low-peak periods and not serving all the potential demand in high-peak periods. The following resource constraints have been used to enforce resource management rules and full asset-utilization operation policy in service network design.

Design balance constraints These constraints come from the full-asset-utilization policy, and state that the number of services entering a node is equal to the number of services departing from this node. In general, we assume that each service is operated by at most one resource, this leads to the balance of resources at each node.

Fleet size The fleet size restricts the number of vehicles available during the planning horizon.

Fleet types The resources of a fleet may be assumed to be *homogeneous* , i.e., every resource of this fleet has same characteristics. *Heterogeneous* fleets refer to the case in which several types of resources are considered, i.e., they are different in their capacity, cost.

Service frequency The service frequency determines how often a service is performed during the planning horizon. A limit on the number of occurrences of a given service is imposed, e.g. "this service should be operated at least three times and at most six times a week", to balance between the economic goal and the quality of service of the carriers.

Route length requirements The route-length requirement limits the length of each route which, for example, helps to strengthen operation rules of carriers. It also enables the cyclic execution of resources over a planning horizon and presents the repetition of the schedule.

Because of the design-balance constraints, in addition to the classic arc-based formulations, service network design with resource management could also be described through cycle-based formulations. These cycle-based formulations have been shown to be efficient and to have more advantages than arc-based frormulations (Andersen *et al.*, 2009a). We use cycle-based formulations to develop models and solution methods for our research problems.

Table 3.1 lists papers that integrate resource management constraints with the corresponding application domain and the solution methods. We will classify and discuss in more detail these papers to have an overview of how the resource management issues have been covered in the literature.

Reference	Application	Solution method
Andersen & Christiansen (2009)	Rail	analysis-CPLEX
Andersen *et al.* (2009b)	Generic	analysis-CPLEX
Andersen *et al.* (2011)	Generic	branch-and-price, cycle-based formulation
Agarwal & Ergun (2008)	Maritime	cycle-based formulations, exact method (column generation + bender decomposition)
Barnhart & Schneur (1996)	Air	path-formulation + column generation
Büdenbender *et al.* (2000)	Air	greedy+tabu/branch&bound
Kim *et al.* (1999)	Air	tree formulation, column generation
Lai & Lo (2004)	Maritime	two-phase heuristics
Pedersen *et al.* (2009)	Generic	tabu-search
Smilowitz *et al.* (2003)	Vehicle	obtain lower bound CG with by path-based formulation; obtain upper bound with heuristics
Tang *et al.* (2008)	Air	Lagrangian relaxation along with a sub-gradient method
Teypaz *et al.* (2010)	Generic	heuristics, decomposition method
Yan & Chen (2002)	Intercity bus	Lagrangian relaxation along with a sub-gradient method
Yan & Tseng (2002)	Air	heuristics with problems assumptions
Wang & Lo (2008)	Maritime	relaxation, decomposition and line search

Table 3.1: Application domain and solution approach of each reference.

3.2.1 Overview and Classification

We first classify these papers by the application domain. Next, we classify them by the characteristics of the models, type of formulations (arc-based, cycle-based, etc). Finally, we identify how resource management constraints are enforced in the papers.

Application domain Many research papers focus on a specific domain such as application in maritime, in air transportation, or in ground transportation. Papers focusing on a specific domain exploit specific information from problems in that domain. The specific information from each application facilitate the design of solution methods and improve solution quality. For example, papers in airline network design may assume that each aircraft (resource) stops at most three times from its origin to its destinations. However, the use of specific information limits the applicability of the solution methods to other application domains, or even other problems in a same domain.

- Ground transportation: Smilowitz *et al.* (2003) - deferred item and vehicle routing , Kim *et al.* (1999) - multimodel express package delivery, Yan & Chen (2002) - inter-city bus

carriers.

- Rail transportation: Andersen & Christiansen (2009) - European rail freight, Andersen *et al.* (2009b) - Polcoridor project.

- Maritime transportation: Agarwal & Ergun (2008) - ship schedule in liner shipping, Lai & Lo (2004), Wang & Lo (2008) - ferry fleet service network design,

- Air transportation: Barnhart *et al.* (2002) - express shipment delivery, Yan & Young (1996), Yan & Tseng (2002) - airline schedule and fleet routing, Büdenbender *et al.* (2000) - direct flight network design, Tang *et al.* (2008) - passenger and cargo routing, Lin (2003) - direct cargo routing.

- Generic application: Pedersen *et al.* (2009), Andersen *et al.* (2009a), Andersen *et al.* (2011).

Formulations and solution methods Most papers propose mixed integer programming models with the goal of minimizing the total fixed cost of services, resources and routing cost, or maximizing the net profit by subtracting the sum of operating costs from the revenue generated (e.g. Agarwal & Ergun (2008), Teypaz *et al.* (2010)). Some papers use non-linear models to capture waiting time, delay time factors, e.g. Andersen & Christiansen (2009). Dynamic formulations using a time-space service network are more popular than static formulations. This because the papers that propose time-space network formulations usually model optimization problems which relates to the selection and departure of services; so the time dimension is required in these formulations.

- Linear model : Smilowitz *et al.* (2003), Kim *et al.* (1999), Yan & Chen (2002), Agarwal & Ergun (2008), Lai & Lo (2004), Barnhart *et al.* (2002), Yan & Young (1996), Yan & Tseng (2002), Büdenbender *et al.* (2000), Pedersen *et al.* (2009), Andersen *et al.* (2011).

- Non-linear model: Wang & Lo (2008), Andersen & Christiansen (2009).

We also notice that arc-based formulations, path-based formulations, cycle-based formulations, and even tree-based formulations are proposed. Path-based and cycle-based formulations are usually solved using methods such as column generation and branch-and-bound methods, while arc-based formulations are usually solved using meta-heuristics, heuristics,

Formulations	References
Arc-based	Others
Path-based	Barnhart & Schneur (1996)
Cycle-based	Agarwal & Ergun (2008), Andersen *et al.* (2009a), Andersen *et al.* (2011)
Tree-based	Kim *et al.* (1999)

Table 3.2: Types of formulations.

and Lagrangian relaxation & sub-gradient methods. The tree-based formulation in (Kim *et al.*, 1999) is developed based on an assumption on routing cost, which allows to reduce the size of the formulation significantly. Table 3.2 summarizes the papers following the types of formulations.

Resource management issues We now analyze how, in these papers, resource management issues are jointly considered with service network design. Table 3.3 summarizes the set of resource management constraints that are considered in each paper.

Design-balance constraint is included in all these papers. Generally, this constraint is enforced at every node of the network to represent the movement, reposition or empty balance of resources. Some papers, such as Lai & Lo (2004); Wang & Lo (2008), which deal with ferry service network design, do not enforce this constraint at the last period of the planning horizon. This is justified by the fact that the cost of ferry reposition at the end of the day is relatively insignificant compared to the total operation cost.

Fleet type and fleet size constraints are considered in most of these papers. Papers dealing with air and maritime service network design usually assume several fleet types in the model because air-crafts and ferries usually do not have the same characteristics (capacity, speed, etc). Service network design problems for ground transportation company such as long-haul truck carrier may assume homogeneous resources; this assumption appears in Smilowitz *et al.* (2003) or Andersen *et al.* (2011).

Service frequency constraint enforces bounds on the number of times a service is operated. In Andersen *et al.* (2011), this constraint is used to limit the minimum and the maximum number of times that a service is executed over the planning horizon. The goal of this constraint is to balance between the profit and expense of the operation during low-peak and high-peak periods of a planning horizon.

Reference	DB	FS	FT	SF	RL
Andersen & Christiansen (2009)	x	x	x		
Andersen *et al.* (2009b)	x	x	x	x	x
Andersen *et al.* (2011)	x	x	x	x	x
Agarwal & Ergun (2008)	x	x	x		x
Barnhart & Schneur (1996)	x	x	x		
Büdenbender *et al.* (2000)	x	x	x		
Kim *et al.* (1999)	x	x	x		
Lai & Lo (2004)	x	x	x		x
Pedersen *et al.* (2009)	x				
Smilowitz *et al.* (2003)	x		x		
Tang *et al.* (2008)	x	x	x		
Teypaz *et al.* (2010)	x		x		x
Yan & Chen (2002)	x	x			
Yan & Tseng (2002)	x	x	x		
Wang & Lo (2008)	x	x	x		

Notation	Explain
DB	Design-balance
FS	Fleet size
FT	Fleet type
SF	Service frequency
RL	Route-length

Table 3.3: Summary of resource management constraints.

The route-length constraint can be used to enforce the work rules of the carrier such as the maximum driving time for long-haul driver or to enforce the cyclic execution of resources by setting a value which equals to the length of the planning horizon (Andersen *et al.*, 2009a, 2011). In Teypaz *et al.* (2010), the author set the total operation time of a resource during a planning horizon in a predefined interval to ensure that a resource is not over-used or under-used.

3.2.2 Detailed review

We now describe in more details each paper that has been referenced in the previous section. Kim *et al.* (1999) develop models and algorithms for a large-scale transportation service network design problem and illustrate an application in the express package delivery industry. The problem assumes that the routing costs do not vary by commodity. By exploiting this assumption, the authors propose a tree-based formulation for the problem and obtain a dramatic decrease in problem size without compromising the correctness of the model. The solution approach includes the solution to an approximate problem which includes only design variables. Then a restricted problem defined on the optimal solution of the approximated problem is solved to identify the solution to the original problem. The experiment on real data sets

indicates that the new plan can save 10% of cost, reduce 12% of the number of aircraft miles flown, and uses about 10% fewer aircraft.

Barnhart *et al.* (2002) propose a static network service design model for express shipment delivery where a fleet of aircraft need to be balanced need to be balanced. The number of flights of each type is limited. They propose a column-generation based solution method in which new routes for flights are generated. The shipments flow information is obtained through a specialized integer multi-commodity flow model with side constraints, which is a location elimination problem represented through a tree-based formulation. The solution found by the algorithm provides a better plan in terms of the number of aircraft, costs, etc. compared to the one provided by the carrier.

Yan & Chen (2002) propose a time-space service network design model for inter-city bus carriers. A model and a solution algorithm have been developed for inter-city bus carriers to efficiently and effectively design their bus routes and schedules, so as to improve their profits and levels of service. The scope of the study is confined to the subject of pure bus routing and scheduling operations for a given passenger demand, bus fleet size and the related cost. The authors propose an algorithm based on Lagrangian relaxation, a sub-gradient method, and heuristics. The experiment is performed on data set from a major Taiwan inter-city bus carrier operations in 1998 which includes 5 cities, 320 buses. The model includes 15,120 nodes, 85,680 arcs, and 18,001 constraints. The results show that the proposed algorithm provides quite good solutions with 2% of optimality gap. Sensitive analysis is also performed to understand the sensitivity of the algorithm to changes on parameters.

Yan & Tseng (2002) study an airline flight scheduling and fleet routing problem. The scope is confined to the subject of fleet routing and flight scheduling operations under a given trip demand, multiple aircraft types, fleet size, available airport capacity and related cost data. A set of solution techniques such as Lagrangian relaxation, a sub-gradient method, and heuristics is proposed to derive a solution method for this problem. The data set comes from a major Taiwan airline's domestic operations in 1996 which includes 11 cities, 170 daily flights, with three types of aircraft categorized into two fleets. The algorithm is able to provide solutions with 2.5% optimality gap. Sensitive analysis is also performed to examine the performance of the algorithm regarding changes on parameters.

Smilowitz *et al.* (2003) study the routing of deferred items using ground vehicles and aircraft. The ground vehicles are assumed to be balanced at terminals. The authors propose a

two phase solution method in which the first phase estimates lower bounds using an exact solution method, and the second phase tries to obtain a feasible solution using four different adhoc rounding techniques. The proposed algorithm provides quite good solutions for small-size tested instances; however large optimality gaps are observed for large-size tested instances .

Lai & Lo (2004) study a ferry network design problem by considering the optimal fleet size, routing, and scheduling for both direct and multi-stop services. A two-phase path-based heuristic algorithm is developed. Phase I determines a set of paths (ferry routes and schedules) that gives a feasible and good upper bounds on the optimal solution. Phase II tries to search for solution improvements using the phase I path set as a base. Two scenarios of ferry services in Hong Kong are solved to demonstrate the performance of the heuristic algorithm. The data set, which includes two scenarios, is quite small. The first scenario has about 100 integer variables while the second has about 800 variables. The results of the numerical studies showed that the heuristic produces solutions that are within 1.3% from the CPLEX optimal solutions with only few seconds computation time.

Wang & Lo (2008) study a multi-fleet ferry service network design that takes into account ferry services with different operation characteristics and passengers with different preferred arrival time windows. The full problem is formulated as a mixed integer nonlinear programming problem and solved with a heuristic procedure that first fixes the demand and then decomposes the resultant model by ferry services. The same instances in Lai & Lo (2004) are used for the experiments. The results shows that the proposed algorithm produces good solutions with 5.9% gap, as compared with the lower bound solutions. The ferry decomposition scheme is also useful to reduce the problem size and computation time substantially, about five times.

Agarwal & Ergun (2008) study a service network design for a liner shipping company. Liner shipping mainly involves carrying containerized cargo on regularly scheduled service routes. The authors propose a cycle-based formulation that exploits the design-balance property to represent route of ships and cyclic schedules. The problem is proved to be NP-Complete. The authors develop three different solution methods which include a greedy heuristic, a column generation algorithm and a two phase Benders decomposition algorithm. As far as we know, these are the first cycle-based solution method contributions regarding service network design problems while cycle-based formulations and algorithms exist for problems in other domain (e.g. p-cycle protection - Sebbah & Jaumard (2012) or truck collaboration - Ergun *et al.* (2007)). The authors show that we can exploit problem-specific assumptions (e.g. the number

of ports visited, the length of cycles, etc.) to support the process of designing solution methods for a specific problem. The authors present several cycles generation techniques, e.g. by using dual information of a flow problem or by column generation. They indicate how to use cutset inequalities to improve convergence speed of the Benders decomposition and how to manage the pool of variables regarding column generation.

Pedersen *et al.* (2009) present the generic capacitated multi-commodity fixed cost network design model with design balance constraints. The notation *design-balance constraints* is introduced and defined to express the requirements that incoming and outgoing resources at each node must be balanced. They also propose a first metaheuristic which is a two-phase tabu-search solution method for this problem. The first phase, which ignores the design-balance constraints, explores the solution space and searches for candidate solutions for the second phase. The second phase converts the candidate solutions into feasible solutions by a procedure which iteratively adds/removes paths to satisfy the design-balance constraints at all nodes. The experiment was performed on a set of 74 instances, the largest instance consists of 30 nodes, 720 arcs and 400 commodities. The results show that the procedure is quite effective to find feasible solutions for this problem. Starting from this work, we define another algorithm described in the next chapter which is able to find feasible solutions for the CMND in particular and is able to handle design-balance property in general.

Andersen *et al.* (2009b) present a comprehensive service network design problem which considers the management and coordination of multiple fleets of resources in an intermodal transporation system. It addresses multi-fleet management and coordination considerations, as well as the interactions between services being designed and services in collaborating transportation systems. Experiments are performed on a real-world problem, the Polcorridor project - zPo (2006). The authors analyze how synchronization with services and removal of border-crossing operations impact the throughput time for the freight. The computational results emphasize comprehensive formulations in addressing complex issues and point to the interest in continuing the research into the design and operation of systems involving multiple fleets.

To understand the characteristics of cycle-based formulations, Andersen *et al.* (2009a) summarize and compare four formulations based on arcs/cycles-based design variables and arc/path-based flow variables. The results show that the cycle-based formulations give promising characteristics such as lower-bound calculation, solution quality and computation time. Taking into account the advantages of cycle-based formulations, Andersen *et al.* (2011) propose a cycle-based branch-and-price algorithm. The experiment is performed on a set of 35 instances from

a rail freight transportation company which models various patterns in regard to the number of terminals, services, and demands. The average gap ranges from 3.8% to 7.3%; this indicates that the proposed branch-and-price framework with cycle-based formulation is an interesting approach to attack service network design problems.

Teypaz *et al.* (2010) propose and address a large-scale freight transportation problem which maximizes the profit of a carrier. In addition to basic descriptions (e.g. in Andersen *et al.* (2009b)), there are different types of vehicles; and the design-balances are enforced for vehicles types instead of each vehicle. The carrier can reject some demands completely or partially and no vehicle must be under nor over-used. In addition, a commodity is transported directly from its origin to its destination, there is no intermediate terminal. Demands are concentrated at some special terminals named hubs (there are maximum 3 hubs). The authors present heuristic algorithms based on a decomposition of the problem into three main steps: the construction of the network, the filling vehicles with commodities and the construction of the vehicle plannings. The algorithm is able to solve instances with 40 terminals, 480 periods, and three different vehicles types with short computation time (20 minutes).

As we observe, resource management has been integrated into service network design models to address issues in various domains such as air transportation, maritime transportation, ground transportation. Formulations, problem settings and solution methods are proposed to handle optimization issues at tactical level. Most of the research subjects focus on domain-specific problems in which domain-specific assumptions are exploited to define solution methods and experimentations. while this could give better experimental results, it may also limit the applicability of these research works to other problems. Studies regarding generic service network design are still limited. However, interesting characteristics are identified, e.g. cycle-based formulations, and are exploited to develop solution methods. However, there is still much work to do regarding service network design with resource management issues which includes problem settings and efficient solution methods to fill the gaps of this field.

3.3 Conclusion

This chapter surveyed the literature pertinent to our research. We reviewed briefly service network design problems, described the resource management issues that appear in the literature. We focused on research papers that study resource management issues in service network

design and present a review of each problem, solution method and experiment. For these works, we have identified the contributions, the limit of their studies and then what are the challenges for us. From that, we have identified the questions and defined the corresponding problems, the solution methods as well as the experiments. Details of our work are presented in the next three chapters.

Summary of the second part of the thesis

The second part of the thesis presents our research problems and our contributions with respect to these problems. Chapter 4 proposes a three-phase matheuristic algorithm for the capacitated multi-commodity network design with design-balance constraints. Chapter 5 proposes a service network design model that considers the management of resources at terminal level. The model explicitly takes into account the limitations of the available resources at each terminal. We propose an algorithmic framework including slope-scaling, column generation, heuristics and exact optimization for this model. Finally, Chapter 6 studies resources allocation issues in service network design which includes operations that allow the acquisition of capacity. A slope-scaling based solution method is developed to find feasible solutions for the problem. The experiments performed on these three problems indicate that the proposed algorithms are able to find good feasible solutions.

4. A THREE-PHASE MATHEURISTIC FOR CAPACITATED MULTI-COMMODITY FIXED-COST NETWORK DESIGN WITH DESIGN-BALANCE CONSTRAINTS

Summary of the Chapter 4

This chapter of the thesis presents our first work which was published in *Journal of Heuristics* with the following reference information:
D. M. Vu, T. G. Crainic, M. Toulouse: A three-phase matheuristic for capacitated multi-commodity fixed-cost network design with design-balance constraints, *J. Heuristics* 19(5): 757-795 (2013)

Abstract

This paper proposes a three-phase matheuristic solution strategy for the capacitated multi-commodity fixed-cost network design problem with design-balance constraints. The proposed matheuristic combines exact and neighbourhood-based methods. Tabu search and restricted path relinking meta-heuristics cooperate to generate as many feasible solutions as possible. The two meta-heuristics incorporate new neighbourhoods, and computationally-efficient exploration procedures. The feasible solutions generated by the two procedures are then used to identify an appropriate part of the solution space where an exact solver intensifies the search. Computational experiments on benchmark instances show that the proposed algorithm finds good solutions to large-scale problems in a reasonable amount of time.

Keywords: Network design, design-balance constraints, tabu search, path relinking, matheuristic

4.1 Introduction

This paper proposes a solution strategy for the *capacitated multi-commodity fixed-cost network design problem with design-balance constraints* (*DBCMND*), which is found in several important application domains, notably in transportation and the design of the service network of consolidation-based carriers when asset (resource) management issues are jointly considered (Crainic & Kim, 2007). The design of efficient solution methods for the DBCMND poses many challenges. The DBCMND is NP-hard and includes the capacitated multi-commodity fixed-cost network design problem (*CMND*), which is also NP-hard (Balakrishnan *et al.*, 1997), as a particular case. The difficulty in addressing the DBCMND, as compared to the more classical CMND, is actually compounded by the strong interplay between the flow circulation and the design-balance property of the design. Pedersen *et al.* (2009) actually states that even the search for feasible solutions of the DBCMND is "far from trivial."

We propose a three-phase matheuristic procedure combining exact and neighbourhood-based methods to address large DBCMND instances. We introduce two new heuristic methods, based on tabu search and path relinking principles, respectively, which cooperate to generate as many feasible solutions as possible. The tabu search algorithm introduces a variant of the cycle-based neighbourhood (Ghamlouche *et al.*, 2003) that exploits the CMND substructure of DBCMND, and a minimum-cost maximum-flow formulation to extract feasible solutions when the CMND ones do not respect the design-balance constraints. A different neighbourhood structure is proposed for the restricted path relinking procedure, which aims to identify a large number of good feasible solutions in a computationally efficient way. The feasible solutions generated by the two procedures are then used to build statistical information and, thus, identify an appropriate part of the solution space where an exact solver intensifies the search. Experiments show that the proposed matheuristic performs well on problem instances with a large network and many commodities.

The rest of the paper is structured as follows. Section 4.2 states the problem and recalls the model formulation. Section 4.3 presents a brief literature review. Section 4.4 details the proposed algorithm. We present and analyze the experimental results in Section 4.5, and provide concluding remarks in Section 4.6.

4.2 Problem Statement and Model Formulation

The DBCMND is the combinatorial optimization problem where given a potential network in terms of its arcs and a set of node-to-node, origin-to-destination (OD), demands, one must select a set of arcs to satisfy demand at minimum total cost, while enforcing the so-called design-balance constraints where the number of design arcs entering and exiting each node must be equal. Arcs are characterized by capacities limiting the total flow of commodities (OD demands) that use them. A fixed cost is incurred when an arc is selected (i.e., has positive flow), and a transportation cost is also paid for each unit of commodity flow passing through the arc.

In terms of consolidation-based carrier planning, nodes correspond to terminals and arcs to transportation services that could be operated between two terminals. The design-balance requirements correspond to so-called asset-management concerns. Assets typically refer to the resources e.g., crews or vehicles, needed to operate the selected transportation services, while asset management usually refers to the assignment of assets to services and their eventual repositioning.

This problem can be modeled by an oriented graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where the nodes, $\mathcal{N}$, represent the terminals, and the arcs, $\mathcal{A}$, represent the transportation services between these terminals. The set of commodities is noted $\mathcal{P}$ and represents the different products that must be transported between particular origin and destination nodes. For each commodity $p \in \mathcal{P}$, w_p stands for the amount of commodity p that must be moved from origin $o(p)$ to destination $d(p)$. A cost c_{ij}^p per unit of commodity p is associated with each arc (i, j) and represents the cost incurred when moving commodity p using this arc. The fixed cost of selecting and using arc $(i, j) \in \mathcal{A}$ is noted f_{ij}, and the capacity of the corresponding arc is noted u_{ij}.

Two sets of decision variables are associated with the formulation. A decision variable y_{ij} is associated with each arc: $y_{ij} = 1$ if arc (i, j) is used and $y_{ij} = 0$ otherwise. The continuous variables x_{ij}^p represent the flow distribution in the network in terms of the quantity of commodity p moved on arc (i, j). The arc-based mixed-integer formulation of the DBCMND (Pedersen *et al.*, 2009) is:

$$\min z(x,y) = \sum_{p\in\mathcal{P}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^p x_{ij}^p + \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} \tag{4.1}$$

$$\text{s.t.} \sum_{j\in\mathcal{N}_i^+} y_{ij} - \sum_{j\in\mathcal{N}_i^-} y_{ji} = 0, \forall i \in \mathcal{N}, \tag{4.2}$$

$$\sum_{j\in\mathcal{N}_i^+} x_{ij}^p - \sum_{j\in\mathcal{N}_i^-} x_{ji}^p = d_i^p, \forall i \in \mathcal{N}, \forall p \in \mathcal{P}, \tag{4.3}$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \leq u_{ij} y_{ij}, \forall (i,j) \in \mathcal{A}, \tag{4.4}$$

$$x_{ij}^p \geq 0, \forall (i,j) \in \mathcal{A}, \forall p \in \mathcal{P}, \tag{4.5}$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in \mathcal{A}. \tag{4.6}$$

In this model, $\mathcal{N}_i^+ = \{j \in \mathcal{N} | (i,j) \in \mathcal{A}\}$ and $\mathcal{N}_i^- = \{j \in \mathcal{N} | (j,i) \in \mathcal{A}\}$ define respectively the outward and inward neighbours of node $i \in \mathcal{N}$, while d_i^p stands for the flow value of commodity p at node i, which equals w_p for $i = o(p)$, $-w_p$ when $i = d(p)$, and 0 otherwise. The objective function (4.1) minimizes the total system cost, i.e., the fixed cost of the selected arcs plus the routing cost of the commodities. Equations (4.2) are the design-balance constraints, which ensure that the total number of open arcs terminating at any node must equal the number of open arcs going out of that node. Constraints (4.3) ensure flow conservation for each node and each commodity. Constraints (4.4), often referred to as bundle or forcing constraints, state that the total flow on an arc (i,j) cannot exceed its capacity u_{ij} when selected and must be 0 if it is not selected. Constraints (4.5) and (4.6) are non negativity and integrality constraints for the decision variables.

We use the following notation borrowed from (Pedersen *et al.*, 2009) in the subsequent sections of this paper. We define the *node imbalance* $\psi^i = \sum_{j\in\mathcal{N}^+(i)} y_{ij} - \sum_{j\in\mathcal{N}^-(i)} y_{ji}$ to be the difference between the number of open outgoing arcs and the number of open incoming arcs at node i. The *total system imbalance* $\psi^{\mathcal{N}} = \sum_{i\in\mathcal{N}} |\psi^i|$ represents the total absolute value of all the node imbalances. The *maximum absolute imbalance* $\psi^{max} = max\,(|\psi^i|)$ is the largest absolute value among all the node imbalances and indicates the difficulty of achieving feasibility.

Figure 4.1 illustrates these concepts on a five-node DBCMND instance. In this figure, the solid arcs represent the current design. We can see that node A has two outgoing design arcs and no incoming design arc, so $\psi^A = 2$; node D has two incoming design arcs

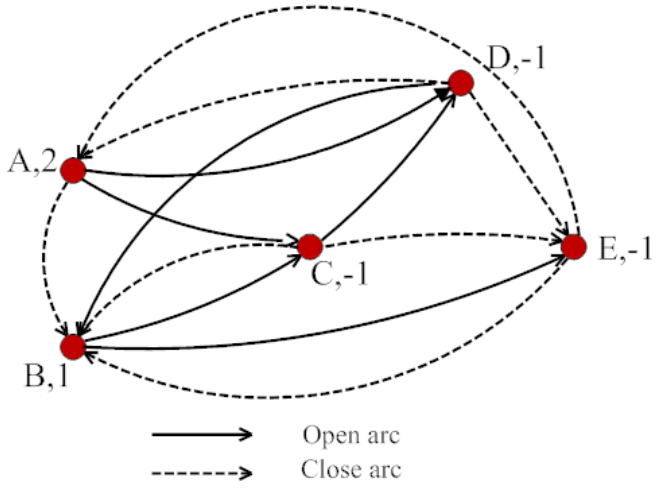


Figure 4.1: Design with node imbalance at each node.

and one outgoing design arc, so $\psi^D = -1$; etc. The total system imbalance is $\psi^{\mathcal{N}} = |\psi^A|+|\psi^B|+|\psi^C|+|\psi^D|+|\psi^E| = 6$. We can interpret the node imbalance at a given node i as the need for an additional $|\psi^i|$ empty vehicles moving in or out of that node. The cost of adding an empty vehicle is approximated by the average of the fixed cost of the arcs in the network $\overline{f}$ or by the average of the fixed cost of the closed arcs in the network $\bar{f}_{closed}$.

4.3 Literature Review

Network design formulations have a wide range of applications in transportation, logistics, telecommunication, and production systems, in particular. Magnanti & Wong (1984), Minoux (1989), and Crainic (2000) present generic models and applications of network design together with solution methods. For specific application areas, see the reviews of Christiansen *et al.* (2007), Cordeau *et al.* (1998), and Crainic & Kim (2007) for optimization models in maritime, rail, and intermodal transportation, respectively.

Early works addressing asset-management issues in service network design include Kim *et al.* (1999), Armacost *et al.* (2004), and Barnhart *et al.* (2002) for multimodal express network design, and Smilowitz *et al.* (2003) for multimodal package delivery with design-balance constraints for ground vehicles. In all these works, there must be an equal number of assets entering and leaving each node in the network. The solution methods use ad hoc techniques

that take advantage of the specific structure of the problems.

Pedersen *et al.* (2009) introduced the *design-balance* notation and the DBCMND. They proposed the first solution method for the DBCMND, a two-phase tabu search algorithm including a local search algorithm to handle the design-balance constraints and restore feasibility. In the first phase of the tabu search algorithm, neighbours are obtained by either adding or dropping arcs from the current solution, while satisfying the flow constraints but ignoring the design-balance constraints. A second phase seeks to convert the last solution of the first phase into a design-balanced feasible solution by generating solutions with a path-based neighbourhood structure. At each iteration, the procedure obtains neighbours by adding or removing paths from the current solution while ignoring the flow constraints. The add/drop moves in this phase reduce the total system imbalance by two after each iteration, but many iterations may be required to obtain a design-balanced feasible solution.

Andersen *et al.* (2009b) and Andersen *et al.* (2009a) considered additional aspects of asset management such as the management and coordination of multiple fleets and fixed-length, cyclic schedules. This problem is denoted SNDAM. The authors compared the cycle-based and path-based formulations of SNDAM, concluding that cycle-based formulations contribute to efficient model solving. However, this study was based on an a prior enumeration of cycles and paths, and it cannot be directly applied to instances of realistic dimensions. Inspired by their previous work, Andersen *et al.* (2011) proposed the first branch-and-price approach using a cycle-based formulation for the SNDAM.

Chouman & Crainic (2010) proposed a hybrid of cutting planes and tabu search procedures. The latter included a cycle-based neighbourhood structure that directly addresses the design-balance requirements, but does not account for the corresponding flow-distribution feasibility. The need to verify this condition for each neighbour requires to solve the associated minimum-cost multi-commodity network flow problem, which is computationally very heavy.

Research into network design with asset management has thus been limited so far. On the one hand, the existing exact solution methods have limitations on the size of instances they may efficiently address. On the other hand, research on efficient meta-heuristics able to address larger instances has been extremely limited. Our goal is to fill this gap, by returning to the generic DBCMND problem setting.

4.4 Proposed Matheuristic for the DBCMND

Our approach to the DBCMND exploits the natural complementarity between exact and neighbourhoodbased search methods. Exact solvers can only partially explore the solution space of large instances. To ensure success, the exploration should focus on promising regions of the solution space. We use heuristic methods to find feasible solutions from which we collect information that can be used to restrict the dimension of the problem. Then, the restricted problem is addressed using an exact solver.

Algorithm 4.1 depicts the implementation sequence of this strategy. In the Initialization phase, CPLEX is used to solve a relaxation of DBCMND obtained by removing the design-balance constraints. This relaxation is the well-known CMND problem. It yields an initial solution that satisfies all the flow constraints but may not satisfy the design-balance constraints. Design-balance feasible solutions are then identified during Phase 1 by a tabu search procedure, which starts from the initial solution returned by the Initialization phase.

The feasible solutions found by the tabu search procedure during Phase 1 form the reference set of the path relinking procedure in Phase 2. Different from the traditional goal of path relinking focusing on improving a given solution, the purpose of the procedure we propose is to generate many new “good" and diverse feasible solutions that, together with those generated by the tabu search, are analyzed in Phase 3 to determine which arcs can be fixed open or closed. Fixing the status of some decision variables reduces the size of the problem instance and an exact MIP solver may then be used to address it. The overall best solution of the algorithm is usually found in Phase 3.

Algorithm 4.1 Proposed solution method template

Initialization. Solve a relaxed formulation of DBCMND to obtain an initial (possibly, design-balance unfeasible) solution.
Phase 1 - Tabu Search. First exploration of the solution space to identify feasible solutions.
Phase 2 - Path Relinking. Using the feasible solutions found by the tabu search method as the reference set, generate new and diverse feasible solutions to enrich the feasible solution set.
Phase 3 - MIP Intensification. Use statistical information obtained from the feasible solution set to fix variables and use a mixed-integer solver to address the corresponding reduced-size problem.
Output the best solution found.

4.4.1 Tabu Search Phase

The tabu search meta-heuristic (Glover, 1989, 1990) we propose for Phase 1 is inspired from Chouman & Crainic (2010) and Pedersen *et al.* (2009) while improving on some of the limitations of these two contributions. Relative to the former, we relax the requirement that neighbours must satisfy the design-balance constraints when satisfying the flow constraints. In contrast to the latter, we seek solutions that are DBCMND feasible at each tabu search iteration. Thus, once a solution in the neighbourhood of the current solution has been selected as the new solution, our tabu search method tries to restore feasibility with respect to the design-balance constraints by solving a minimum-cost maximum-flow problem. As supported by numerical experiments, compared to the search-based feasibility restoration approach of Pedersen *et al.* (2009), our approach is much faster and always identifies the set of arcs with the smallest possible cost.

4.4.1.1 Tabu Search Neighbourhood Exploration

The neighbourhood in our tabu search builds on the cycle-based neighbourhood of Ghamlouche *et al.* (2003), proposed to explore the space of the design variables for CMND. This procedure identifies neighbours by first selecting a pair of nodes and two paths connecting those nodes, thus forming a cycle. Next, the flow on one path of the cycle is redirected to the other path (the alternative path) until the flow ceases on at least one arc.

Our cycle-based neighbourhood procedure performs flow redirections that satisfy the flow distribution but do not guarantee that the design-balance constraints will be satisfied. Flow redirection is performed independently for each commodity that has flow on a given arc (i, j). This increases the chances of identifying neighbours that satisfy the flow distribution constraints because the entire demand can be satisfied via several paths. We further enhanced the procedure of Ghamlouche *et al.* (2003) by considering other nodes, not always i and j, as the source and destination nodes of the alternative path, to facilitate the search for an alternative path for each commodity. Our flow redirection procedure is a loop that iterates on the set of commodities, identifying alternative nodes to nodes i and j for each commodity p that has flow on arc (i, j), and computing an alternative path to take the flow of commodity p from arc (i, j). We now describe the implementation of this cycle-based neighbourhood exploration procedure.

Finding alternative source and destination nodes. Let $\mathcal{P}_{ij} = \{p \in \mathcal{P} | x_{ij}^p > 0\}$ be the set of commodities with flow on arc (i, j). To close arc (i, j), we have to redirect this flow to other arcs while maintaining the flow constraints. We seek an alternative source node s^p for node i and an alternative destination node t^p for node j. For example, suppose we want to remove arc $(3, 4)$ with a flow of 2 units in the single-commodity graph of Figure 4.2. The flow on arc (3, 4) can be redirected to the subpath {(3, 7), (7,4)}. However, if the alternative source and destination nodes of nodes 3 and 4 are 2 and 5, then we can redirect the flow on arc $(3, 4)$ to subpath {(2, 3), (3, 7), (7, 4), (4, 5)}, or subpath {(2, 7), (7, 4)}, or subpath {(2, 3), (3, 7), (7, 5)}, etc. We cannot start from node 1 because arc $(1, 2)$ routes only 1 unit whereas arc $(3, 4)$ currently routes 2 units.

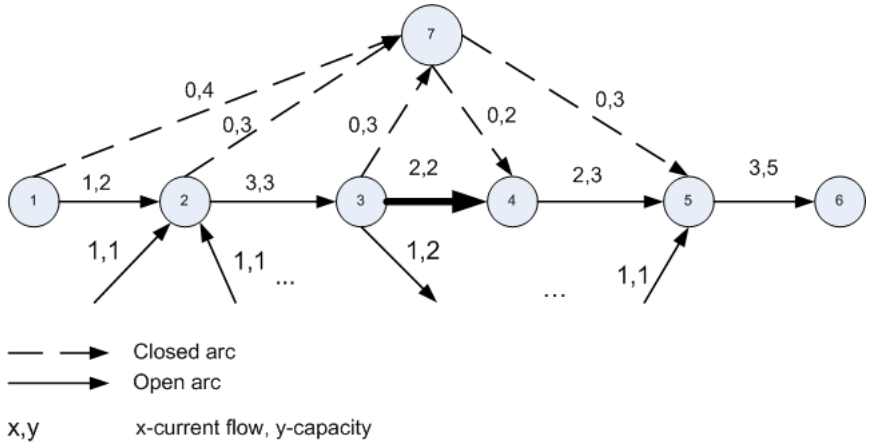


Figure 4.2: Illustration of alternative origin & destination cycle nodes

For each commodity p, we need to find nodes s^p and t^p such that the path connecting s^p and t^p passing through arc (i, j) can route at least x_{ij}^p units of commodity p. We find s^p (respectively t^p) using a breadth-first search algorithm starting from node i (j). We choose the node s^p (t^p) that is the furthest from node i (j), where the distance is measured by the number of arcs on each path. Intuitively, the greater the distance between s^p and i (j and t^p), the greater the probability of obtaining a feasible alternative path between s^p and t^p.

The procedure for identifying an alternative source node s^p is described in Algorithm 4.2. We initialize the queue with node i. When a node v is popped from the queue, the procedure tries to add another node u for which there is an arc (u, v) in the current design and the flow $x_{uv}^p \geq x_{ij}^p$. The last node v with no incoming arcs satisfying this condition is chosen as the source node s^p because we want to choose the node with the greatest distance from node i. The version of Algorithm 4.2 that finds an alternative destination node t^p is similar and is not shown here.

Algorithm 4.2 Source_Search((i,j),p). Search for alternative source node s^p for arc (i,j).

$queue \leftarrow i$
while $queue \neq \varnothing$ **do**
  $v = queue.pop()$;
  for each $(u,v) \in \mathcal{A} \wedge y_{uv} = 1$ **do**
    if $x^p_{uv} \geq x^p_{ij}$ **then**
      $queue.push(u)$;
      Save trace-back information;
    end if
  end for
  if $queue = \varnothing$ **then** $s^p = v$, **break**
end while
return s^p.

Residual graph definition. Once the alternative source node s^p and destination node t^p are known, we proceed to compute a path connecting s^p and t^p to redirect the flow of commodity p from arc (i,j). A residual graph $\mathcal{G}^{\mathcal{R}}_p = (\mathcal{N}, \mathcal{A}^{\mathcal{R}}_p)$ is generated for this purpose for each commodity p. The set of arcs $\mathcal{A}^{\mathcal{R}}_p = \mathcal{A}^{\mathcal{R},open}_p \cup \mathcal{A}^{\mathcal{R},closed}_p$ is defined as follows. All closed arcs are included in the residual graph if their capacity u_{ij} is greater than or equal to the flow x^p_{ij} of commodity p on arc (i,j), i.e., $\mathcal{A}^{\mathcal{R},closed}_p = \left\{(k,l) \in \mathcal{A}\} | y_{kl} = 0 \wedge u_{kl} \geq x^p_{ij}\right\}$. Each open arc (k,l) (except arc (i,j)) is included in the residual graph if its residual capacity r_{kl} is greater than or equal to x^p_{ij} or if arc (k,l) is in $path_{s,t}$:

$$\mathcal{A}^{\mathcal{R},open}_p = \left\{(k,l) \in \mathcal{A} | \left(y_{kl} = 1 \wedge r_{kl} \geq x^p_{ij} \wedge (k,l) \notin path_{s,t}\right) \vee \left(y_{kl} = 1 \wedge (k,l) \in path_{s,t}\right)\right\}.$$

The residual capacity of (k,l) is $r_{kl} = u_{kl} + x^p_{ij} - \sum_{p \in \mathcal{P}} x^p_{kl}$, when $(k,l) \in path_{s,t}$, or $r_{kl} = u_{kl} - \sum_{p \in \mathcal{P}} x^p_{kl}$, otherwise.

The cost of a closed arc $(k,l) \in \mathcal{A}^{\mathcal{R},closed}_p$ is the fixed cost f_{kl} of opening the arc plus the cost of routing the commodity. The cost of an open arc $(k,l) \in \mathcal{A}^{\mathcal{R},open}_p$ is simply $c^p_{kl} x^p_{ij}$, the cost of routing the commodity. We use a classic shortest-path algorithm to find an alternative path between source node s^p and destination node t^p in the residual graph; the details are not given here.

Generating neighbours. When the flow has been redirected successfully for all the commodities using arc (i,j), the cycle-based neighbourhood procedure closes arc (i,j) and any other arcs with no residual flow. This yields a neighbouring solution of the current solution x

that we indicate by x_{ij}. The steps of the cycle-based neighbourhood procedure are summarized in Algorithm 4.3. The current solution x is represented by its flow distribution $\mathcal{X}$ and its arc set $\mathcal{Y}$. The cost of x is indicated by $c(x)$, and $\mathcal{P}_{ij}$ stands for the set of commodities routed over arc (i,j). The procedure first computes the cost of the neighbouring solution x_{ij} after arc (i,j) has been removed from the current solution x (lines 1 and 2). Next, for each commodity $p \in \mathcal{P}_{ij}$ routed by arc (i,j), the appropriate source node s^p and destination node t^p are identified, the corresponding residual graph is generated, and a shortest path is obtained to reroute the flow of commodity p (lines 4 to 7). If a feasible path is found, the cost, the flow distribution, and the design of the neighbouring solution x_{ij} are updated (lines 9 and 10). The generation of the neighbour continues until all the commodities routed by arc (i,j) are rerouted. If it is not possible to reroute all the commodities, it is assumed that arc (i,j) cannot be removed and the neighbour for this arc cannot be generated.

Algorithm 4.3 Close_Open_Arc($\mathcal{X}, \mathcal{Y}, c(x)$, (i,j), $\mathcal{P}_{ij}$). Cycle-based neighbourhood procedure.

1. $c(x_{ij}) = c(x) - f_{ij} - \sum_{p \in \mathcal{P}_{ij}} c_{ij}^p x_{ij}^p$;
2. $\mathcal{Y}_{ij} = \mathcal{Y} \backslash (i,j)$;
3. **while** $\mathcal{P}_{ij} \neq \emptyset$ **do**
4. Choose a commodity $p \in \mathcal{P}_{ij}$;
5. Find source node s^p and destination node t^p;
6. Construct residual graph $\mathcal{G}^p$ for commodity p;
7. Find shortest path π^p connecting s^p and t^p on $\mathcal{G}^p$;
8. **if** $\pi^p \neq \emptyset$ **then**
9. Update $c(x_{ij})$ based on π^p;
10. Update flow information of $\mathcal{X}_{ij}$ and design of $\mathcal{Y}_{ij}$;
11. Remove commodity p, $\mathcal{P}_{ij} = \mathcal{P}_{ij} \backslash p$;
12. **else return** *unfeasible*
13. **end while**
14. Close all arcs with no residual flow and update fixed cost;
15. **return** x_{ij}.

4.4.1.2 Unfeasibility-monitoring Scheme

To generate the entire neighbourhood of the current solution, Algorithm 4.3 is called iteratively for each arc (i,j) of the current solution. The solutions returned are feasible with respect to the commodity flows but may not be feasible in terms of the design-balance constraints. Satisfying the design-balance constraints for all the solutions in the neighbourhood would be too costly. Instead, we would like to choose the neighbour that has the best chance of becoming

completely or almost completely feasible. We use the unfeasibility-monitoring scheme proposed by Pedersen *et al.* (2009) for this purpose. A penalty is used to estimate how far the solution is from feasibility with respect to the design-balance constraints.

Using the vocabulary of transportation applications, the node imbalance at a given node i may be interpreted as the need to add $|\psi^i|$ empty-vehicle services in or out of that node. The cost $\widetilde{f}$ of adding an empty vehicle is approximated as the product of the average $\overline{f}$ of the fixed cost of the arcs of neighbouring solution x_{ij} and an empirical scaling parameter τ used to control the importance of the penalty when evaluating neighbouring solutions. Then, to obtain a penalty that increases non-linearly with the unfeasibility of the solution, we multiply the estimated cost of an "empty-vehicle service" (arc), the total system imbalance of solution x_{ij}, and the maximum node imbalance of x_{ij}, i.e., the penalty $P_{ij} = \widetilde{f}\psi^{\mathcal{N}}\psi^{max}$, where $\widetilde{f} = \tau\overline{f}$. This penalty is added to the cost of neighbouring solution x_{ij} yielding its *total system cost* $V_{ij} = c(x_{ij}) + P_{ij}$. When comparing neighbouring solutions, P_{ij} provides the means to skew the evaluation towards feasibility, a neighbour with a relatively high total system cost that is close to feasibility may be preferred to one with a lower total system cost that is further from feasibility.

4.4.1.3 Design-balance Feasibility

The neighbour that minimizes the total system cost is thus selected and then, if required, one attempts to restore feasibility with respect to the design-balance constraints.

Consider a current solution x that violates the design-balance constraints (Figure 4.1 illustrates such a case). We seek to reduce the total system imbalance of x to zero by introducing paths that contain only arcs that are closed in the current solution. These paths connect nodes with opposite imbalance signs. To reduce the total system imbalance, these paths will start from nodes with negative imbalances and end in nodes with positive imbalances. We add the paths that give the greatest reduction in the total system imbalance. When several paths satisfy this criterion, we choose the one with the smallest cost. The arcs on these paths are found by solving a minimum-cost maximum-flow problem as described below.

Let $\mathcal{N}^+ = \{u \in \mathcal{N} | \psi^u > 0\}$ and $\mathcal{N}^- = \{u \in \mathcal{N} | \psi^u < 0\}$ be the subsets of nodes in x that have positive and negative imbalances, respectively. Recall that in a network, the total positive and negative imbalances are the same, i.e., $|\sum_{u\in\mathcal{N}^+} \psi^u| = |\sum_{u\in\mathcal{N}^-} \psi^u|$, implying

$\sum_{u\in\mathcal{N}^+}\psi^u+\sum_{u\in\mathcal{N}^-}\psi^u=0$. Let $\mathcal{G}^{\mathcal{F}}=(\mathcal{N}\cup\{s,t\},\mathcal{A}^{\mathcal{F}}\cup\mathcal{A}^{\mathcal{G}})$ be the *minimum-cost maximum-flow graph* corresponding to solution x. $\mathcal{A}^{\mathcal{F}}=\{(i,j)\in\mathcal{A}|y_{ij}=0\}$ is the set of closed arcs in solution x. The cost of each arc in $\mathcal{A}^{\mathcal{F}}$ is set to the fix cost of the corresponding arc in the original problem, while its capacity is fixed to 1. Nodes s and t are artificial nodes connected to nodes in $\mathcal{N}$ by arcs in $\mathcal{A}^{\mathcal{G}}$ as follows: There is an arc $(s,u)\in\mathcal{A}^{\mathcal{G}}$ with cost 0 and capacity $-(\psi^u)$ for every $u\in\mathcal{N}^-$. Similarly, there is an arc $(u,t)\in\mathcal{A}^{\mathcal{G}}$ with cost 0 and capacity ψ^u for every $u\in\mathcal{N}^+$. The graph of the problem displays now the classic structure of a minimum-cost maximum-flow problem.

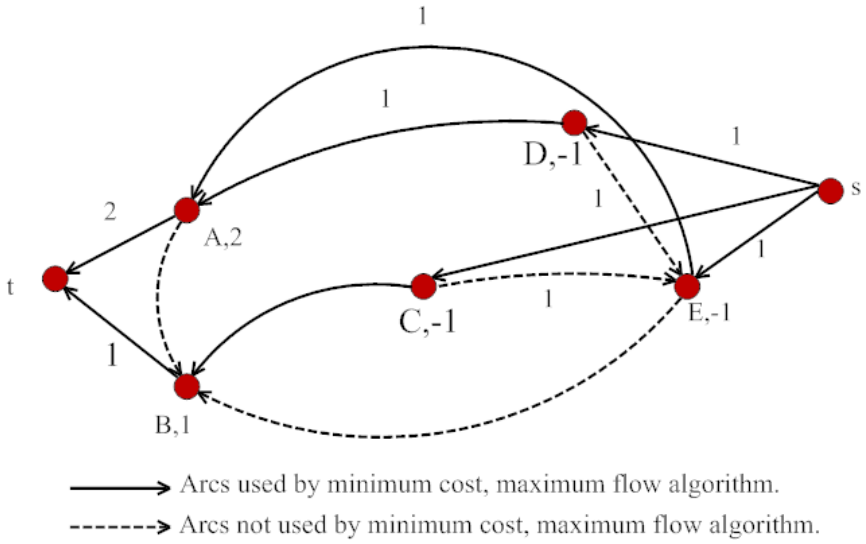


Figure 4.3: The equivalent minimum-cost maximum-flow network

Figures 4.1 and 4.3 illustrate this procedure. The latter displays the minimum-cost maximum-flow graph corresponding to the unfeasible solution in the former. Arcs (s,D), (s,C), and (s,E) in Figure 4.3 have capacity 1 and connect the artificial node s to nodes in the solution of Figure 4.1 with negative imbalances. Arcs (A,t) and (B,t), with respective capacities of 2 and 1, connect nodes A and B with positive imbalances in the solution of Figure 4.1 to the artificial node t. All the other arcs in Figure 4.3 form the set $\mathcal{A}^{\mathcal{F}}$ of non-artificial arcs in $\mathcal{G}^{\mathcal{F}}$ and correspond to closed arcs in the solution of Figure 4.1.

The minimum-cost maximum-flow algorithm then seeks to route $\sum_{u\in\mathcal{N}^+}\psi^u$ units of flow between s and t in $\mathcal{G}^{\mathcal{F}}$. Discarding the arcs connecting the artificial nodes s and t, the solution of the minimum-cost maximum-flow algorithm yields a set of arcs $\mathcal{A}'\subseteq\mathcal{A}^{\mathcal{F}}$, illustrated in Figure 4.3 by the non-artificial solid arcs, which are added to the design of the current solution x to satisfy the design-balance constraints (illustrated in Figure 4.4). Notice that, since the

capacity of the arcs in $\mathcal{A}^{\mathcal{F}}$ is set to 1, the flow routed over $\mathcal{G}^{\mathcal{F}}$ decomposes into several disjoint paths. Each path starts from a node u with $\psi^u < 0$ and ends in a node v with $\psi^v > 0$. Moreover, for each node $z \in \mathcal{N}$ where $\psi^z = 0$, the numbers of new incoming and outgoing arcs are equal because of the flow conservation constraints. Therefore, the addition of the new arcs of $\mathcal{A}'$ does not change the balance at node z when $\psi^z = 0$ in the current solution.

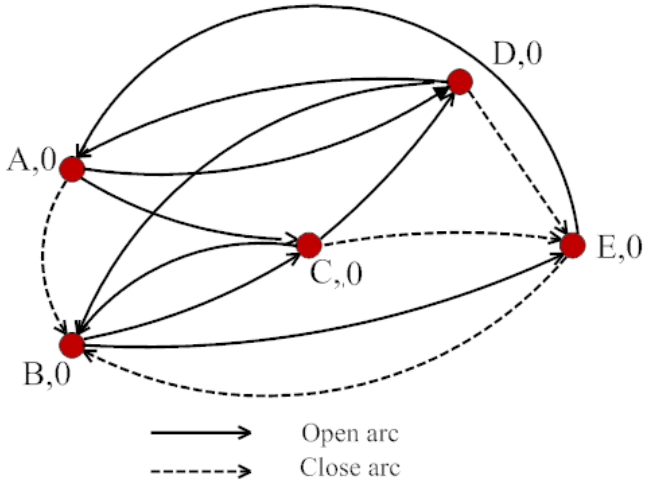


Figure 4.4: Feasible solution with arcs used by minimum-cost maximum-flow algorithm

The minimum-cost maximum-flow algorithm allows us to find the maximum flow that can be routed from s to t over $\mathcal{G}^{\mathcal{F}}$ at the smallest cost, which is equal to the cost of the arcs in $\mathcal{A}^{\mathcal{F}}$ used to route the flow. The maximum flow is the number of disjoint paths that we can obtain in $\mathcal{A}^{\mathcal{F}}$ by connecting a node u and a node v where $\psi^u < 0$ and $\psi^v > 0$. Each path reduces the total system imbalance by two, and the maximum flow corresponds to the smallest total system imbalance that we can obtain by adding arcs to the current design. This allows us to directly obtain a good feasible solution from an unbalanced and unfeasible neighbouring solution, if such a feasible solution exists for the unbalanced solution. Even if we do not obtain a feasible solution after matching the nodes with opposite signs, the added arcs reduce the total system imbalance as much as possible, bringing the solution closer to feasibility with respect to the design-balance constraints.

4.4.1.4 The Tabu Search Algorithm

The evolution of the tabu search algorithm is controlled through a short-term tabu list recording each arc (i, j) that originated a move, i.e., flow has been diverted from arc (i, j) and the

corresponding neighbouring solution x_{ij} has been selected as the new solution. The tabu list prevents the reopening of arc (i, j) for a predefined number of tabu search iterations, and it is used when the algorithm computes alternative paths, as well as by the minimum-cost maximum-flow algorithm. In the former case, a tabu arc cannot be used to divert the flow from another arc, even when this prevents finding an alternative path. With respect to the latter, closed arcs in the tabu list cannot be reopened to satisfy the design-balance constraints. In this case, the tabu list prevents the cycling that might otherwise occur.

Arcs that are opened by the minimum-cost maximum-flow algorithm to satisfy the design-balance constraints are also placed in the tabu list. The tabu status of these arcs prevents their rapid removal from the solution by tabu search moves. This induces the tabu search algorithm to explore regions of the solution space where the design-balance constraints are satisfied.

Algorithm 4.4 gives the main steps of the proposed tabu search algorithm, which stops after executing a predefined number of iterations or when a maximum computational time is reached. The search space is the set of design solution vectors that are feasible with respect to all the flow constraints and variable constraints. Moves maintain the flow and variable constraints, but new solutions may violate the design-balance constraints. However, only solutions that satisfy the design-balance constraints are stored in the solution set. The solution set becomes the input to the restricted path relinking phase of our algorithm.

Algorithm 4.4 Tabu search algorithm for the DBCMND

Initialization Solve a relaxation of DBCMND to obtain an initial (potentially unfeasible) solution;
Exploration & Feasibility.
while *stopping conditions* not satisfied, **do**
  For each arc of the current solution, generate a neighbour solution (Section 4.4.1.1);
  Select the neighbour that minimizes the total system cost as the new solution;
  Solve the minimum-cost maximum-flow problem (Section 4.4.1.3) to obtain a set of arcs reducing the total system imbalance of the new current solution;
  Add these arcs to the current solution;
  Update the tabu list;
  Solve the corresponding multi-commodity minimum-cost flow problem using the new design;
  If a DBCMND feasible solution is obtained, store it in the solution set and update the global optimal solution, if necessary.
end while

4.4.2 Path Relinking Phase

Despite a neighborhood structure that identifies rather easily good candidate solutions at each tabu search iteration, and despite the fact that the minimum-cost maximum-flow algorithm facilitates obtaining DBCMND feasible solutions, we have observed that the number of feasible solutions found by tabu search is still too small to accurately determine regions of the solution space in which to intensify the search. The goal of the restricted path relinking phase is to enrich this set starting from the feasible solutions found by tabu search algorithm. Notice that, different from most path relinking methods in the literature, the focus is on generating many “good" feasible solutions, not on improving a given solution.

Path Relinking (Glover, 1997; Glover *et al.*, 2000) is a population-and-neighborhood-based meta-heuristic that operates on a set of solutions, called the *reference set*, generating paths between solutions in this set that yield new solutions. Each iteration of this algorithm starts from an *initial solution* and builds a path toward a *guiding solution*, by performing moves that progressively introduce into the initial solution the desirable *attributes* of the guiding solution. This exploration allows the search to perform moves that may be unattractive according to the objective function value but appear essential for reaching solutions with given characteristics. The design of a path relinking algorithm involves specifying the reference set, how the initial and guiding solutions are selected in the reference set, which guiding attributes need to be introduced into the initial solution, and at least a neighborhood defining how the path from the initial solution to the guiding solution is to be built.

The reference set contains all the feasible solutions found during the tabu search phase if the number is less than or equal to a parameter l_r, or the best l_r solutions, otherwise. The backward path relinking procedure we designed then selects the best solution in the reference set as its initial solution, and the second-best as guiding solution. When a path has been completed between the two, the guiding solution is removed from the reference set. We have observed that backward path relinking tends to find better solutions than a forward version. The former explores more thoroughly the neighborhood of the initial solution, the best solution, where one expects to find good solutions, while the increasingly restricted neighborhood along the path may not provide the same opportunity. In the following, “current solution” refers to the initial solution at the current iteration of the path relinking algorithm, as transformed in all previous iterations.

The guiding attributes are arcs that are open or closed in the guiding solution. Therefore, we seek to introduce into the current solution arcs that are open in the guiding solution but closed in the initial solution, and remove arcs that are closed in the guiding solution but open in the initial solution. We define a neighborhood and move accordingly.

A neighbor therefore is a solution obtained from the current solution by opening at least one closed arc in the current solution that is open in the guiding solution and by closing at least one open arc that is closed in the guiding solution. This move is performed using a path-exchange neighborhood procedure, where the path removed from the current solution contains only arcs that are open in the current solution and closed in the guiding solution, while the alternate path to be introduced contains only arcs that are closed in the current solution with at least one of them being open in the guiding solution.

Algorithm 4.5 details the neighbour-identification procedure for an arc (i, j) open in the current solution but closed in the guiding solution. It returns the path to open, if it exists, "Fail", otherwise. The procedure starts by identifying the start s and end t nodes of the path to be removed (this is similar to the tabu search neighborhood). A simple backward search is performed starting from i to find node s such that the path between i and s contains only open arcs that do not belong to the guiding solution. The same procedure is applied to j to find a node t such that the path between j and t contains only arcs that are closed in the guiding solution. We thus obtain a path $s..i - j..t$ to close, its arcs being stored in $path\_to\_close$.

Algorithm 4.5 Path_exchange_neighbourhood((i, j))

Find nodes s and t;
$path\_to\_close$ = path connecting s and t passing through arc (i, j), all arcs being closed in the guiding solution;
Generate graph $\mathcal{G}^{PR} = \left(\mathcal{N}, \mathcal{A}^{PR}\right)$;
Find the shortest paths from s to all nodes u in $\mathcal{G}^{PR}$ and save these in $d^s[u]$;
Find the shortest paths from t to all nodes v in $\mathcal{G}^{PR,inverse}$ and save these in $d^t[v]$;
Find an arc (u, v) closed in the current solution but open in the guiding solution minimizing $d^s[u] + d^t[v] + f_{uv}$;
$path\_to\_open$ = path connecting s and t passing through arc (u, v);
if No $path\_to\_open$ has been found **then**
  Return "Fail";
end if
Return $path\_to\_close$, $path\_to\_open$.

Similarly, the procedure seeks to identify a path to open between nodes s and t and, if successful, this path is stored in $path\_to\_open$. The path is computed based on graphs $\mathcal{G}^{PR} =$

$\left(\mathcal{N}, \mathcal{A}^{PR}\right)$, defined on the set of closed arcs in the current solution $\mathcal{A}^{PR} = \{(i,j) \in \mathcal{A} | y_{ij} = 0\}$, and $\mathcal{G}^{PR,opposite} = \left(\mathcal{N}, \mathcal{A}^{PR,opposite}\right)$, obtained by reversing the direction of all arcs in $\mathcal{A}^{PR}$. The cost of each arc $(i,j) \in \mathcal{G}^{PR}$ (and its opposite) is set as the product of its capacity u_{ij} and the maximum-flow cost on arc (i,j), $c_{ij}^{PR} = f_{ij} + max\{c_{ij}^{p}\}_{p \in \mathcal{P}} u_{ij}$. This is an upper bound on the contribution to the objective value of the current solution from the arc (i,j). We then compute the shortest paths from s to all the other nodes in $\mathcal{G}^{PR}$ and store the results in the array $d^s[..]$. Similarly, we compute the shortest paths from t to all other nodes in $\mathcal{G}^{PR,opposite}$, and store the results in $d^t[..]$. Finally, we identify an arc (u,v), which is closed in the current solution but open in the guiding solution, minimizing the sum $d^s[u] + d^t[v] + f_{uv}$, where $d^s[u]$ and $d^t[v]$ are the costs of the shortest paths from node s to node u and from node v to node t, respectively. A random selection is made when several nodes s and t are identified (instead of choosing the furthest node as in the tabu search neighborhood). The resulting path $s..u - v..t$ completes the exchange.

The restricted path relinking procedure is described in Algorithm 4.6. The main loop iterates over the elements of the reference set, each being discarded after it has been used as the "guide" for an iteration of this loop, the procedure stopping when the reference set becomes empty. The inner **while** loop then generates solutions on the path between the current pair of solutions, exploring the path-exchange neighbourhood (Algorithm 4.5) between the paired solutions. The arcs to close/open, sets $ArcsToClose$ and $ArcsToOpen$, respectively, are those open/closed in the current solution but closed/open in the guiding solution. The inner loop ends when there are no more arcs to open or close. Since a move opens at least one arc in the current solution that is closed in the guiding solution and closes at least one arc that is open in the guiding solution, the current solution evolves towards the guiding solution. Note that an arc cannot be re-opened if it has been closed in the current iteration of the inner loop. Similarly, arcs that have been open in an iteration of the inner **while** loop cannot be closed even if they do not belong to the set of open arcs in the guiding solution.

Notice that, similarly to most cycle-based neighbourhoods (e.g., Ghamlouche *et al.*, 2003), the path-exchange neighborhood defined above does not attempt to guarantee the feasibility of the neighbour solution with respect to the flow-conservation constraints. Thus, while the new design obtained by replacing path $s..i - j..t$ with path $s..u - v..t$ is always design-balance feasible, it may not satisfy the flow-balance constraints. A feasible solution is then computed by solving the restricted DBCMND problem in which all the arcs of the current solution are fixed. This feasible solution may have a higher objective-function value than the best solution

Algorithm 4.6 Main path relinking procedure

Reference set $\mathcal{R}$ = the best feasible solutions yielded by the tabu search procedure;
Set $InitialSolution$ to the best solution in $\mathcal{R}$;
while $\mathcal{R} \neq \emptyset$ **do**
  Set $GuidingSolution$ to the second-best solution in $\mathcal{R}$, and remove it from $\mathcal{R}$;
  Set $CurrentSolution = InitialSolution$;
  Define $ArcsToClose$ = Set of open arcs in the $CurrentSolution$ that are not open in the $GuidingSolution$;
  Define $ArcsToOpen$ = Set of closed arcs in the $CurrentSolution$ that are not open in the $GuidingSolution$;
  while $ArcsToClose \neq \emptyset$ and $ArcsToOpen \neq \emptyset$ **do**
    Randomly select an arc $(i, j) \in ArcsToClose$
    Set $ArcsToClose = ArcsToClose \setminus (i, j)$;
    if Path_exchange_neighbourhood((i, j)) (Algorithm 4.5) $\neq$ Fail **then**
      $ArcsToClose = ArcsToClose \setminus path\_to\_close$;
      $ArcsToOpen = ArcsToOpen \setminus path\_to\_open$;
      Close all arcs in $CurrentSolution$ that belong to $path\_to\_close$;
      Open all arcs in $CurrentSolution$ that belong to $path\_to\_open$;
      Set up a restricted DBCMND problem by fixing all the design arcs in $CurrentSolution$ and solve the resulting MIP to identify a flow-feasible solution;
      if Feasible solution is found **then**
        Add it to the solution set;
      end if
    end if
  end while
end while

of the tabu search procedure, the path-exchange move actually performs a very disturbing diversification move without a subsequent enhancement phase, but, as the experimental results show, it is generally not too far. In any case, the feasible solution is added to the solution set, but it does not replace the current solution of the path relinking phase.

4.4.3 Intensification Phase

The goal of Phase 3 is to find better solutions by intensifying the search in promising regions of the solution space. The identification of such promising regions is obtained by fixing to open or closed certain arcs according to their frequency in good solutions identified during the search. Two restricted DBCMND problems are thus defined, which are then solved by an exact MIP solver.

Let S be the solution set generated by the tabu search and restricted path relinking procedures, $c(x)$ the cost of a solution $x \in S$, and $maxcost^S$ the cost of the worst solution in the set. Define the $rate_{ij}$ of an arc (i, j) as

$$rate_{ij} = \sum_{x \in S} y_{ij}^x * \left(1 - \frac{c(x)}{maxcost^S}\right), \forall (i,j) \in \mathcal{A}, \tag{4.7}$$

where y_{ij}^x = 1 if arc (i, j) is open in the solution x, 0 otherwise. The rate expression assigns greater weights to arcs that either belong to good solutions in the set or that appear in many solutions of the set. Rates are then normalized in the range $[0, 1]$ by dividing them by the maximum value $rate_{max} = max\{rate_{ij} | (i, j) \in \mathcal{A}\}$. An arc (i, j) is then fixed to open or closed based on its normalized $rate_{ij}$ value.

The first restricted problem is defined by assuming that arcs with a high rating are more likely to appear in the optimal solution. Consequently, the arcs for which $rate_{ij}$ is greater than a certain threshold α are fixed to open. Let x_{open} be the optimal solution of the corresponding restricted DBCMND problem. Symmetrically, we assume that arcs with a low rating are unlikely to be part of the optimal solution, and fix to closed the arcs with a $rate_{ij}$ lower than a certain threshold β. Obviously, we do not fix arcs appearing in x_{open}. The resulting restricted DBCMND is then solved to optimality. The final solution of the proposed matheuristic is then the best solution among those generated by the tabu search, path relinking, and intensification phases.

4.5 Computational Results

The algorithm we propose, and that we call *TS-PR* in the following, is validated empirically using the problem instances used in Pedersen *et al.* (2009), as well as in a number of other papers in the literature (e.g., Ghamlouche *et al.*, 2003, 2004). There are two sets of instances identified as R and C, respectively. Both sets are general transshipment networks with no parallel arcs and one commodity per origin-destination pair. The same arc unit cost is used for all commodities, but instances vary in size (nodes, arcs, commodities), as well as in the ratios of fixed to variable costs, and total demand to capacity. The Appendix details the characteristics of these instance sets. In summary, the names of the C instances indicate the numbers of nodes, arcs, and commodities, respectively, while the letter F or V points to a high or low cost ratio, respectively, and the letter T or L points to tight or loose capacity ratio, respectively. Similar information is given for R instances, the number indicating size (from 20 nodes, 220 arcs and 40 commodities to 20, 515 and 200, respectively), the F values standing for increasing levels of cost ratios, and C values for increasingly tight capacity relative to total demand.

The algorithm is coded in C++ using Microsoft Visual Studio 2008. CPLEX 12.1 was used to solve the flow problems and the restricted MIPs. All computing times reported in this section were obtained from computers with the AMD Dual-Core Opteron 64-bit microprocessor with 2.4 Ghz and 16 GB Ram, under the Linux operating system.

We first discuss the calibration of the algorithm parameters, followed by an analysis of the behavior of each phase and, finally, by a comparative analysis of its performance in relation to results in the literature. Note that, the initial solutions for our search algorithm were obtained using CPLEX running for a maximum of 30 minutes on the CMND relaxation of the problem.

4.5.1 Parameter Calibration

The calibration is based on the same ten instances used by Pedersen *et al.* (2009). These instances display various combinations of characteristics and are representative of the overall set of instances. They include C and R instances. For convenience, these instances are listed in Table 4.1.

We calibrated the five parameters shown in Table 4.2. Each parameter was tested with two

Instance	Nodes	Arcs	Commodities	Capacity Ratio	Cost Ratio
R10,F05,C2	20	120	40	Medium	Medium
R12,F10,C2	20	120	200	Medium	High
R13,F01,C8	20	220	40	Tight	Low
R15,F10,C8	20	220	200	Tight	High
C20,230,200,F,L	20	230	200	Loose	High
R16,F10,C1	20	314	40	Loose	High
R17,F01,C1	20	318	100	Loose	Low
R18,F05,C2	20	315	200	Medium	Medium
C30,520,100,V,T	30	519	100	Tight	Low
C100,400,30,F,L	100	400	30	Loose	High

Table 4.1: Problem instances used for calibration

values, indicated in the "Values Tested" column, inspired principally by those used in Pedersen *et al.* (2009), to which we compare. We tested the 32 different combinations of values for the 5 parameters above, for the 10 instances in Table 4.2. Column "Value selected" displays the selected values, which were then used for the complete set of test instances.

Parameter	Values Tested	Value Selected
Length of tabu list	10, 15	10
Length of reference set	10, 20	20
Penalty scaling factor	0.2, 0.5	0.2
Intensification open parameter	0.8, 0.9	0.8
Intensification closed parameter	0.1, 0.2	0.2

Table 4.2: Calibrated parameters

To make the selection, we computed for each combination the average gap and the average execution time. Table 4.3A shows the combinations corresponding to the top ten best average gaps with their corresponding average execution times. The difference between the best and the worst among the ten top average gaps is about 0.5%, indicating that these combinations could provide about the same average solution quality. We have thus assigned a score to each combination of values, a score of 10 being assigned to the best combination, 9 to the second best, etc. Table 4.3B scores each parameter value by summing the score of each combination where this value appears.

Figures in Table 4.3B show that the values in the combination (10,20,0.2,0.9,0.1) (same order of parameters as in Table 4.2) have the best scores. This combination provides solutions with 3.1% average gap. Taking the execution time into account, however, the combination (10,20,0.2,0.8,0.2) is better because it provides similar average results (3.2% gap) with an

Combination	Gap	Time
Sorted by average gap		
10,10,0.2,0.8,0.1	2.8%	2770
10,10,0.2,0.9,0.1	3.0%	2673
15,20,0.5,0.9,0.2	3.0%	3611
10,20,0.2,0.9,0.2	3.1%	2759
15,20,0.5,0.9,0.1	3.1%	3655
10,20,0.2,0.9,0.1	3.2%	2679
10,20,0.2,0.8,0.2	3.3%	2513
10,20,0.2,0.8,0.1	3.3%	3085
10,20,0.5,0.8,0.1	3.3%	3139
10,20,0.5,0.9,0.1	3.3%	3378
Table 4.3A		

Parameter	Value	Score
Length of tabu search	10	46
	15	13
Length of reference set	10	19
	20	36
Penalty value	0.2	38
	0.5	17
Intensification open	0.8	19
	0.9	36
Intensification close	0.1	36
	0.2	19
Table 4.3B		

Table 4.3: Ten top average results sorted by gap (A) and the score of each parameter (B)

average running time that is the best among all the combinations in the top-ten list. These two combinations have the same values for the length of the tabu search, the length of the reference set and the penalty, which are the main parameters of the algorithm. But the combination (0.8,0.2) can fix more arcs than the combination (0.9,0.1) during the intensification phase, which reduces the time for intensification while maintaining the solution quality. We used this latter combination in all our experiments.

4.5.2 Internal Performance Analysis

Tables 4.4 and 4.5 summarize our results for the 78 problem instances. Each instance was solved ten times with the same parameter values to account for stochastic variations in the algorithm. Each table is divided into three sections labeled “TS”, “PR”, and “Intensification” for the three phases of the algorithm, the tabu search, path relinking, and intensification phases, respectively. The columns labeled “T” give the average computational time in minutes. The maximum running time for the tabu search phase was 1 hour, for the path relinking was 2 hours, and 1 hour for each restricted DBCMND of the intensification phase. The total maximum CPU running time for the complete algorithm was thus 5 hours. These numbers were chosen so that the results are easy to compare with those of other methods. The columns labeled “S” give the average number of feasible solutions found, while the columns labeled “Best” give the best solutions obtained over the ten runs. Table 4.6 reports the average standard deviation over these repetitions. The values are very low, which underlines the robustness of the algorithm we

propose.

Instance	TS			PR			Intensification	
	T	S	Best	T	S	Best	T	Best
C20,230,200,V,L	60	10	100,708	8	85	101,640	23	97,274
C20,230,200,F,L	60	6	145,958	5	40	148,051	27	139,395
C20,230,200,V,T	60	10	102,906	11	43	104,374	7	100,720
C20,230,200,F,T	60	7	142,038	6	33	151,571	18	138,962
C20,300,200,V,L	60	15	80,166	31	95	80,162	71	77,584
C20,300,200,F,L	60	9	124,846	5	47	127,816	45	119,987
C20,300,200,V,T	60	13	77,692	23	68	78,384	7	76,450
C20,300,200,F,T	60	11	119,384	19	88	123,211	62	111,776
C30,520,100,V,L	60	29	55,002	21	185	55,117	4	54,783
C30,520,100,F,L	60	21	102,735	41	187	103,644	17	100,098
C30,520,100,V,T	60	16	53,313	42	166	53,516	1	53,035
C30,520,100,F,T	60	13	102,484	50	109	105,938	55	101,412
C30,520,400,V,L	60	17	118,627	105	157	117,824	56	115,528
C30,520,400,F,L	60	22	155,270	67	110	203,505	19	153,409
C30,520,400,V,T	60	19	119,795	91	127	118,965	59	117,226
C30,520,400,F,T	60	26	156,694	15	123	214,384	61	155,906
C30,700,100,V,L	60	25	48,879	13	91	49,185	1	48,807
C30,700,100,F,L	60	23	61,745	50	146	62,838	43	61,408
C30,700,100,V,T	60	24	47,141	49	155	47,032	26	46,812
C30,700,100,F,T	60	29	56,810	71	225	57,058	21	56,237
C30,700,400,V,L	60	35	101,423	53	125	139,217	23	100,589
C30,700,400,F,T	60	21	141,037	5	185	142,046	15	141,037
C30,700,400,V,T	60	14	108,481	100	116	108,710	62	97,875
C30,700,400,F,T	60	13	134,320	3	60	134,707	67	133,686

Table 4.4: Computational results on C instances

We notice that the restricted path relinking phase achieves its stated goal of enlarging the set of feasible solutions obtained during the tabu search phase (and not of improving the best solution found by the tabu search). It identified many new feasible solutions, at a small computational cost and without loosing too much on solution quality. Tables 4.4 and 4.5 indicate that, on average, path relinking generates 5 times the number of new feasible solutions obtained by the tabu search. Moreover, as indicated by the results displayed in Table 4.7, path reliking contributes significantly to the overall performance of the algorithm. Thus, the solutions obtained in the third phase without path relinking present a 3.68% average gap for R instances and 3.06% average gap for C instances. With path relinking, the solutions obtained at the end of the third phase display a 1.61% average gap for R instances and 1.82% average gap for C instances, a significant improvement.

Instance	TS			PR			Intensification	
	T	S	Best	T	S	Best	T	Best
R13,F01,C1	60	16	148,425	1	78	150,511	1	147,349
R13,F05,C1	60	19	289,535	2	113	293,431	1	277,891
R13,F10,C1	60	15	403,490	2	115	419,908	1	385,396
R13,F01,C2	60	18	157,743	1	80	158,392	1	155,887
R13,F05,C2	60	18	306,783	2	128	311,442	1	295,655
R13,F10,C2	60	13	456,269	1	93	467,241	2	436,773
R13,F01,C8	60	18	219,105	1	82	223,460	1	218,787
R13,F05,C8	60	11	500,403	2	98	509,431	9	492,959
R13,F10,C8	60	18	813,589	1	136	831,592	2	789,641
R14,F01,C1	60	23	437,836	8	140	440,335	1	424,039
R14,F05,C1	60	16	853,765	4	121	861,162	1	784,626
R14,F10,C1	60	6	1,214,782	2	28	1,242,096	11	1,131,900
R14,F01,C2	60	14	455,609	4	81	458,743	1	454,031
R14,F05,C2	60	13	914,839	7	109	931,059	28	883,051
R14,F10,C2	60	9	1,378,765	3	57	1,415,336	22	1,308,030
R14,F01,C8	60	16	714,841	7	71	717,294	40	703,259
R14,F05,C8	60	9	1,743,116	2	39	1,831,803	46	1,695,160
R14,F10,C8	60	10	2,874,717	15	126	2,940,505	94	2,757,660
R15,F01,C1	60	6	1,033,662	5	31	1,038,819	2	1,019,390
R15,F05,C1	60	9	2,156,433	7	60	2,199,282	47	2,017,150
R15,F10,C1	60	4	3,362,675	4	18	3,673,775	61	2,985,570
R15,F01,C2	60	12	1,177,198	15	66	1,180,909	12	1,174,520
R15,F05,C2	60	3	2,785,075	8	26	2,857,848	78	2,571,880
R15,F10,C2	60	5	4,430,712	8	44	4,528,368	102	4,017,230
R15,F01,C8	60	4	2,408,210	18	10	2,408,210	39	2,408,210
R15,F05,C8	60	7	5,874,704	24	29	5,916,736	65	5,796,510
R15,F10,C8	60	7	9,205,777	21	28	9,306,490	4	9,129,360
R16,F01,C1	60	14	141,352	1	54	143,649	1	140,082
R16,F05,C1	60	23	260,685	2	96	272,657	1	248,703
R16,F10,C1	60	11	371,854	2	81	395,754	1	345,243
R16,F01,C2	60	22	143,345	3	103	145,190	1	142,605
R16,F05,C2	60	14	264,345	2	84	282,548	1	261,937
R16,F10,C2	60	12	375,258	3	105	388,991	4	363,999
R16,F01,C8	60	31	181,350	5	146	186,039	1	180,132
R16,F05,C8	60	15	398,082	2	119	407,366	2	391,796
R16,F10,C8	60	18	629,107	2	142	643,342	11	604,430
R17,F01,C1	60	15	372,914	10	83	381,485	1	366,492
R17,F05,C1	60	13	695,550	13	121	737,784	60	676,528
R17,F10,C1	60	13	1,037,684	12	123	1,085,083	59	953,009
R17,F01,C2	60	26	385,225	19	129	389,175	1	383,871
R17,F05,C2	60	11	782,334	5	68	816,318	1	741,136
R17,F10,C2	60	15	1,142,773	7	118	1,242,485	23	1,092,510
R17,F01,C8	60	25	535,475	38	178	551,247	17	530,366
R17,F05,C8	60	9	1,299,937	11	70	1,326,098	116	1,231,700
R17,F10,C8	60	7	2,223,244	5	61	2,308,072	72	1,999,950
R18,F01,C1	60	20	866,802	70	148	878,193	6	844,260
R18,F05,C1	60	7	1,609,622	19	60	1,709,883	17	1,575,715
R18,F10,C1	60	9	2,380,708	10	38	2,544,101	2	2,240,660
R18,F01,C2	60	11	966,838	51	129	974,302	108	941,763
R18,F05,C2	60	3	1,955,396	4	18	2,003,180	27	1,883,870
R18,F10,C2	60	4	3,157,951	5	26	3,297,777	75	2,792,870
R18,F01,C8	60	5	1,572,109	12	26	1,585,390	55	1,540,690
R18,F05,C8	60	5	4,312,752	7	21	4,572,846	61	4,039,410
R18,F10,C8	60	3	6,899,618	4	7	7,007,133	4	6,608,210

Table 4.5: Computational results on R instances

Instances	TS	PR	Intensification
R	.0090	.0201	.0052
C	.0114	.0061	.0035

Table 4.6: Average ratios of standard deviations to mean

Instances	Without PR	With PR
R	3.68%	1.61%
C	3.06%	1.82%

Table 4.7: Average gap with and without the path relinking phase

The results in Table 4.8 focus the analysis on the intensification phase by reporting the average relative difference between the optimal solutions found during intensification to the restricted DBCMND problems compared with the best solution found by the tabu search. Columns "Open/TS" and "Close/TS" report these figures for the two cases, i.e., when the intensification is performed by fixing arcs to open and closed, respectively. The figures indicate that improved solutions are found during intensification, which emphasizes the importance of the procedure within the proposed algorithm. We notice that the second case, fixing arcs to closed status, finds better solutions. This is because arcs that are open during the first intensification case are not closed during the second one even when they have a small $rate_{ij}$ value.

Instances	Open/TS	Close/TS
54 R	-1.86%	-3.94%
24 C	-0.98%	-2.26%

Table 4.8: Intensification improvement relative to the tabu search

4.5.3 Comparative Analysis

We compare the results of the meta-heuristic we propose and those from the state-of-the-art tabu search algorithm of Pedersen *et al.* (2009) obtained after 1 hour of computation, as well as those of the MIP algorithm of CPLEX 12.1 obtained after 1 and 5 hours, and with the lower bound obtained after 10 hours. Table 4.9 provides a general view of the effectiveness of TS-PR by displaying the average improvement gap (negative values indicate better results) and number of improved solutions, respectively, obtained by TS-PR with respect to those of other methods. Columns "TS-PR/P-TS", "TS-PR/CPLEX 1 h" and "TS-PR/CPLEX 5 h" report these measures relative to the tabu search algorithm of Pedersen *et al.* (2009) and CPLEX after

1 and 5 hours, respectively, while Column "TS-PR/LB.CPLEX" reports the average gap with respect to the best CPLEX lower bound obtained after 10 hours. Table 4.9 shows that TS-PR improves on CPLEX run for 1 hour, identifying 34 equal or better solutions for the R instances and 19 better solutions for the C instances. Allowing CPLEX 5 hours of computing times improves slightly the solution for a few instances, but still cannot find feasible solutions for two C instances, while the proposed method identifies 22 better solutions for R instances and 7 for C instances. The excellent performance of the proposed TS-PR meta-heuristic is further underlined by the low gaps with the lower bound identified by CPLEX after 10 hours, which indicates that TS-PR reaches excellent-quality solutions leaving little to be further improved.

Instances	TS-PR/CPLEX 1h		TS-PR/CPLEX 5h		TS-PR/B. CPLEX
	Gap	No.Sol.	Gap	No.Sol.	
54 R	-0.70%	34	0.10%	22	1.61%
24 C	-0.71%	19	0.12%	7	1.82%

Table 4.9: Comparative performance measures of TS-PR versus CPLEX

We also compared the performance of the algorithm we propose to that of Pedersen *et al.* (2009). The figures in Table 4.10 show that TS-PR outperforms the state-of-the-art tabu search, identifying better solutions for all problem instances with an average improvement of 5.94% for R instances and 4.71% for C instances. Table 4.10 also displays the performance of the tabu search phase of TS-PR. The figures clearly show that the proposed tabu search procedure outperforms the one in Pedersen *et al.* (2009) by identifying 43 improved solutions for R instances and 22 for C instances. The improvements are significant, of more than 2% on average.

Instances	TS-PR/P-TS		TS/P-TS	
	Gap	No.Sol.	Gap	No.Sol.
54 R	-5.94%	54	-2.05%	43
24 C	-4.71%	24	-2.45%	22

Table 4.10: Comparison of TS-PR and tabu search in TS-PR with Pedersen et al. (2009)

We complete this analysis by comparing the two restricted DBCMND problems derived during the intensification phase with solutions obtained by CPLEX. Column "Open/CPLEX" in Table 4.11 compares the number of arcs fixed to open in the first restricted problem with the number of open arcs in solutions found by CPLEX 12.1 after 5 hours. These ratios are high indicating that the solver needs to add a few arcs only to satisfy all the constraints of the DBCMND instances. Column "Closed/Arcs" gives the average ratio of the number of arcs fixed to closed for the second restricted formulation to the total number of arcs in the instance.

As can be seen, the “close" fixing scheme closes many arcs, which substantially reduces the size of the search space explored by CPLEX. Column “(Closed+O.CPLEX)/Arcs” gives the average ratio of the number of arcs fixed to closed in the second restricted formulation plus the arcs that are open in the CPLEX solution, to the total number of arcs in the instance. The ratios indicate that the intensification procedure is successful in reducing the search space, so that the successive search performed by CPLEX is computationally effective.

Instances	Open/CPLEX	Close/Arcs	(Closed+O.CPLEX)/Arcs
54 R	80%	61%	86%
24 C	82%	80%	94%

Table 4.11: Ratios of fixed arcs by the intensification phase versus CPLEX

4.6 Conclusions

We have proposed a matheuristic solution framework that combines tabu search and path relinking for the DBCMND problem. We considered a total of 78 instances. The tabu search phase is competitive with the current state-of-the-art tabu search algorithm, obtaining 65 improved solutions. We introduced a cycle-based neighbourhood structure and a minimum-cost maximum-flow model to satisfy the design-balance constraints for the DBCMND. With the help of the tabu search and restricted path relinking phases, the proposed matheuristic found 78 improved solutions compared to those of the state-of-the-art tabu search, and 63 improved solutions compared to those of CPLEX 12.1 (1 hour). The results are also competitive with those found by CPLEX 12.1 (5 hours) with a small average difference and 29 improved solutions with less computational time on average.

Several directions could be investigated to improve the performance of our algorithm. An adaptive guiding scheme could force the tabu search to escape from local optima or to concentrate on the region close to the current solution. We could also extend the algorithm to sparse graphs in which it is not easy to find feasible solutions because of the lack of arcs to add to the candidate solutions.

Acknowledgments

While working on this project, T.G. Crainic was the NSERC Industrial Research Chair in Logistics Management, ESG UQAM. and Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal, and the Department of Economics and Business Administration, Molde University College, Norway, while M. Toulouse was Adjunct Professor with the Department of Computer Science and Operations Research, Université de Montréal.

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Industrial Research Chair and Discovery Grant programs, and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT).

Appendix

Table 4.12 explains the column headers used in the subsequent tables.

Column	Meaning
Instances	Instances used for experiments
TS-PR	Best solutions found by the TS-PR matheuristic, which includes tabu search, path relinking, and intensification phases
P-TS	Best solutions found by Pedersen *et al.* (2009)
TS	Best objective values found by proposed tabu search
PR	Best objective values found by proposed path relinking
MP	Best objective values found by Express-MP in 1 hour Pedersen *et al.* (2009)
CPLEX 1 h	Best solutions found by CPLEX 12.1 in 1 hour
CPLEX 5 h	Best solutions found by CPLEX 12.1 in 5 hours
B. CPLEX	Best lower bound found by CPLEX 12.1 in 10 hours
X/Y	Comparison of results found by procedure X in percentage improvement with respect to those of procedure Y
Nodes	Number of nodes in each instance
Arcs	Number of arcs in each instance
Com.	Number of commodities in each instance
O. Int.	Average number of arcs fixed to open during intensification
C. Int.	Average number of arcs fixed to closed during intensification
O. CPLEX	Number of open arcs in best solutions found by CPLEX after 5 hours
Int. Open	Best value found by fixing-open intensification scheme
Int. Close	Best value found by fixing-close intensification scheme
Total System Imbalance	Total system imbalance of solution found by CPLEX
Running Time	Running time for each instance
Time CPLEX 5 h	CPLEX CPU time (min); The 300 figure indicates that CPLEX cannot find the optimal solution in 5 hours
No. TS	Average number of solutions found by the tabu search phase
No. PR	Average number of solutions found by the path relinking phase

Table 4.12: Notation used in tables of the Appendix

Table 4.13 lists the solutions for the instances used in the calibration phase.

Tables 4.14 and 4.15 display the characteristics of the R and C instances in terms of number of nodes, arcs, and commodities, as well as cost and capacity ratios Ghamlouche *et al.* (see, e.g., 2003, for more detailed information). For the C instances, a high or low fixed cost relative

Instance	Bound	P-TS	TS	TS-PR
R10,F05,C2	436,073	443,547	445,383	439,244
R12,F10,C2	7,408,996	7,530,870	7,467,136	7,436,420
R13,F01,C8	218,787	223,231	219,105	218,787
R15,F10,C8	9,105,010	9,366,760	9,166,884	9,105,010
C20,230,200,F,L	128,014	146,643	146,445	139,365
R16,F10,C1	309,383	352,681	345,198	340,641
R17,F01,C1	364,784	365,801	366,914	364,995
R18,F05,C1	1,362,596	1,597,610	1,601,402	1,582,820
C30,520,100,V,T	52,622	53,972	53,144	53,032
C100,400,30,F,L	58,316	67,603	69,658	67,103

Table 4.13: Best solutions for instances used in calibration

to the routing cost is signaled by the letter F or V, respectively, while the letters T and L indicate, respectively, if the problem has a tight or loose capacity given the total demand. For the R instances, the fixed cost ratio is computed as $|\mathcal{P}|\sum_{(i,j)\in\mathcal{A}} f_{ij}/\sum_{p\in\mathcal{P}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^{p}$, and the three values considered are F01 = 0.01, F05 = 0.05, and F10 = 0.10 corresponding to increasing levels of fixed costs compared to routing costs. The capacity ratio is computed as $|\mathcal{A}|\sum_{p\in\mathcal{P}} w^{p}/\sum_{(i,j)\in\mathcal{A}} u_{ij}$, and the values considered are C1 = 1, C2 = 2, and C8 = 8, indicating that the total capacity becomes increasingly tight relative to the total demand.

Instance	Nodes	Arcs	Commodities
R13	20	220	40
R14	20	220	100
R15	20	220	200
R16	20	314	40
R17	20	318	100
R18	20	315	200

Table 4.14: Characteristics of R instances

Tables 4.16 and 4.17 display the objective values for each method and the gaps between the proposed algorithm and the other methods for the C instances. Tables 4.18 and 4.19 display the same information for the R instances. Negative gaps indicate the proposed method improves over previous ones; some of these values (Columns "TS-PR/P-TS") are considerable. The values in Columns "TS-PR/CPLEX 1 h" and "TS-PR/CPLEX 5 h" indicate that our algorithm is also competitive with the state-of-the-art solver. The maximum difference between the results of the proposed algorithm and CPLEX 5 h is less than 1.5%, and many values are less than 0.5%. This quality is obtained in less computational time than the MIP solver.

Instance	Nodes	Arcs	Commodities
C20,230,200,V,L	20	230	200
C20,230,200,F,L	20	230	200
C20,230,200,V,T	20	230	200
C20,230,200,F,T	20	230	200
C20,300,200,V,L	20	300	200
C20,300,200,F,L	20	300	200
C20,300,200,V,T	20	300	200
C20,300,200,F,T	20	300	200
C30,520,100,V,L	30	520	100
C30,520,100,F,L	30	520	100
C30,520,100,V,T	30	520	100
C30,520,100,F,T	30	520	100
C30,520,400,V,L	30	520	400
C30,520,400,F,L	30	520	400
C30,520,400,V,T	30	520	400
C30,520,400,F,T	30	520	400
C30,700,100,V,L	30	700	100
C30,700,100,F,L	30	700	100
C30,700,100,V,T	30	700	100
C30,700,100,F,T	30	700	100
C30,700,400,V,L	30	700	400
C30,700,400,F,T	30	700	400
C30,700,400,V,T	30	700	400
C30,700,400,F,T	30	700	400

Table 4.15: Characteristics of C instances

Instance	P-TS	CPLEX 1 h	CPLEX 5 h	B. CPLEX	TS-PR
C20,230,200,V,L	102,919	98,976	97,847	96,038	97,274
C20,230,200,F,L	150,764	141,689	140,843	136,276	139,395
C20,230,200,V,T	103,371	101,696	100,558	100,209	100,720
C20,230,200,F,T	149,942	141,671	140,108	136,204	138,962
C20,300,200,V,L	82,533	78,168	77,742	76,459	77,584
C20,300,200,F,L	128,757	122,164	119,823	116,773	119,987
C20,300,200,V,T	78,571	76,602	76,208	76,190	76,450
C20,300,200,F,T	116,338	114,816	111,475	109,012	111,776
C30,520,100,V,L	55,981	54,683	54,683	54,677	54,783
C30,520,100,F,L	104,533	101,346	99,900	95,090	100,098
C30,520,100,V,T	54,493	53,041	53,023	52,998	53,035
C30,520,100,F,T	105,167	102,090	101,271	98,653	101,412
C30,520,400,V,L	119,735	115,167	114,646	113,769	115,528
C30,520,400,F,L	162,360	153,311	153,311	150,144	153,409
C30,520,400,V,T	120,421	n/a	n/a	116,111	117,226
C30,520,400,F,T	161,978	n/a	n/a	152,725	155,906
C30,700,100,V,L	49,902	48,855	48,693	48,688	48,807
C30,700,100,F,L	63,889	61,846	61,362	60,236	61,408
C30,700,100,V,T	48,202	46,792	46,750	46,582	46,812
C30,700,100,F,T	58,204	56,251	56,287	55,732	56,237
C30,700,400,V,L	103,932	101,237	99,905	98,323	100,589
C30,700,400,F,T	157,043	n/a	139,495	133,762	141,037
C30,700,400,V,T	103,085	n/a	97,800	96,013	97,875
C30,700,400,F,T	141,917	133,194	133,194	130,066	133,686

Table 4.16: Best solutions for C instances

Instance	TS-PR/ P-TS	TS-PR/ CPLEX 1 h	TS-PR/ CPLEX 5 h	TS-PR/ B.CPLEX
C20,230,200,V,L	-5.80%	-1.75%	-0.59%	1.27%
C20,230,200,F,L	-8.16%	-1.65%	-1.04%	2.24%
C20,230,200,V,T	-2.63%	-0.97%	0.16%	0.51%
C20,230,200,F,T	-7.90%	-1.95%	-0.82%	1.98%
C20,300,200,V,L	-6.38%	-0.75%	-0.20%	1.45%
C20,300,200,F,L	-7.31%	-1.81%	0.14%	2.68%
C20,300,200,V,T	-2.77%	-0.20%	0.32%	0.34%
C20,300,200,F,T	-4.08%	-2.72%	0.27%	2.47%
C30,520,100,V,L	-2.19%	0.18%	0.18%	0.19%
C30,520,100,F,L	-4.43%	-1.25%	0.20%	5.00%
C30,520,100,V,T	-2.75%	-0.01%	0.02%	0.07%
C30,520,100,F,T	-3.70%	-0.67%	0.14%	2.72%
C30,520,400,V,L	-3.64%	0.31%	0.76%	1.52%
C30,520,400,F,L	-5.83%	0.06%	0.06%	2.13%
C30,520,400,V,T	-2.73%	n/a	n/a	0.95%
C30,520,400,F,T	-3.89%	n/a	n/a	2.04%
C30,700,100,V,L	-2.24%	-0.10%	0.23%	0.24%
C30,700,100,F,L	-4.04%	-0.71%	0.07%	1.91%
C30,700,100,V,T	-2.97%	0.04%	0.13%	0.49%
C30,700,100,F,T	-3.50%	-0.02%	-0.09%	0.90%
C30,700,400,V,L	-3.32%	-0.64%	0.68%	2.25%
C30,700,400,F,T	-11.35%	n/a	1.09%	5.16%
C30,700,400,V,T	-5.32%	n/a	0.08%	1.90%
C30,700,400,F,T	-4.71%	-0.71%	0.10%	1.82%

Table 4.17: TS-PR improvement with respect to other methods for C instances

Instance	P-TS	CPLEX 1 h	CPLEX 5 h	B.CPLEX	TS-PR
R13,F01,C1	147,837	147,349	147,349	147,349	147,349
R13,F05,C1	281,668	277,891	277,891	277,891	279,389
R13,F10,C1	404,434	385,396	385,396	385,396	385,396
R13,F01,C2	159,852	155,887	155,887	155,887	156,616
R13,F05,C2	311,209	295,180	295,180	295,180	295,180
R13,F10,C2	470,034	444,545	433,117	431,140	434,383
R13,F01,C8	225,339	218,787	218,787	218,787	218,787
R13,F05,C8	512,027	491,603	491,560	486,754	492,959
R13,F10,C8	875,984	789,479	782,049	772,790	791,213
R14,F01,C1	431,562	422,709	422,709	422,709	422,709
R14,F05,C1	811,102	807,553	784,884	784,626	784,626
R14,F10,C1	1,193,950	1,199,250	1,132,900	1,094,083	1,137,820
R14,F01,C2	465,762	452,591	452,591	452,591	453,434
R14,F05,C2	942,678	892,534	884,234	875,645	891,138
R14,F10,C2	1,401,880	1,320,570	1,314,460	1,256,060	1,307,770
R14,F01,C8	720,882	702,781	702,614	702,614	702,614
R14,F05,C8	1,795,650	1,747,030	1,691,770	1,671,457	1,693,240
R14,F10,C8	2,997,290	2,767,180	2,758,650	2,735,090	2,769,360
R15,F01,C1	1,039,440	1,017,740	1,017,740	1,017,740	1,017,740
R15,F05,C1	2,170,310	2,011,860	2,011,860	1,976,249	2,055,803
R15,F10,C1	3,194,270	3,011,740	2,980,870	2,850,055	2,971,500
R15,F01,C2	1,205,790	1,176,050	1,174,960	1,174,517	1,174,520
R15,F05,C2	2,698,680	2,607,500	2,564,840	2,508,981	2,561,060
R15,F10,C2	4,447,950	4,422,350	4,030,490	3,887,960	4,045,030
R15,F01,C8	2,472,860	2,401,180	2,401,115	2,401,110	2,408,210
R15,F05,C8	6,067,350	5,795,320	5,795,320	5,795,320	5,796,510
R15,F10,C8	10,263,600	9,105,010	9,105,010	9,105,010	9,129,360
R16,F01,C1	142,692	140,082	140,082	140,082	140,082
R16,F05,C1	261,755	248,703	248,703	248,703	248,703
R16,F10,C1	374,819	344,446	340,641	340,641	350,958
R16,F01,C2	145,266	142,381	142,381	142,381	142,605
R16,F05,C2	277,307	260,993	259,313	259,313	260,822
R16,F10,C2	391,386	361,626	365,001	361,626	368,572
R16,F01,C8	187,176	179,639	179,639	179,639	180,228
R16,F05,C8	423,320	391,101	390,549	380,855	388,180
R16,F10,C8	649,121	599,456	596,660	583,389	598,835
R17,F01,C1	374,016	364,784	364,784	364,784	365,788
R17,F05,C1	718,135	686,722	686,234	660,755	676,528
R17,F10,C1	1,041,450	989,809	968,207	908,867	966,116
R17,F01,C2	393,608	382,593	382,593	382,593	384,579
R17,F05,C2	786,198	755,416	741,984	734,117	741,744
R17,F10,C2	1,162,290	1,113,030	1,086,710	1,034,154	1,086,640
R17,F01,C8	539,817	530,435	529,350	525,189	529,876
R17,F05,C8	1,348,750	1,232,750	1,224,770	1,204,567	1,230,910
R17,F10,C8	2,227,780	2,043,920	2,008,220	1,958,676	1,999,950
R18,F01,C1	864,425	845,718	846,857	842,109	844,260
R18,F05,C1	1,640,200	1,606,880	1,597,740	1,556,693	1,588,890
R18,F10,C1	2,399,230	2,359,170	2,233,090	2,149,024	2,264,470
R18,F01,C2	962,402	942,884	941,635	934,187	944,708
R18,F05,C2	1,958,150	1,908,250	1,885,570	1,833,797	1,883,870
R18,F10,C2	2,986,000	2,813,750	2,813,750	2,709,098	2,806,020
R18,F01,C8	1,617,320	1,563,820	1,536,600	1,520,032	1,542,500
R18,F05,C8	4,268,580	4,039,340	3,993,820	3,934,504	4,039,410
R18,F10,C8	7,440,780	6,639,848	6,578,230	6,503,621	6,603,500

Table 4.18: Best solutions for R instances

Instance	TS-PR/ P-TS	TS-PR/ CPLEX 1 h	TS-PR/ CPLEX 5 h	TS-PR/ B.CPLEX
R13,F01,C1	-0.33%	0.00%	0.00%	0.00%
R13,F05,C1	-1.36%	0.00%	0.00%	0.00%
R13,F10,C1	-4.94%	0.00%	0.00%	0.00%
R13,F01,C2	-2.54%	0.00%	0.00%	0.00%
R13,F05,C2	-5.26%	0.16%	0.16%	0.16%
R13,F10,C2	-7.62%	-1.78%	0.84%	1.29%
R13,F01,C8	-2.99%	0.00%	0.00%	0.00%
R13,F05,C8	-3.87%	0.28%	0.28%	1.26%
R13,F10,C8	-10.93%	0.02%	0.96%	2.13%
R14,F01,C1	-1.77%	0.31%	0.31%	0.31%
R14,F05,C1	-3.37%	-2.92%	-0.03%	0.00%
R14,F10,C1	-5.48%	-5.95%	-0.09%	3.34%
R14,F01,C2	-2.58%	0.32%	0.32%	0.32%
R14,F05,C2	-6.75%	-1.07%	-0.13%	0.84%
R14,F10,C2	-7.17%	-0.96%	-0.49%	3.97%
R14,F01,C8	-2.51%	0.07%	0.09%	0.09%
R14,F05,C8	-5.93%	-3.06%	0.20%	1.40%
R14,F10,C8	-8.69%	-0.35%	-0.04%	0.82%
R15,F01,C1	-1.97%	0.16%	0.16%	0.16%
R15,F05,C1	-7.59%	0.26%	0.26%	2.03%
R15,F10,C1	-6.99%	-0.88%	0.16%	4.54%
R15,F01,C2	-2.66%	-0.13%	-0.04%	0.00%
R15,F05,C2	-4.93%	-1.38%	0.27%	2.45%
R15,F10,C2	-10.72%	-10.08%	-0.33%	3.22%
R15,F01,C8	-2.68%	0.29%	0.29%	0.29%
R15,F05,C8	-4.67%	0.02%	0.02%	0.02%
R15,F10,C8	-12.42%	0.27%	0.27%	0.27%
R16,F01,C1	-1.86%	0.00%	0.00%	0.00%
R16,F05,C1	-5.25%	0.00%	0.00%	0.00%
R16,F10,C1	-8.57%	0.23%	1.33%	1.33%
R16,F01,C2	-1.87%	0.16%	0.16%	0.16%
R16,F05,C2	-5.87%	0.36%	1.00%	1.00%
R16,F10,C2	-7.52%	-0.28%	0.65%	0.65%
R16,F01,C8	-3.91%	0.27%	0.27%	0.27%
R16,F05,C8	-8.05%	0.18%	0.32%	2.79%
R16,F10,C8	-7.39%	0.82%	1.29%	3.48%
R17,F01,C1	-2.05%	0.47%	0.47%	0.47%
R17,F05,C1	-6.15%	-1.51%	-1.43%	2.33%
R17,F10,C1	-9.28%	-3.86%	-1.59%	4.63%
R17,F01,C2	-2.54%	0.33%	0.33%	0.33%
R17,F05,C2	-6.08%	-1.93%	-0.11%	0.95%
R17,F10,C2	-6.39%	-1.88%	0.53%	5.34%
R17,F01,C8	-1.78%	-0.01%	0.19%	0.98%
R17,F05,C8	-9.50%	-0.09%	0.56%	2.20%
R17,F10,C8	-11.39%	-2.20%	-0.41%	2.06%
R18,F01,C1	-2.39%	-0.17%	-0.31%	0.25%
R18,F05,C1	-4.09%	-1.98%	-1.40%	1.21%
R18,F10,C1	-7.08%	-5.29%	0.34%	4.09%
R18,F01,C2	-2.19%	-0.12%	0.01%	0.80%
R18,F05,C2	-3.94%	-1.29%	-0.09%	2.66%
R18,F10,C2	-6.92%	-0.75%	-0.75%	3.00%
R18,F01,C8	-4.97%	-1.50%	0.27%	1.34%
R18,F05,C8	-5.67%	0.00%	1.13%	2.60%
R18,F10,C8	-12.60%	-0.48%	0.45%	1.58%

Table 4.19: TS-PR improvement with respect to other methods for R instances

Tables 4.20 and 4.21 compare the performance of the proposed tabu search to that of the other methods. The tabu search phase finds feasible solutions for all instances, and improves over the state-of-the-art tabu search method 22 (of 24) C instances and 47 (of 54) R instances. The largest improvements are obtained for instances with high cost ratios ("F10" for R instances and "F" for C instances) because the feasibility phase always adds the arcs with the smallest total cost when satisfying the design-balance constraints, which has a more important impact when fixed costs are high.

Instance	P-TS	CPLEX 1 h	TS	TS/P-TS	TS/CPLEX 1 h
C20,230,200,V,L	102,919	98,976	100,708	-2.20%	1.72%
C20,230,200,F,L	150,764	141,689	145,958	-3.29%	2.92%
C20,230,200,V,T	103,371	101,696	102,906	-0.45%	1.18%
C20,230,200,F,T	149,942	141,671	142,038	-5.56%	0.26%
C20,300,200,V,L	82,533	78,168	80,166	-2.95%	2.49%
C20,300,200,F,L	128,757	122,164	124,846	-3.13%	2.15%
C20,300,200,V,T	78,571	76,602	77,692	-1.13%	1.40%
C20,300,200,F,T	116,338	114,816	119,384	2.55%	3.83%
C30,520,100,V,L	55,981	54,683	55,002	-1.78%	0.58%
C30,520,100,F,L	104,533	101,346	102,735	-1.75%	1.35%
C30,520,100,V,T	54,493	53,041	53,313	-2.21%	0.51%
C30,520,100,F,T	105,167	102,090	102,484	-2.62%	0.38%
C30,520,400,V,L	119,735	115,167	118,627	-0.93%	2.92%
C30,520,400,F,L	162,360	153,311	155,270	-4.57%	1.26%
C30,520,400,V,T	120,421	n/a	119,795	-0.52%	n/a
C30,520,400,F,T	161,978	n/a	156,694	-3.37%	n/a
C30,700,100,V,L	49,902	48,855	48,879	-2.09%	0.05%
C30,700,100,F,L	63,889	61,846	61,745	-3.47%	-0.16%
C30,700,100,V,T	48,202	46,792	47,141	-2.25%	0.74%
C30,700,100,F,T	58,204	56,251	56,810	-2.45%	0.98%
C30,700,400,V,L	103,932	101,237	101,423	-2.47%	0.18%
C30,700,400,F,T	157,043	n/a	141,037	-11.35%	n/a
C30,700,400,V,T	103,085	n/a	108,481	4.97%	n/a
C30,700,400,F,T	141,917	133,194	134,320	-5.66%	0.84%

Table 4.20: Improvement achieved by the tabu search procedure on C instances

Tables 4.22 and 4.23 give the number of arcs fixed by each type of intensification and CPLEX after 5 hours, as well as comparative ratios. Column 5 compares the number of open arcs by the intensification procedure and CPLEX. The high values in this column indicate that the intensification phase yields good results, the solver having to add only a small number of arcs to satisfy all the constraints. Column 6 displays the ratio of the number of arcs fixed to closed and the number of arcs in the instance. When the number of arcs is large, closing unpromising

Instance	P-TS	CPLEX 1 h	TS	TS/P-TS	TS/CPLEX 1 h
R13,F01,C1	147,837	147,349	148,425	0.40%	0.72%
R13,F05,C1	281,668	277,891	289,535	2.72%	4.02%
R13,F10,C1	404,434	385,396	403,490	-0.23%	4.48%
R13,F01,C2	159,852	155,887	157,743	-1.34%	1.18%
R13,F05,C2	311,209	295,180	306,783	-1.44%	3.78%
R13,F10,C2	470,034	444,545	456,269	-3.02%	2.57%
R13,F01,C8	225,339	218,787	219,105	-2.85%	0.15%
R13,F05,C8	512,027	491,603	500,403	-2.32%	1.76%
R13,F10,C8	875,984	789,479	813,589	-7.67%	2.96%
R14,F01,C1	431,562	422,709	437,836	1.43%	3.45%
R14,F05,C1	811,102	807,553	853,765	5.00%	5.41%
R14,F10,C1	1,193,950	1,199,250	1,214,782	1.71%	1.28%
R14,F01,C2	465,762	452,591	455,609	-2.23%	0.66%
R14,F05,C2	942,678	892,534	914,839	-3.04%	2.44%
R14,F10,C2	1,401,880	1,320,570	1,378,765	-1.68%	4.22%
R14,F01,C8	720,882	702,781	714,841	-0.85%	1.69%
R14,F05,C8	1,795,650	1,747,030	1,743,116	-3.01%	-0.22%
R14,F10,C8	2,997,290	2,767,180	2,874,717	-4.26%	3.74%
R15,F01,C1	1,039,440	1,017,740	1,033,662	-0.56%	1.54%
R15,F05,C1	2,170,310	2,011,860	2,156,433	-0.64%	6.70%
R15,F10,C1	3,194,270	3,011,740	3,362,675	5.01%	10.44%
R15,F01,C2	1,205,790	1,176,050	1,177,198	-2.43%	0.10%
R15,F05,C2	2,698,680	2,607,500	2,785,075	3.10%	6.38%
R15,F10,C2	4,447,950	4,422,350	4,430,712	-0.39%	0.19%
R15,F01,C8	2,472,860	2,401,180	2,408,210	-2.68%	0.29%
R15,F05,C8	6,067,350	5,795,320	5,874,704	-3.28%	1.35%
R15,F10,C8	10,263,600	9,105,010	9,205,777	-11.49%	1.09%
R16,F01,C1	142,692	140,082	141,352	-0.95%	0.90%
R16,F05,C1	261,755	248,703	260,685	-0.41%	4.60%
R16,F10,C1	374,819	344,446	371,854	-0.80%	7.37%
R16,F01,C2	145,266	142,381	143,345	-1.34%	0.67%
R16,F05,C2	277,307	260,993	264,345	-4.90%	1.27%
R16,F10,C2	391,386	361,626	375,258	-4.30%	2.73%
R16,F01,C8	187,176	179,639	181,350	-3.21%	0.94%
R16,F05,C8	423,320	391,101	398,082	-6.34%	1.75%
R16,F10,C8	649,121	599,456	629,107	-3.18%	4.71%
R17,F01,C1	374,016	364,784	372,914	-0.30%	2.18%
R17,F05,C1	718,135	686,722	695,550	-3.25%	1.27%
R17,F10,C1	1,041,450	989,809	1,037,684	-0.36%	4.61%
R17,F01,C2	393,608	382,593	385,225	-2.18%	0.68%
R17,F05,C2	786,198	755,416	782,334	-0.49%	3.44%
R17,F10,C2	1,162,290	1,113,030	1,142,773	-1.71%	2.60%
R17,F01,C8	539,817	530,435	535,475	-0.81%	0.94%
R17,F05,C8	1,348,750	1,232,750	1,299,937	-3.76%	5.17%
R17,F10,C8	2,227,780	2,043,920	2,223,244	-0.20%	8.07%
R18,F01,C1	864,425	845,718	866,802	0.27%	2.43%
R18,F05,C1	1,640,200	1,606,880	1,609,622	-1.90%	0.17%
R18,F10,C1	2,399,230	2,359,170	2,380,708	-0.78%	0.90%
R18,F01,C2	962,402	942,884	966,838	0.46%	2.48%
R18,F05,C2	1,958,150	1,908,250	1,955,396	-0.14%	2.41%
R18,F10,C2	2,986,000	2,813,750	3,157,951	5.45%	10.90%
R18,F01,C8	1,617,320	1,563,820	1,572,109	-2.88%	0.53%
R18,F05,C8	4,268,580	4,039,340	4,312,752	1.02%	6.34%
R18,F10,C8	7,440,780	6,639,848	6,899,618	-7.84%	3.76%

Table 4.21: Improvement achieved by the tabu search procedure on R instances

arcs helps reduce the search space and the running time. We note that there are some instances for which the entries in this column column are relatively low (about 10% to 40%). These are instances for which the number of arcs in the best solutions found by CPLEX is generally quite large (e.g., instances "R14,F01,C8" and "R15,F01,C8") compared to the number of arcs in these instances, which indicates that most arcs are needed in the design. The values in the last two columns are the ratios of the open-intensification and closed-intensification schemes compared to the tabu search methods indicating the good behaviour of the procedures.

Instance	O.Int.	C.Int.	O.CPLEX	O.Int./ O.CPLEX	C.Int./ Arcs	Int. Open/ TS	Int. Close/ TS
C20,230,200,V,L	43	129	57	75%	56%	-2.57%	-3.53%
C20,230,200,F,L	38	153	46	83%	67%	-1.61%	-4.71%
C20,230,200,V,T	48	139	58	83%	60%	-1.20%	-2.17%
C20,230,200,F,T	53	142	62	85%	62%	-1.13%	-2.21%
C20,300,200,V,L	60	193	64	94%	64%	-1.15%	-3.33%
C20,300,200,F,L	53	197	62	85%	66%	-3.10%	-4.05%
C20,300,200,V,T	62	192	71	87%	64%	-0.45%	-1.62%
C20,300,200,F,T	57	183	65	88%	61%	-4.40%	-6.81%
C30,520,100,V,L	69	387	91	76%	74%	-0.26%	-0.40%
C30,520,100,F,L	61	395	75	81%	76%	-1.37%	-2.63%
C30,520,100,V,T	103	365	111	93%	70%	-0.08%	-0.52%
C30,520,100,F,T	79	390	91	87%	75%	0.43%	-1.06%
C30,520,400,V,L	124	339	127	98%	65%	-1.33%	-2.68%
C30,520,400,F,L	171	309	113	104%	59%	-0.67%	-1.21%
C30,520,400,V,T	152	324	n/a	n/a	62%	-0.07%	-2.19%
C30,520,400,F,T	227	254	n/a	n/a	49%	-0.35%	-0.51%
C30,700,100,V,L	78	575	84	93%	82%	0.00%	-0.15%
C30,700,100,F,L	62	577	68	91%	82%	-0.18%	-0.55%
C30,700,100,V,T	94	544	104	90%	78%	-0.37%	-0.70%
C30,700,100,F,T	76	540	98	78%	77%	-1.02%	-1.02%
C30,700,400,V,L	105	554	115	91%	79%	-0.73%	-0.83%
C30,700,400,F,T	104	502	113	92%	72%	0.00%	0.00%
C30,700,400,V,T	131	511	138	95%	73%	-1.83%	-10.84%
C30,700,400,F,T	132	536	128	103%	77%	-0.98%	-2.26%

Table 4.22: Open and closed arc statistics for C instances

Tables 4.24 and 4.25 display information relating to the execution of the proposed procedures. The second column gives the system imbalance for the solution obtained by the initialization step, and the third column reports the total running time. The last two columns give the number of solutions found by tabu search and path relinking phases.

Instance	O.Int.	C.Int.	O.CPLEX	O.Int./ O.CPLEX	C.Int./ Arcs	Int. Open/ TS	Int. Close/ TS
R13,F01,C1	39	162	45	87%	74%	-0.37%	-0.73%
R13,F05,C1	20	160	34	59%	73%	-1.05%	-4.19%
R13,F10,C1	18	151	31	58%	69%	-3.46%	-4.69%
R13,F01,C2	34	149	44	77%	68%	-0.31%	-1.19%
R13,F05,C2	21	158	37	57%	72%	-2.08%	-3.76%
R13,F10,C2	23	162	36	64%	74%	-4.13%	-4.46%
R13,F01,C8	71	101	81	88%	46%	-0.10%	-0.15%
R13,F05,C8	48	126	68	71%	57%	-0.91%	-1.51%
R13,F10,C8	49	124	66	74%	56%	-2.55%	-3.03%
R14,F01,C1	47	125	59	80%	57%	-0.86%	-3.25%
R14,F05,C1	29	141	42	69%	64%	-2.32%	-8.81%
R14,F10,C1	26	161	35	74%	73%	-2.95%	-7.32%
R14,F01,C2	54	115	63	86%	52%	-0.35%	-0.35%
R14,F05,C2	36	146	50	72%	66%	-1.14%	-3.60%
R14,F10,C2	31	155	49	63%	70%	-1.57%	-5.41%
R14,F01,C8	114	22	126	90%	10%	-1.01%	-1.65%
R14,F05,C8	81	93	97	84%	42%	0.90%	-2.83%
R14,F10,C8	65	105	90	72%	48%	-1.70%	-4.24%
R15,F01,C1	63	111	67	94%	50%	-0.42%	-1.40%
R15,F05,C1	48	75	53	91%	34%	-2.84%	-6.90%
R15,F10,C1	37	144	49	76%	65%	-10.27%	-12.63%
R15,F01,C2	72	118	88	82%	54%	-0.15%	-0.23%
R15,F05,C2	65	65	75	87%	30%	-4.81%	-8.29%
R15,F10,C2	71	63	77	92%	29%	-4.03%	-10.29%
R15,F01,C8	159	35	175	91%	16%	0.00%	0.00%
R15,F05,C8	123	39	125	98%	18%	-0.17%	-1.35%
R15,F10,C8	106	81	110	96%	37%	-0.34%	-0.84%
R16,F01,C1	32	245	41	78%	78%	-0.46%	-0.91%
R16,F05,C1	19	219	31	61%	70%	-3.45%	-4.82%
R16,F10,C1	18	250	29	62%	80%	0.00%	-7.71%
R16,F01,C2	36	258	41	88%	82%	-0.16%	-0.52%
R16,F05,C2	22	254	30	73%	81%	-0.66%	-0.92%
R16,F10,C2	24	227	34	71%	72%	-1.81%	-3.09%
R16,F01,C8	50	226	69	72%	72%	-0.09%	-0.68%
R16,F05,C8	37	209	60	62%	67%	-1.36%	-1.60%
R16,F10,C8	36	231	58	62%	74%	-3.73%	-4.08%
R17,F01,C1	40	235	47	85%	74%	-0.29%	-1.75%
R17,F05,C1	28	256	39	72%	81%	-2.81%	-2.81%
R17,F10,C1	18	253	36	50%	80%	-5.24%	-8.89%
R17,F01,C2	50	231	57	88%	73%	0.00%	-0.35%
R17,F05,C2	32	251	43	74%	79%	-3.75%	-5.56%
R17,F10,C2	35	244	42	83%	77%	-3.77%	-4.60%
R17,F01,C8	84	181	99	85%	57%	-0.55%	-0.96%
R17,F05,C8	67	203	90	74%	64%	-3.06%	-5.54%
R17,F10,C8	57	208	88	65%	65%	-3.53%	-11.16%
R18,F01,C1	54	183	65	83%	58%	-0.97%	-2.67%
R18,F05,C1	47	226	45	104%	72%	-2.15%	-2.15%
R18,F10,C1	44	211	40	110%	67%	-1.37%	-6.25%
R18,F01,C2	57	200	75	76%	63%	-1.91%	-2.66%
R18,F05,C2	62	238	60	103%	76%	-1.34%	-3.80%
R18,F10,C2	51	246	58	88%	78%	-8.36%	-13.07%
R18,F01,C8	158	124	160	99%	39%	-0.32%	-2.04%
R18,F05,C8	126	157	116	109%	50%	0.00%	-6.77%
R18,F10,C8	108	170	99	109%	54%	-1.86%	-3.94%

Table 4.23: Open and closed arc statistics for R instances

Instance	System Imbalance	Running Time	Time CPLEX 5 hours	No. TS	No. PR
C20,230,200,V,L	18	91	300	10	85
C20,230,200,F,L	8	92	300	6	40
C20,230,200,V,T	14	78	300	10	43
C20,230,200,F,T	12	84	300	7	33
C20,300,200,V,L	22	162	300	15	95
C20,300,200,F,L	20	110	300	9	47
C20,300,200,V,T	10	90	300	13	68
C20,300,200,F,T	16	141	300	11	88
C30,520,100,V,L	26	85	300	29	185
C30,520,100,F,L	22	118	300	21	187
C30,520,100,V,T	30	103	300	16	166
C30,520,100,F,T	22	165	300	13	109
C30,520,400,V,L	16	221	300	17	157
C30,520,400,F,L	28	146	300	22	110
C30,520,400,V,T	24	210	n/a	19	127
C30,520,400,F,T	38	136	n/a	26	123
C30,700,100,V,L	22	74	67	25	91
C30,700,100,F,L	16	153	300	23	146
C30,700,100,V,T	36	135	300	24	155
C30,700,100,F,T	22	152	300	29	225
C30,700,400,V,L	24	136	300	35	125
C30,700,400,F,T	26	80	300	21	185
C30,700,400,V,T	32	222	300	14	116
C30,700,400,F,T	28	130	300	13	60

Table 4.24: Statistical information execution C instances

Instance	System Imbalance	Running Time	Time CPLEX 5 hours	No. TS	No. PR
R13,F01,C1	18	62	1	16	78
R13,F05,C1	12	63	54	19	113
R13,F10,C1	8	63	49	15	115
R13,F01,C2	18	62	1	18	80
R13,F05,C2	12	63	105	18	128
R13,F10,C2	16	63	300	13	93
R13,F01,C8	28	62	20	18	82
R13,F05,C8	28	71	300	11	98
R13,F10,C8	22	63	300	18	136
R14,F01,C1	24	69	15	23	140
R14,F05,C1	18	65	300	16	121
R14,F10,C1	14	73	300	6	28
R14,F01,C2	16	65	7	14	81
R14,F05,C2	16	95	300	13	109
R14,F10,C2	16	85	300	9	57
R14,F01,C8	34	107	290	16	71
R14,F05,C8	18	108	300	9	39
R14,F10,C8	28	169	300	10	126
R15,F01,C1	20	67	45	6	31
R15,F05,C1	14	114	300	9	60
R15,F10,C1	10	125	300	4	18
R15,F01,C2	20	87	300	12	66
R15,F05,C2	20	146	300	3	26
R15,F10,C2	18	170	300	5	44
R15,F01,C8	34	117	300	4	10
R15,F05,C8	18	149	17	7	29
R15,F10,C8	34	85	5	7	28
R16,F01,C1	6	62	1	14	54
R16,F05,C1	10	63	126	23	96
R16,F10,C1	12	63	300	11	81
R16,F01,C2	14	64	1	22	103
R16,F05,C2	18	63	100	14	84
R16,F10,C2	18	67	300	12	105
R16,F01,C8	32	66	73	31	146
R16,F05,C8	32	64	300	15	119
R16,F10,C8	24	73	300	18	142
R17,F01,C1	14	71	26	15	83
R17,F05,C1	8	133	300	13	121
R17,F10,C1	18	131	300	13	123
R17,F01,C2	20	80	23	26	129
R17,F05,C2	18	66	300	11	68
R17,F10,C2	20	90	300	15	118
R17,F01,C8	34	115	300	25	178
R17,F05,C8	32	187	300	9	70
R17,F10,C8	32	137	300	7	61
R18,F01,C1	14	136	300	20	148
R18,F05,C1	10	96	300	7	60
R18,F10,C1	10	72	300	9	38
R18,F01,C2	24	219	300	11	129
R18,F05,C2	12	91	300	3	18
R18,F10,C2	12	140	300	4	26
R18,F01,C8	38	127	300	5	26
R18,F05,C8	40	128	300	5	21
R18,F10,C8	20	68	300	3	7

Table 4.25: Statistical information execution R instances

5. SERVICE NETWORK DESIGN WITH RESOURCES CONSTRAINTS

Summary of the Chapter 5

This chapter of the thesis presents our second work which was accepted to publish in *Transportation Science* with the following reference information:
T. G. Crainic, M. Hewitt, M. Toulouse, D. M. Vu: Service Network Design with Resource Constraints, *Transportation Science* (2013)

Abstract

We first present a new service network design model for freight consolidation carriers, one that selects services and routes both commodities and the resources needed to support the services that transport them, while explicitly recognizing that there are limits on how many resources are available at each terminal. We next present a solution approach that combines column generation, meta-heuristic, and exact optimization techniques to produce high-quality solutions. We demonstrate the efficacy of the approach with an extensive computational study and benchmark its performance against both a leading commercial solver and a column generation-based heuristic.

Keywords: *Service network design, resource constraints, matheuristics, slope scaling, column generation.*

5.1 Introduction

Service network design formulations are extensively used to address planning issues within many application fields, in particular for the tactical planning of operations of consolidation-based modal and multimodal carriers and organizations (e.g., Christiansen *et al.*, 2007; Cordeau *et al.*, 1998; Crainic, 2000, 2003; Crainic & Kim, 2007; Crainic & Laporte, 1997). The main goal of such formulations is to produce an operations (or load) plan that services the estimated demand while achieving the economic and service-quality targets of the carrier. Building such a plan involves principally selecting the services to operate and their schedules (departure times) and routing the demand through the selected service network. Most service network design models proposed in the literature consider the resources required to perform the services (vehicles, power units, drivers, etc.) only indirectly, however, which is increasingly inadequate to reflect the operation strategies of a broad range of transportation systems (e.g., Bektaş, T. & Crainic, T. G., 2008).

Only recently have researchers proposed models and solution methods that recognize management issues related to the resources needed to implement a service network (e.g., Andersen *et al.*, 2009b,a). The range of resource-management issues considered is still very limited, however. Moreover, introducing resource-management considerations within service network design formulations raises significant methodological challenges far from being satisfactorily addressed.

One goal of this paper is to address these challenges by enlarging the range of resource-management aspects included into tactical planning models. To do so, we consider a setting wherein resources required to perform services are assigned to terminals to which they must ultimately return and there are a finite number of resources assigned to each terminal in the network. Such a restriction may be seen when resources represent crews (e.g., truck drivers and railroad train operators), power units, particular vehicles, etc. While resources must return to a home terminal in many transportation settings, each setting likely imposes a different set of additional rules governing their movements (e.g., drivers have to take periodic rests while vehicles do not). Recognizing this, our second goal for this paper is to introduce an efficient and general purpose solution methodology that can be easily adapted to *service network design with resource constraints* (*SNDRC*) formulations, with different formulations modeling different operational realities. To achieve this latter goal, we turn to a formulation different from what has been generally used in the service network design literature up to now. The formulation

builds upon the structure introduced by the resource constraints and explicitly works with the resulting circular routes performed by resources supporting the selected services.

More precisely, our contributions are to 1) propose a new mathematical formulation for the scheduled SNDRC problem explicitly accounting for the limited number of resources available at each terminal and taking advantage of the structure introduced into the model by the resource-management constraints; 2) introduce an advanced matheuristic solution methodology, combining long-term memory-enhanced slope scaling, column generation, and mathematical programming techniques, which is both general for the SNDRC problem class and very efficient for the particular problem at hand; 3) demonstrate this efficiency through a comprehensive experimental study and benchmarking against a leading commercial software.

The paper is organized as follows. Following a brief literature review in Section 5.2, we state the problem and detail the network model and mathematical formulation in Section 5.3. Section 5.4 is dedicated to the proposed matheuristic solution approach, while Section 5.5 presents the experimental study. We conclude in Section 5.6.

5.2 Literature Review

One resource (or asset) management requirement modeled by researchers is that a resource must be available at the origin of a service for the service to be performed. Among other things, this enables a model to capture the (sometime) necessity for a resource to first move empty before it can carry a load. Mathematically, it is modeled by requiring that the number of resources entering and leaving a terminal must be the same. Pedersen *et al.* (2009) refer to these as “design-balance constraints" and present service network design models based on arc and cycle network formulations. Indeed, the design-balance constraints induce a cyclic structure for the resource movements supporting the service operations and the routing of flow. Pedersen *et al.* (2009) also proposed a two-phase tabu search method for the arc-based formulation. Meta-heuristic algorithms improving over the results of Pedersen *et al.* (2009) were proposed by Chouman & Crainic (2013) and Vu *et al.* (2013), the latter being used as a subroutine in the approach presented in this paper. This requirement has been modeled for various modes of transportation, e.g., Barnhart & Schneur (1996); Kim *et al.* (1999) for air-based express package delivery, Lai & Lo (2004) for ferries, Andersen & Christiansen (2009); Andersen *et al.* (2009b) for intermodal rail, and Erera *et al.* (2012); Smilowitz *et al.* (2003) for trucking.

The cycle structure induced by the design-balance constraints is actually modeling the fact that, in most settings, a resource is associated with a specific terminal in a network and must return there before the end of the schedule. An example in trucking is drivers, who, by federal (and sometimes labor union) regulations, must periodically return to the terminal closest to their home. Andersen *et al.* (2009a) thus study the computational performance of arc and cycle-based formulations, by solving limited-size instances with a commercial mixed integer programming solver, and concluded that the latter offers the most promising approach. Recognizing that a priori complete cycle generation will not scale to large instances, Andersen *et al.* (2011) then presented an effective branch-and-price scheme for the cycle-based formulation. The problem is NP-hard, however, and even more efficient methods are required. This is the goal of this paper for the problem setting that also extends the range of resource limitations considered.

5.3 Problem Statement and Model

As in many tactical planning settings for consolidation-based carriers, we assume the demand and, thus, the carrier services display a somewhat regular, repetitive pattern. The *planning horizon* is then the length of time, e.g., a “season", for which this pattern is assumed. Services are then selected and scheduled over a number of periods making up the *schedule length* (e.g., a week), the resulting schedule being repeated a certain number of times during the planning horizon. Let’s assume the schedule length is divided into $\mathcal{T} = \{1, 2, \ldots, \text{TMAX}\}$ time periods (likely days).

We model the operations of a carrier with a time-space network, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where terminal activities in different periods are modeled with different nodes. Specifically, we assume a set $\mathcal{L}$ of terminals in the carrier’s network and define the set of nodes, $\mathcal{N}$, to model the operations of terminals in different periods, i.e., $\mathcal{N} = \mathcal{L} \times \mathcal{T} = \{l_t | l \in \mathcal{L}, t \in \mathcal{T}\}$, where l_t represents terminal l at period t.

There are two types of arcs in the set $\mathcal{A}$. The first is a *service arc* and models the operation of a service between two terminals at a particular point in time. The second is a *holding arc* and models the opportunity for a resource or shipment to idle at a terminal from one period to the next. We denote the set of service arcs by $\mathcal{S}$ and the set of holding arcs by $\mathcal{H}$ and thus $\mathcal{A} = \mathcal{S} \cup \mathcal{H}$. For each possible service $s = (l, m)$ between terminals $l, m \in \mathcal{L}$ and

$t \in \{1, \ldots, \text{TMAX}\}$, we add the arc $(l_t, m_{(t+\pi) \bmod \text{TMAX}})$ to $\mathcal{S}$ (assuming the arc length is π periods). Due to the cyclic nature of the problem and model, our time-space network "wraps around." Specifically, we model a service of length π that departs from a terminal in period t such that $t + \pi >$ TMAX as arriving at the destination in period $(t + \pi)$ mod TMAX. As an example, see the arc originating from T1 in period 7 and terminating at T2 in period 1 in Figure 5.1.

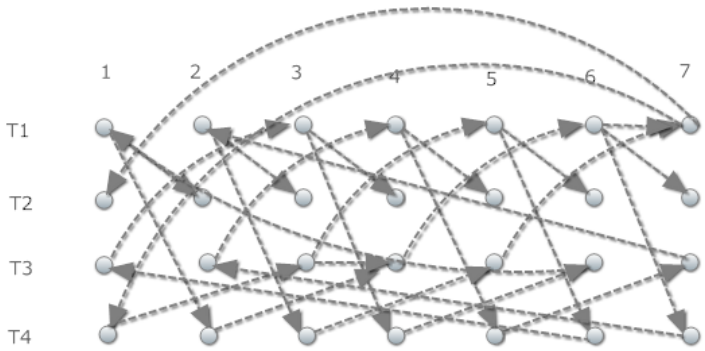


Figure 5.1: Time-Space Network for Cyclic Service Schedules

We also assume a limit, u_s, on how much shipment demand can be carried by service s and set the capacity, u_{ij}, of executions of that service at different times to u_s. Regarding the *holding arcs*, we add to $\mathcal{H}$ an arc of the form $(l_t, l_{(t+1) \bmod \text{TMAX}})$ for each terminal l and period t. While we assume these arcs are uncapacitated (both with respect to shipment demands and resources) in our experiments, terminal capacities (on shipments or resources) could be modeled by placing capacities on these arcs.

We model a shipment that needs to be delivered from terminal l and available in period t to terminal, m, by period $t^{'}$ as a commodity with index k, origin node $o(k) = l_t$, and destination node $d(k) = m_{t'}$. We denote the size of this shipment as w_k. The set of all shipments is represented by $\mathcal{K}$.

We model three types of costs. The first is a variable cost associated with commodity k traveling on arc $(i, j) \in \mathcal{A}$ and is denoted c_{ij}^k. These costs can model the impact the weight of a shipment can have on the cost of executing a service or the labor costs associated with handling a commodity at a terminal. The second is a fixed cost associated with the use of a resource, and is denoted by F. These costs can model the cost of paying drivers or the depreciation of a capital asset. The third is a fixed cost associated with executing a service departing from

terminal l and arriving at terminal m and is denoted by f_{lm}. These costs can model the actual transportation cost of a resource traveling from terminal l to terminal m.

Like many network design models, for a commodity to travel on arc $(l_t, m_{t'}) \in A$, that service must be "executed", or, the arc must be "installed" in the network. However, we also model the situation where a resource is needed to execute that service, resources are assigned to terminals to which they must ultimately return, and there are a limited number of resources assigned to each terminal. To do so, we model a sequence of possible movements during the schedule length, called *itinerary* hereafter, for a resource assigned to terminal l with a cycle in the graph $\mathcal{G}$ that begins and ends at node $l_t \in \mathcal{N}$ for some $t \in \mathcal{T}$. We denote the set of such cycles by θ_{lt} and set $\theta_l = \cup_{t=1}^{TMAX}\theta_{lt}$, as the set of all cycles that depart from terminal l during the schedule length. Thus, choosing a cycle from θ_l implies specifying the itinerary for a resource assigned to terminal l. With the set θ_l we can model that terminal l has a fixed number of resources, ub_l, that are available during the schedule length by ensuring that at most ub_l cycles are chosen from the set θ_l. We let r_{ij}^{τ} (binary) denote whether arc (i, j) is contained in cycle τ, and $\theta = \cup_{l\in\mathcal{L}}\theta_l$. Finally, we note that in this paper we are focusing on a setting wherein a service can be executed by at most one resource, as is often seen in rail transportation (Andersen *et al.*, 2011). However, the model and solution approach can be adapted to other settings.

Thus, the rules governing the movements a resource may make during the schedule length are encoded in the definition of the sets θ_{lt}. For our experiments, we do not impose rules other than that the itinerary for a resource must begin and end at the resource's assigned terminal. Thus a valid cycle is one that begins by departing from l in period t and ends by arriving at l in period t, albeit TMAX periods later. We note that we allow a cycle beginning at l_t to return to l multiple times (see the path beginning at terminal 1 in Figure 5.2), and if it last returns to l in period $t^{'} < t+$TMAX we append holding arcs so that it reaches $l_{t+\text{TMAX}}$ (see the path beginning at terminal 3 in Figure 5.2). While we model a simple set of rules governing what movements a resource may make, much of the methodology we propose can be applied to other cases by changing the definition of the sets θ_l. Some examples are operating environments wherein union or federal regulations governing what a driver may do must be modeled or the length of time a resource may be away from its assigned terminal can exceed the length of the schedule.

Because we only model that the execution of a service requires a resource we do not explicitly assign shipments to resources and instead allow the transfer of a shipment from one resource to another. Specifically, while a shipment may travel on a sequence of services, each

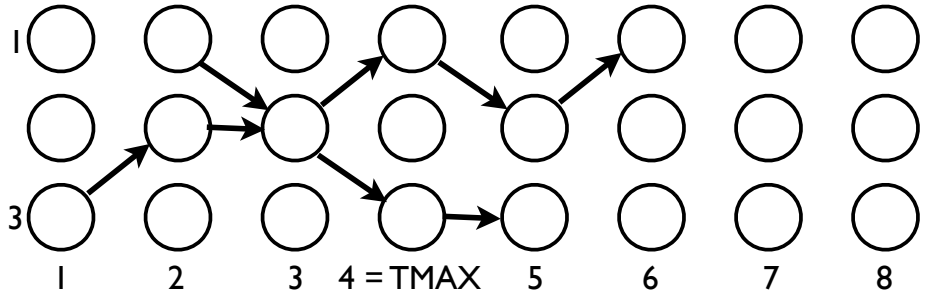


Figure 5.2: Valid Cycles (shown as paths in $\mathcal{G}'$)

of those services can be contained in a different cycle, that is, may be supported by a different resource. Similarly, we do not model that a resource is required for a shipment to travel on a *holding arc*, although the model and algorithm we propose could be adapted to such a scenario with little change.

We have two types of variables in our model. The first, x_{ij}^k, is a continuous variable that indicates the amount of commodity k that flows on arc $(i,j) \in \mathcal{A}$. The second, z_τ, is a binary variable that indicates whether cycle $\tau \in \theta$ is selected. Thus, the cycle-based formulation of scheduled service network design with resources constraints (SNDRC), which seeks to select cycles that can carry all commodities from their origins to their destinations and route those commodities, seeks to

$$\text{minimize} \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \sum_{\tau\in\theta} F z_\tau + \sum_{(i,j)\in\mathcal{A}} \sum_{\tau\in\theta} r_{ij}^\tau z_\tau f_{ij}$$

subject to

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^k = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k)\ \ \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \\ -w^k & \text{if } i = d(k) \end{cases} \tag{5.1}$$

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij} \sum_{\tau\in\theta} r_{ij}^\tau z_\tau, \quad \forall (i,j) \in \mathcal{S}, \tag{5.2}$$

$$\sum_{\tau\in\theta} r_{ij}^\tau z_\tau \leq 1, \quad \forall (i,j) \in \mathcal{S}, \tag{5.3}$$

$$\sum_{\tau\in\theta_l} z_\tau \leq ub_l, \quad \forall l \in \mathcal{L}, \tag{5.4}$$

$$x_{ij}^k \geq 0, \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{K}, \tag{5.5}$$

$$z_\tau \in \{0,1\}, \quad \forall \tau \in \theta. \tag{5.6}$$

The objective is to minimize the total cost of using resources ($\sum_{\tau\in\theta} F z_\tau$), operating services ($\sum_{(i,j)\in\mathcal{A}} \sum_{\tau\in\theta} r_{ij}^\tau z_\tau f_{ij}$), and routing commodities ($\sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k$). Constraint set (5.1), named the ***flow balance constraints***, ensures that each commodity is routed from its origin node to its destination node. Constraint set (5.2), named the ***weak forcing constraints***, ensures that a service is executed when it is used by a commodity and that its capacity is not exceeded. Constraint set (5.3), named the ***0-1 service constraints***, ensures that each service belongs to at most one executed cycle. Constraint set (5.4), named the ***resource bound constraints***, ensures that the total number of resources originating from each terminal does not exceed the number available. Lastly, constraint sets (5.5) and (5.6) define the domains of the variables. While constraints of the form $x_{ij}^k \leq \min(w^k, u_{ij}) \sum_{\tau\in\theta} r_{ij}^\tau z_\tau$ can significantly strengthen the linear relaxation, there are often too many of them to consider them all explicitly.

5.4 Solution Approach

One of the challenges in producing a high quality solution to the SNDRC is that for even reasonably-sized instances, the set θ is too large to be enumerated. Thus, to produce a high-quality solution to the SNDRC, one needs a method for producing the cycles that appear in good solutions and a method that can produce a high quality solution given a fixed set of cycles. We illustrate the steps of the solution approach in Figure 5.3, where the number next to each step corresponds to the section where it is described. As indicated in Figure 5.3, column generation (Barnhart *et al.*, 1998; Desaulniers *et al.*, 2005) is one method used by the solution approach to generate cycles and we next describe how it is done for the SNDRC.

5.4.1 Column Generation

We define SNDRC($\bar{\theta}$) to be the SNDRC restricted to the cycles in $\bar{\theta}$ and its linear relaxation to be the SNDRC with the constraints $z_\tau \in \{0,1\}, \tau \in \bar{\theta}$ replaced by $0 \leq z_\tau \leq 1, \tau \in \bar{\theta}$. The motivation behind how column generation solves a linear program is that variables need not be explicitly added to the instance until optimality or feasibility conditions dictate that they may

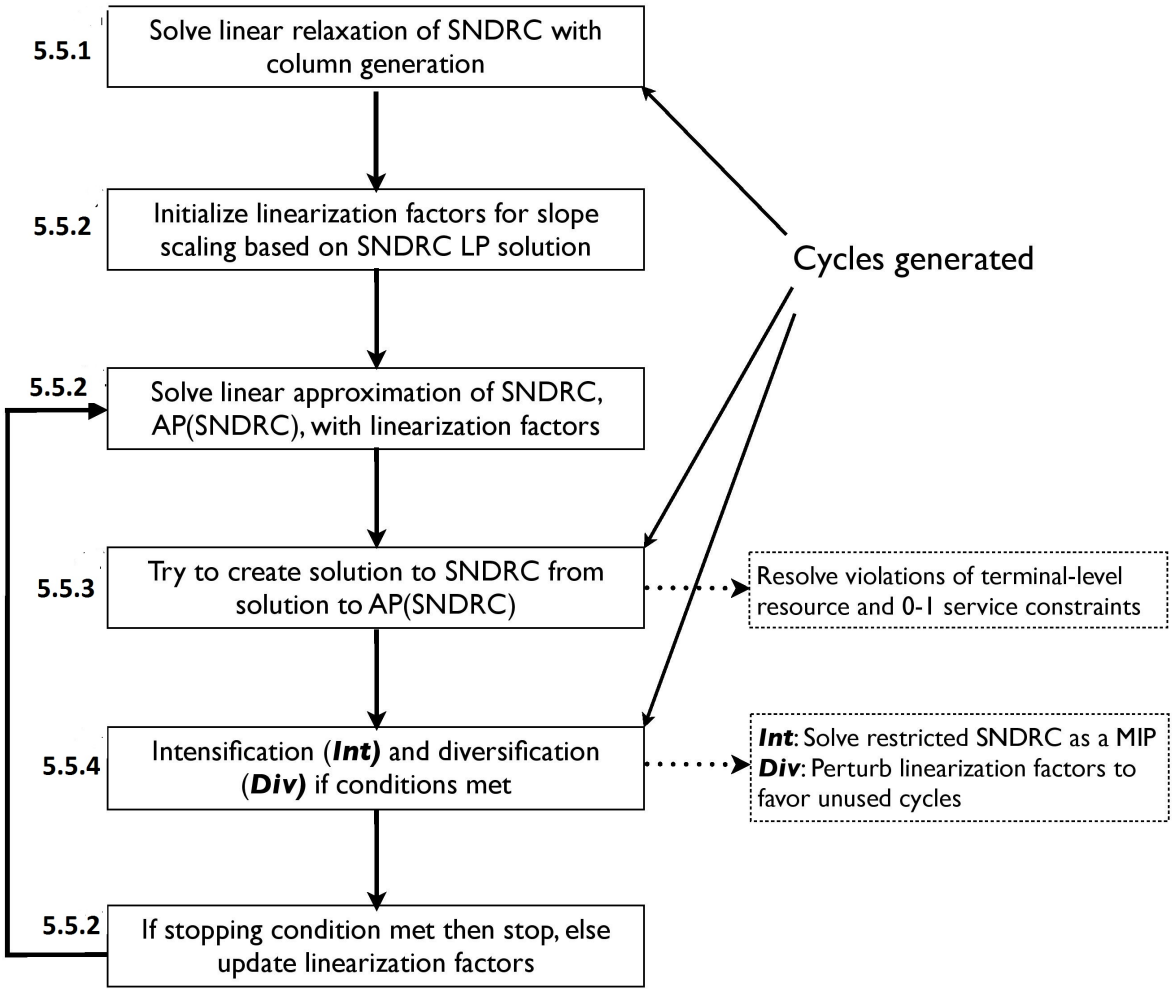
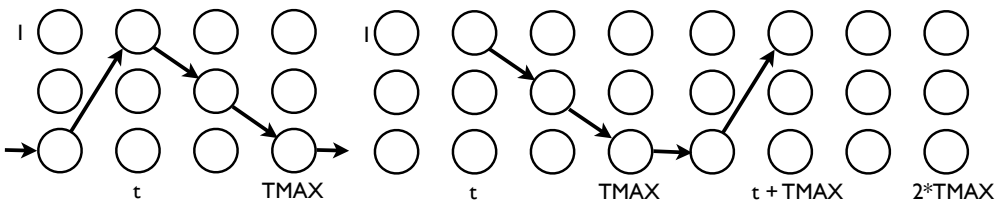


Figure 5.3: Steps of solution approach

be necessary for finding the optimal solution. In particular, assuming $\bar{\theta}$ contains cycles such that SNDRC($\bar{\theta}$) is feasible, SNDRC($\bar{\theta}$) will be repeatedly solved, with new cycles (z_τ variables) added to $\bar{\theta}$ when reduced cost calculations indicate that they may lead to an improved solution.

Thus, we associate the dual variables α_{ij} (≤ 0) with each constraint in set (5.2), β_{ij} (≤ 0) with each constraint in set (5.3), and γ_l (≤ 0) with each constraint in set (5.4). Then, the reduced cost associated with variable z_τ is $F - \gamma_l + \sum_{(i,j)\in\tau}(f_{ij} + \alpha_{ij}u_{ij} - \beta_{ij})$. Given these dual values, we search for cycles with negative reduced costs to add to the set $\bar{\theta}$.

In general, we search for a cycle in $\mathcal{G}$ that begins and ends at node l_t by creating a second network, $\mathcal{G}'$ where we append a copy of $\mathcal{G}$ after $\mathcal{G}$ and search for a path in this network from l_t to $l_{t+\text{TMAX}}$. We illustrate the network $\mathcal{G}$ and a cycle that originates at terminal 1 in period t in Figure 5.4(a) and the corresponding path in $\mathcal{G}'$ in Figure 5.4(b). Thus, we often describe cycles in $\mathcal{G}$ as paths in $\mathcal{G}'$.



(a) Cycle in $\mathcal{G}$ (b) Path in $\mathcal{G}'$

Figure 5.4: Correspondence between cycles and paths.

Thus, the search for a cycle with negative reduced cost can be performed by finding the shortest path in the time-space network $\mathcal{G}'$ with respect to arc costs $f_{ij} + \alpha_{ij}u_{ij} - \beta_{ij}$ from l_t to $l_{t+\text{TMAX}}$ for each $l \in \mathcal{L}, t \in \mathcal{T}$. We note that the search may generate two cycles that contain the same services (in terms of terminal to terminal moves) but begin in different periods (i.e. are associated with nodes $l_t, l_{t'}$). However, because shipments may originate at any period during the schedule, both the services and their departure times determine the usefulness of a cycle. Similarly, the same cycle may be generated when searching for cycles with negative reduced costs for different starting nodes, l_t. However, because a cycle models the use of a resource at its starting node and the number of such resources is limited by terminal, the starting node distinguishes two cycles that contain the same arcs in $\mathcal{A}$. Finally, we note that we do not preclude the procedure from generating a cycle that visits terminal l at times t and t' multiple times.

Algorithm 5.1 Column generation procedure for solving the linear relaxation of SNDRC

Find an initial set of cycles $\bar{\theta}$
while have not solved linear relaxation **do**
  Solve linear relaxation of SNDRC($\bar{\theta}$)
  Determine dual values
  for all $l_t \in \mathcal{N}$ **do**
    Find the shortest path in $\mathcal{G}$ from l_t to l_{t+TMAX} with respect to arc costs $f_{ij} + \alpha_{ij} u_{ij} - \beta_{ij}$.
    Let ω denote the length of this shortest path.
    if $F - \gamma_l + \omega < 0$ **then**
      Add the corresponding cycle to $\bar{\theta}$
    end if
  end for
  Stop if no variables with negative reduced cost found
end while

We present the algorithm for solving the linear relaxation of SNDRC in Algorithm 5.1. At the beginning of Algorithm 5.1 we generate cycles that can deliver flow for at least one commodity from its source to its destination. Specifically, for each commodity $k \in \mathcal{K}$, originating at $o(k) = l_t$, we generate λ cycles that begin at $o(k)$, pass through $d(k) = m_{t'}$ and then return to $o(k)$. We find these cycles by finding λ shortest paths from $o(k) = l_t$ to $l_{t+\text{TMAX}}$ in $\mathcal{G}'$ with respect to the arc costs f_{ij} for service arcs, 0 for holding arcs not associated with terminal m, and a value $< -1 * \sum_{(i,j)\in\mathcal{A}} f_{ij}$ for holding arcs associated with terminal m. A very negative value is assigned to holding arcs associated with terminal m to ensure each shortest path visits the destination terminal for commodity k. To find multiple shortest paths, we use Yen's algorithm (Yen, 1971).

Solving the linear relaxation of SNDRC yields a bound on the optimal value of SNDRC. However, we also use the cycles generated during this solution process and the solution to the linear relaxation itself to inform a slope scaling procedure that produces high-quality primal solutions. We next describe this slope-scaling procedure.

5.4.2 Slope Scaling

When using slope scaling (Kim & Pardalos, 1999; Crainic *et al.*, 2004) to produce solutions to an integer program, a linear relaxation of the problem is repeatedly solved with an objective function that is parameterized by *linearization factors*. Each time the linear relaxation is

solved, a solution to the integer program is constructed (typically through rounding) and the linearization factors are updated so that the cost of the solution to the linear relaxation equals the cost of the solution to the integer program. Our slope scaling approach differs from what is traditionally seen in that instead of solving the linear relaxation of SNDRC, our approach solves an approximation (which is also a linear program) and then executes a separate procedure to construct a feasible solution to SNDRC from a solution to the approximation. We next describe this approximation problem, called AP(SNDRC).

We build an instance of AP(SNDRC) with a fixed set of cycles $\tilde{\theta}$ and solve it to identify a set of cycles enabling each commodity's demand to flow from its source to its sink. As such we introduce additional (continuous) variables $x_{ij}^{k\tau} \geq 0$ which indicates the amount of commodity k's demand routed on arc (i,j) and carried by cycle τ (we set $x_{ij}^{k\tau} = 0$ when $(i,j) \notin \tau$). We relate these new variables to the variables x_{ij}^{k} with the equation $x_{ij}^{k} = \sum_{\tau \in \tilde{\theta}} x_{ij}^{k\tau}$. With linearization factors $\rho_{ij}^{k\tau}(t)$ that are parameterized by an iteration counter, t, AP(SNDRC($\rho(t)$)) seeks to

$$\text{minimize} \sum_{\tau \in \tilde{\theta}} \sum_{ij \in \mathcal{A}} \sum_{k \in K} x_{ij}^{k\tau} \left(c_{ij}^{k} + \rho_{ij}^{k\tau}(t) \right)$$

subject to

$$\sum_{\tau \in \tilde{\theta}} x_{ij}^{k\tau} = x_{ij}^{k}, \forall (i,j) \in \mathcal{A},\ \forall k \in \mathcal{K}, \tag{5.7}$$

$$\sum_{j:(i,j) \in \mathcal{A}} x_{ij}^{k} - \sum_{j:(j,i) \in \mathcal{A}} x_{ji}^{k} = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k)\ \ \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \\ -w^k & \text{if } i = d(k) \end{cases} \tag{5.8}$$

$$\sum_{k \in \mathcal{K}} x_{ij}^{k} \leq u_{ij}, \quad \forall (i,j) \in \mathcal{A}, \tag{5.9}$$

$$x_{ij}^{k\tau} \geq 0, \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{K},\ \tau \in \tilde{\theta}. \tag{5.10}$$

Solving AP(SNDRC(ρ)) in iteration t yields the vector $\tilde{x}_{ij}^{k\tau}(t)$ which can be used to produce a (possibly infeasible) solution to SNDRC by setting $\tilde{z}_{\tau}(t) = 1$ if $\sum_{(i,j) \in \tau} \sum_{k \in \mathcal{K}} \tilde{x}_{ij}^{k\tau}(t) > 0$. This solution $(\tilde{x}(t), \tilde{z}(t))$ has the objective function value

$$\sum_{\tau \in \tilde{\theta}} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^{k} \tilde{x}_{ij}^{k\tau}(t) + \sum_{\tau \in \tilde{\theta}} F \tilde{z}_{\tau}(t) + \sum_{\tau \in \tilde{\theta}} \sum_{(i,j) \in \mathcal{A}} r_{ij}^{\tau} f_{ij} \tilde{z}_{\tau}(t).$$

Thus, for a solution $\tilde{x}(t)$ of AP(SNDRC($\rho(t)$)), the values $\rho_{ij}^{k\tau}(t+1)$ are calculated to satisfy

the relation $\sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}}\tilde{x}_{ij}^{k\tau}(t)(c_{ij}^{k}+\rho_{ij}^{k\tau}(t+1)) = \sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}}\tilde{x}_{ij}^{k\tau}(t)c_{ij}^{k} + F\tilde{z}_{\tau}(t) + \sum_{(i,j)\in\mathcal{A}} r_{ij}^{\tau}f_{ij}\tilde{z}_{\tau}(t)$, which can be done by setting

$$\rho_{uv}^{k\tau}(t+1) = \begin{cases} \frac{F+\sum_{(i,j)\in\tau} r_{ij}^{\tau}f_{ij}}{\sum_{k\in\mathcal{K}}\sum_{ij\in\tau}\tilde{x}_{ij}^{k\tau}(t)}, & \text{if } \sum_{k\in\mathcal{K}}\sum_{ij\in\tau}\tilde{x}_{ij}^{k\tau}(t) > 0 \\ \rho_{uv}^{k\tau}(t) & , \ otherwise \end{cases}, \ \forall\tau\in\tilde{\theta},\ k\in\mathcal{K},\ (u,v)\in\tau. \tag{5.11}$$

If the cycle τ appears in the solution to the linear relaxation of the SNDRC, we set $\rho_{ij}^{k\tau}(0) = 1$. Otherwise, we set $\rho_{ij}^{k\tau}(0) = \left(F + \sum_{(i,j)\in S} r_{ij}^{\tau}f_{ij}\right) \diagup \sum_{(i,j)\in S} r_{ij}^{\tau}u_{ij}$.

When the solution $(\tilde{x}(t), \tilde{z}(t))$ violates the ***0-1 service constraints*** (constraint set (5.3)), or the ***resource bound constraints*** (constraint set (5.4)), our approach next executes the following procedures to try and construct a feasible solution to SNDRC from $(\tilde{x}(t), \tilde{z}(t))$.

5.4.3 Creating a Feasible Solution to SNDRC from a Solution to AP(SNDRC)

Because the AP(SNDRC) does not include all the constraints that are present in the SNDRC (constraint sets (5.3) and (5.4)), and unlike most slope scaling approaches, rounding procedures are not sufficient to construct a feasible solution to the SNDRC from a solution, $(\tilde{x}(t), \tilde{z}(t))$, to the AP(SNDRC). Instead we must execute another procedure to construct a feasible solution to the SNDRC. Our procedure, instead of modifying the solution to the AP(SNDRC) directly, creates a subgraph, $\bar{\mathcal{G}}$ of $\mathcal{G}$ that contains the nodes $\mathcal{N}$ and a subset, $\bar{\mathcal{A}}$, of the arcs, $\mathcal{A}$, that can be decomposed into cycles. Then, the procedure attempts to extract cycles from this subgraph that can be used to construct a feasible solution to the SNDRC.

We create $\bar{\mathcal{G}}$ by first adding to $\bar{\mathcal{A}}$ all service arcs (i,j) in $\mathcal{A}$ such that either $\sum_{\tau\in\tilde{\theta}}\tilde{x}_{ij}^{k\tau}(t) > 0$ for some $k\in\mathcal{K}$, or they belong to a cycle used in the AP(SNDRC) solution. We also add all holding arcs to $\bar{\mathcal{A}}$. Next, if in the solution $(\tilde{x}(t), \tilde{z}(t))$, there are terminals where the resource bound constraint is violated (i.e. $\sum_{\tau\in\theta_l}\tilde{z}_l(t) > ub_l$), we solve an optimization problem to add service arcs that are in $\mathcal{A}\backslash\bar{\mathcal{A}}$ to $\bar{\mathcal{A}}$ that will enable an unused resource at one terminal to move to a terminal where an excess number of resources is used. Finally, we solve another optimization problem to add service arcs in $\mathcal{A}\setminus\bar{\mathcal{A}}$ to $\bar{\mathcal{A}}$ to ensure that $\bar{\mathcal{G}}$ can be decomposed into cycles in such a way that each service arc appears in at most one cycle but holding arcs may appear in multiple cycles, as doing so maximizes the number of cycles that can be extracted from $\bar{\mathcal{G}}$. The

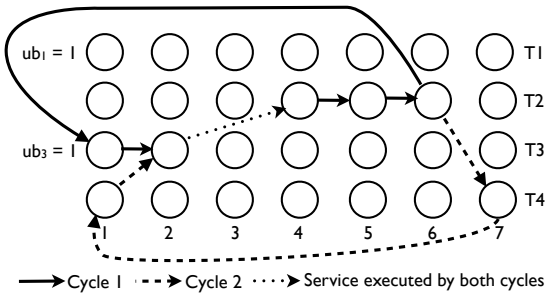
objective of both of these optimization problems is to minimize the total cost of the arcs added with respect to the cost coefficient f_{ij}. Lastly, we note that we formulate and solve both of these optimization problems as a minimum cost, maximum flow problem (Ahuja *et al.*, 1994). See Vu *et al.* (2013) for details of how a similar procedure was done for a network design problem with Eulerian-type constraints.

After creating $\bar{\mathcal{G}}$, we next extract a set of cycles from this network that are guaranteed to satisfy all the constraints of the SNDRC other than the flow conservation constraints. Specifically, to extract cycles from $\bar{\mathcal{G}}$ into the set $\bar{\theta}$ we execute Algorithm 5.2.

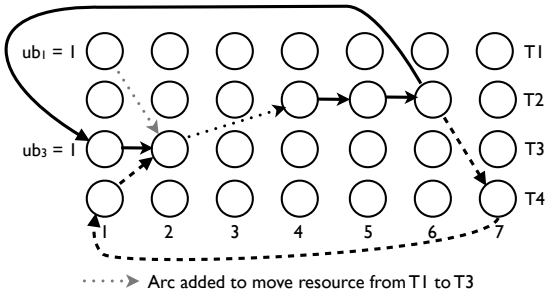
Algorithm 5.2 Cycle extraction procedure

Require: Graph $\bar{\mathcal{G}}$
Require: An ordering $O_{\mathcal{N}}$ of the nodes $\mathcal{N}$
1: Set $\bar{\theta} = \emptyset$
2: **for all** $l \in \mathcal{L}$ **do**
3: Set $\kappa_l = ub_l$
4: **end for**
5: **for all** $l_t \in O_{\mathcal{N}}$ such that $\kappa_l > 0$ **do**
6: Perform depth-first search from l_t to identify a feasible cycle τ in $\bar{\mathcal{G}}$.
7: Add τ to $\bar{\theta}$
8: Set $\kappa_l = \kappa_l - 1$
9: **for all** service arcs $(i, j) \in \tau$ **do**
10: Remove (i, j) from $\bar{\mathcal{A}}$.
11: **end for**
12: **end for**
13: Repeat steps 5 to 12 until no cycles found

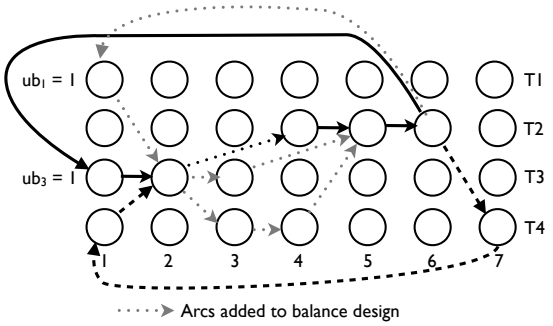
With steps 3 and 8 of Algorithm 5.2 and the condition of the for loop in step 5, we ensure that at most ub_l resources from each terminal l are used. Similarly, step 10 ensures that each service arc in $\bar{\mathcal{G}}$ will appear in at most one cycle in $\bar{\theta}$. As a result, setting $z_\tau = 1 \quad \forall \tau \in \bar{\theta}$ will not violate either the resource bound (5.4) or 0-1 service (5.3) constraints of the SNDRC. To see if the flow conversation constraints can be satisfied using the cycles in $\bar{\theta}$, we solve a minimum-cost multicommodity network flow problem (Ahuja *et al.*, 1994) with respect to the arc costs c_{ij}^k on the network induced by those cycles. If the flow conservation constraints can be satisfied, we have a new feasible solution. Finally, to generate a diverse set of solutions, we call Algorithm 5.2 multiple times, each time perturbing the ordering $O_{\mathcal{N}}$ of how nodes are processed and the ordering in which nodes are considered during the execution of the depth-first search.



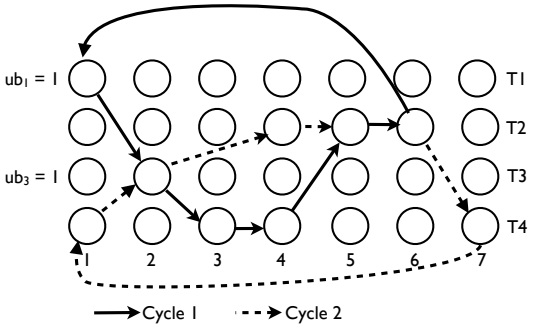
(a) A solution to AP(SNDRC)



(b)Resolve resource bound constraints



(c) Ensure subgraph can decompose into cycles



(d)Extract feasible cycles

	T1	T2	T3	T4
T1	-	2	1	1
T2		-	2	1
T3			-	1
T4				-

(e) Travel times (in periods)

Figure 5.5: Converting a solution to AP(SNDRC) to a solution to SNDRC.

We illustrate this procedure in Figures 5.5(a) to 5.5(d), with the travel times for the underlying network depicted in Figure 5.5(e). While our approach does not assume that travel times are symmetric, they are in this example, and thus the matrix depicting travel times is not full. Similarly, while the travel times in this example satisfy the triangle inequality, our approach also works when travel times do not. Finally, we note that the arcs depicted represent distinct movements and do not overlap.

In Figure 5.5(a) we depict a solution to the AP(SNDRC) that violates both the resource bound constraints (the two cycles that are chosen represent itineraries for resources assigned to T3 when there is only one resource located at that terminal) and the 0-1 service constraints (both cycles model a resource executing the service arc departing from T3 in period 2 and arriving at T2 in period 4). Next, we depict in Figure 5.5(b) the addition of the arc from T1 at period 1 to T3 at period 2 to enable the unused resource at T1 to move to T3 where an excess number of resources are used. Next, in Figure 5.5(c) we show the arcs that are added to ensure that the subgraph may be decomposed into cycles, and in Figure 5.5(d) we illustrate the cycles that are extracted from that graph. While we do not illustrate this, the last step executed is to solve a minimum-cost multicommodity network flow problem to determine whether the flow balance constraints can be satisfied using only these two cycles. Finally, we note that the extracted cycles depicted in Figure 5.5(d) contain the same holding arc, as is allowed in our model.

5.4.4 Intensification and Diversification

Metaheuristics (Glover & Laguna, 1997) have long included intensification procedures, wherein a region of the solution space is explored deeply, and diversification procedures, wherein the search is directed towards regions of the solution space that have not been thoroughly searched. Because such procedures often enable the search to find high quality solutions in limited times we have included them in our solution approach. We first present our intensification procedure and then the diversification procedure.

For intensification, we introduce an exact optimization step into the search, wherein SNDRC($\tilde{\theta}$) is solved (for what is presumably a set of cycles, $\tilde{\theta}$, of very limited cardinality) with a commercial mixed integer programming solver. See De Franceschi *et al.* (2006); Archetti *et al.* (2008); Hewitt *et al.* (2010); Erera *et al.* (2012) for other examples of heuristics that find high-

quality solutions by solving a restriction of the original problem. To create the set of cycles $\tilde{\theta}$, our intensification procedure first chooses the cycles that appear in the last q solutions to AP(SNDRC). Then, a subgraph of $\mathcal{G}$ is created based on the arcs that appear in those cycles. Because some cycles may share arcs, this graph may not be Eulerian and thus we next add arcs to the subgraph to make it Eulerian with the procedure presented in Vu *et al.* (2013). Finally, a depth-first search-type approach is performed on this subgraph to extract cycles which are TMAX periods long. These extracted cycles are used to construct the set $\tilde{\theta}$ which is then used to create SNDRC($\tilde{\theta}$). This mixed integer program (MIP) is then solved with the value of the best-known solution as an upper bound on the objective function value and time limit t_{int} (in seconds). The intensification procedure is executed when an improved solution has not been found after a predefined number of iterations.

For diversification, because it is the cycles in the solution to AP(SNDRC) at a given iteration that dictate the structure of the resulting solution to SNDRC, we periodically modify the objective function of AP(SNDRC) to avoid frequently used cycles. To do so, we collect the number of times, fre_τ, each cycle τ appears in a solution to AP(SNDRC). Then, if the diversification condition is met for each cycle τ in the current solution of the approximated problem AP(SNDRC), we set the linearization factor $\rho_{uv}^{k\tau}(t)$ to $\rho_{uv}^{k\tau}(t)(1 + \epsilon * fre_\tau)$ where ϵ is an algorithm parameter. We continue to update $\rho(t)$ in this manner for a fixed number of iterations that is dictated by the algorithm parameter I_{max}^{diver}. Lastly, it is possible for two consecutive solutions of AP(SNDRC) to be the same, in which case the slope-scaling procedure will terminate. To continue the execution of slope-scaling when this occurs, we add a penalty value P to the linearization factors as seen in (5.12). In our experiments, P is set to the objective value of the current solution to AP(SNDRC).

$$\rho_{uv}^{k\tau}(t) = \begin{cases} \frac{P+F+\sum_{(i,j)\in\mathcal{S}} r_{ij}^{\tau} f_{ij}}{\sum_{k\in\mathcal{K}}\sum_{ij\in\tau} \tilde{x}_{ij}^{k\tau}}, & \text{if } \sum_{k\in\mathcal{K}}\sum_{ij\in\tau} \tilde{x}_{ij}^{k\tau} > 0 \\ \rho_{uv}^{k\tau}(t-1) & , \ otherwise \end{cases} , \forall \tau \in \theta,\ k \in \mathcal{K},\ (u,v) \in \tau \quad (5.12)$$

5.5 Computational Results

We next report on an extensive computational study that we performed in order to determine the effectiveness of the solution approach presented in Section 5.4 and to understand which of

Parameter	Description	Section	Value
λ	Number of initial cycles created for each commodity	5.4.1	3
I_{max}^{diver}	Number of diversification iterations	6.5.4	5
ϵ	The effect of frequency on linearization factor during diversification	6.5.4	1
t_{int}	Time limit for MIP solved during intensification	6.5.4	600 sec.
q	Number of iterations from which AP(SNDRC) solution cycles are used	6.5.4	1

Table 5.1: Algorithm parameter values

its features contribute to its effectiveness. We performed all experiments on a cluster of computers with 2 Intel Xeon 2.6 GHz processors and 24 GB of RAM that were running Scientific Linux 6.1. We used CPLEX 12.4 to solve linear and mixed integer programs. Initial calibration experiments on a small number of instances yielded the parameter values displayed in Table 5.1.

We executed the solution approach on four sets of instances. The first is a set of instances that are small enough that all cycles can be enumerated. We present characteristics of these five instances in Table 5.2. The number of resources at each terminal is a random number in the interval [5,10]. The second is the set of instances used in Andersen *et al.* (2011). This test set, with characteristics described in Table 5.3, includes 7 classes of 5 randomly generated instances based on a real-life case study in rail transportation planning, albeit with an increased number of terminals, services, time periods, and commodities than seen in the study. We label these instances classes 6-12 to maintain consistency with Andersen *et al.* (2011). The number of resources at each terminal is again a random number in the interval [5,10]. In these instances the resource fixed cost (F) is the driving factor in the cost structure, with the cost coefficient $f_{ij} = 0 \quad \forall (i,j) \in \mathcal{A}$. Thus, to see whether the performance of the algorithm is dependent on this cost structure we created a third set of instances that are exactly the same as those in Andersen *et al.* (2011) except that the cost parameters $f_{ij} > 0$ are set to a value that is proportional to the duration (in periods) of the service. Finally, to understand the impact of the parameter F on SSCG's performance we created a fourth set of instances, containing six copies of each instance described in Table 5.3, with each copy having a different value of F (one of 500; 1,000; 2,000; 3,000; 4,000; and 5,000).

Instance	$\|\mathcal{L}\|$	$\|\mathcal{S}\|$	$\|\mathcal{T}\|$	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$
1	5	10	15	75	225	20
2	5	15	20	100	400	25
3	5	15	25	125	500	25
4	5	15	15	75	300	100
5	5	15	15	75	300	200

Table 5.2: "Small" instances.

Instance Class	$\|\mathcal{L}\|$	$\|\mathcal{S}\|$	$\|\mathcal{T}\|$	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$
6	5	15	40	200	800	200
7	5	15	50	250	1,000	400
8	7	30	30	210	1,110	200
9	7	30	30	210	1,110	400
10	7	30	50	350	1,850	300
11	10	40	30	300	1,500	200
12	10	50	30	300	1,850	100

Table 5.3: Rail-based instances

5.5.1 Benchmarking against a MIP Solver

We first benchmark the performance of the solution approach against a commercial MIP solver (CPLEX 12.4) on instances that are small enough that the set of cycles, θ, can be enumerated in a reasonable amount of time. For these experiments, CPLEX was executed with an optimality tolerance of 1% and a time limit of one hour. In Table 5.4 we report results from experiments run on the instances described in Table 5.2. Specifically, we report the number of enumerated cycles for the instance ($|\theta|$) when performing a full enumeration, the optimality gap reported by CPLEX (CPLEX Opt gap), the number of cycles generated by SSCG (SSCG # cycles), a comparison of the number of cycles generated (% of cycles), the optimality gap (SSCG opt gap) associated with the solution produced by SSCG as measured against the dual bound produced by SSCG (recall SSCG begins by solving the linear programming relaxation of SNDRC with column generation), the optimality gap (SSCG opt gap CPLEX bound) associated with the solution produced by SSCG as measured against the dual bound produced by CPLEX, and the gap (SSCG gap with CPLEX) of the solution produced by SSCG as measured against the primal solution produced by CPLEX.

We see that while CPLEX was able to solve three of the instances to optimality in one hour, it struggled on the instances for which $|\theta|$ is very large. For instance 3 memory problems prevented CPLEX from solving the root node linear relaxation. However, SSCG was very competitive, producing (in less than thirty minutes) solutions that are on average only 1.28% worse. For the instances for which we know the dual bound produced by CPLEX is quite tight (instances 1, 4, and 5) we see that the average optimality gap for the solution produced by SSCG is 3.03%, suggesting that it is producing high-quality solutions. We also note that the dual bound produced by SSCG is roughly 1.5% weaker on average than the one produced by CPLEX. Finally, we note that while SSCG generated very few cycles it still produced high-

quality solutions. Thus we conclude that for these instances the approaches SSCG uses to generate cycles that are present in high-quality solutions worked well.

Instance	$\|\theta\|$	CPLEX opt gap	SSCG # cycles	% of cycles	SSCG opt gap	SSCG opt gap CPLEX bound	SSCG gap with CPLEX
1	5,520	0.80%	100	1.81%	5.01%	0.80%	0.00%
2	126,040	9.60%	238	0.19%	12.10%	12.10%	2.76%
3	1,655,800	n/a	275	0.02%	15.61%	n/a	n/a
4	8,985	1.00%	412	4.59%	2.05%	1.81%	0.82%
5	8,985	0.48%	1,242	13.82%	2.03%	2.00%	1.52%
Average	361,066	2.97%	453	4.09%	7.36%	4.18%	1.28%

Table 5.4: Results for "small" instances

We next benchmark the approach on a subset of the instances from Andersen *et al.* (2011), described in Table 5.3 and with $f_{ij} = 0$. Specifically, we chose two instances from each of the seven classes, reducing the number of periods to 20 in each instance to ensure all cycles could be enumerated. We also adjusted the origin and destination nodes of commodities whose original available or due periods exceeded 20. Lastly, we experimented on each of the six copies of an instance, one for F=500; 1,000; 2,000; 3,000; 4,000; 5,000. We again executed both CPLEX and SSCG for one hour and report the same gap-type results as discussed in Table 5.4, albeit averaged over instances with the same value of F. We see that SSCG is competitive with CPLEX, albeit its performance does degrade as F grows. As we will discuss later, some of SSCG's success can be attributed to its use of cycles that are present in the solution to the linear programming relaxation of the SNDRC. Because the linear programming relaxation provides a weaker bound as F grows (as is common in fixed charge-type models) we hypothesize that the cycles in its solution become less useful for producing high quality solutions.

F	CPLEX opt gap	SSCG opt gap	SSCG opt gap CPLEX bound	SSCG gap with CPLEX
500	1.08%	2.5%	2.2%	0.80%
1,000	1.51%	3.4%	3.1%	1.62%
2,000	1.95%	4.4%	4.0%	1.97%
3,000	2.32%	5.4%	4.8%	2.45%
4,000	2.35%	5.9%	5.2%	3.23%
5,000	2.21%	6.8%	5.9%	3.78%
Average	1.90%	4.74%	4.21%	2.31%

Table 5.5: Results for different cost structures

From the results in this section we conclude that SSCG is capable of producing high-quality solutions and is competitive with CPLEX when instances are small enough that the set θ can be enumerated in a reasonable amount of time.

5.5.2 Benchmarking against Column Generation-based Heuristic

We next turn our attention to the performance of SSCG on instances that are too large for the full set of cycles, θ, to be enumerated in a reasonable running time. For comparison with SSCG, we developed a column generation-based heuristic, which we call CGMIP-H, that is based on ideas that have been shown to be effective in the literature. CGMIP-H uses the column generation portion of SSCG to solve the linear programming relaxation of an instance, keeping the cycles generated, $\bar{\theta}$. The heuristic then solves the resulting MIP, SNDRC($\bar{\theta}$), with a commercial MIP solver (CPLEX 12.4).

We benchmark SSCG against CGMIP-H on the instances described in Table 5.3 wherein $f_{ij} = 0$. For these experiments we executed SSCG for both one and ten hours (labeled SSCG 1 Hour and SSCG 10 Hours) and executed CGMIP-H for ten hours (labeled CGMIP-H 10 Hours), but also recorded its progress after one hour (labeled CGMIP-H 1 Hour). For both executions of SSCG we also terminated its execution after 500 iterations of slope scaling or 100 iterations without finding an improved solution.

First, in Table 5.6 we report the number of instances (out of 35) for which each method was able to produce a feasible solution. We see that CGMIP-H struggled to produce feasible solutions, only doing so for 60% of the instances in one hour. On the other hand, SSCG was able to produce a feasible solution for every instance in one hour.

Next, in Table 5.7 we compare the quality of the solutions produced by SSCG when executed for one and ten hours with the solutions produced by CGMIP-H after one and ten hours. Note in Table 5.7 we are only averaging over instances wherein both methods produced a feasible solution. We report the average gap in solution quality over all 35 instances (column Gap) and the number of instances where SSCG produced an equivalent or better solution (# SSCG beat or tie). Finally, we report the average optimality gap (SSCG opt gap) for the solutions produced by SSCG. We see that our approach significantly outperformed the CGMIP-H heuristic, including producing in one hour solutions that are 4.5% better than CGMIP-H can produce in 10 hours.

	SSCG 1 Hour	SSCG 10 Hours	CGMIP-H 1 Hour	CGMIP-H 10 Hours
# Instances produced feasible solution (out of 35)	35	35	21	33

Table 5.6: Frequency of finding a feasible solution for rail-based instances, $f_{ij} = 0$

	CGMIP-H 1 Hour		CGMIP-H 10 Hours		
	Gap	# SSCG beat or tie	Gap	# SSCG beat or tie	SSCG opt gap
SSCG 1 Hour	-16.16%	9	-4.49%	5	10.31%
SSCG 10 Hours	-20.14%	13	-7.72%	10	7.79%

Table 5.7: Solution quality for rail-based instances, $f_{ij} = 0$

Because SSCG generates cycles throughout its execution (and after the linear relaxation of SNDRC is solved), we also executed CGMIP-H for 10 hours for each of the 35 instances where $\bar{\theta}$ contains all the cycles generated by SSCG. We found that, on average, in 10 hours SSCG produced a solution that was 2.70% worse than what CGMIP-H could produce. Because CGMIP-H performed much better with this expanded set of cycles we conclude that the cycles produced by SSCG after the column generation procedure are necessary to produce high-quality solutions. Because SSCG produced solutions that are of comparable quality to what CGMIP-H produced given the same time frame suggests that our approach is capable of producing high-quality solutions given the set of cycles it generates.

In Table 5.8 we report averages of various performance matrics collected when producing the SSCG 10 Hours results reported in the previous two tables. We see that the vast majority of execution time is spent trying to create a feasible solution from a solution to AP(SNDRC) (as discussed in Section 6.5.3). This suggests that to reduce the time SSCG requires to produce high quality solutions for these instances, this is the procedure that should be improved. At the same time because this procedure must generate cycles it is also very likely to change when different operational settings are modeled.

Time (seconds)	# Iterations	# Iterations to best solution	% Time solving LP relaxation via CG	% Time solving AP(SNDRC)	% Time creating feasible solution from AP(SNDRC) solution
14,418	166.51	67.00	25.57	11.52	62.91

Table 5.8: SSCG 10 Hours performance statistics

We next study in Tables 5.9 and 5.10 the performance of SSCG on the instances from Andersen *et al.* (2011) where the cost coefficients f_{ij} are positive. Because both methods found feasible solutions for fewer instances, we conclude that the instances with $f_{ij} > 0$ are, in a sense, more difficult. The CGMIP-H 1 Hour columns of Tables 5.7 and 5.10 also support this conclusion. We see that when only one hour of computational time is available SSCG is far superior to CGMIP-H. However, unlike the instances with $f_{ij} = 0$, SSCG did not do as well as CGMIP-H when 10 hours was allowed.

	SSCG 1 Hour	SSCG 10 Hours	CGMIP-H 1 Hour	CGMIP-H 10 Hours
# Instances produced feasible solution (out of 35)	32	35	20	28

Table 5.9: Frequency of finding a feasible solution for rail-based instances, $f_{ij} > 0$

	CGMIP-H 1 Hour		CGMIP-H 10 Hours		
	Gap	# SSCG beat or tie	Gap	# SSCG beat or tie	SSCG opt gap
SSCG 1 Hour	-45.32%	8	12.08%	0	17.11%
SSCG 10 Hours	-58.98%	14	4.20%	2	9.62%

Table 5.10: Solution quality for rail-based instances, $f_{ij} > 0$

5.5.3 Robustness of SSCG

We next study how sensitive SSCG's performance is to two instance characteristics, the distribution of resources across the network, and the magnitude of the parameter F. For the results in this subsection we again focus on the rail-based instances described in Table 5.3 (and with $f_{ij} = 0$). To achieve the results discussed in this subsection we executed SSCG for 10 hours and we report quality of the solution produced by SSCG against the dual bound it produced.

To understand the impact of the distribution of resources across the network on SSCG's performance, we consider three cases: (1) each terminal is assigned the same number of resources (recall that we only differentiate resources by the terminal to which they are assigned), (2), the number of resources assigned to a terminal is proportional to the total volume of all shipments originating at that terminal, and, (3) the number of resources assigned to a terminal is determined randomly. We note that in each of these cases the instances are more constrained than

those reported on in the previous tables as the total number of resources assigned to terminals is much lower. We report in Table 5.11 the results of our experiments for each of the three cases, including the (average) optimality gap of the solutions produced (as measured against the dual bound produced by SSCG) and the number of instances wherein it produced a feasible solution. We see that the performance of SSCG was robust with respect to the distribution of resources as the optimality gaps and number of instances wherein the approach was able to produce a solution are roughly the same across all three cases. We also report in Table 5.11 the performance CGMIP-H and see that while the quality of the solutions it produces did not vary significantly from one case to another the number of instances for which it could produce a solution did.

	Equal distribution		Proportional distribution		Random distribution	
	Opt gap	# instances found solution	Opt gap	# instances found solution	Opt gap	# instances found solution
SSCG	7.26%	32	7.30%	32	7.19%	29
CGMIP-H	7.30%	31	7.23%	28	7.58%	23

Table 5.11: Results for different distributions of resources

Next, to understand the impact of the parameter F on SSCG's performance, we ran experiments on the six copies of each instance described in Table 5.3, with each copy having a different value of F (one of 500; 1,000; 2,000; 3,000; 4,000; and 5,000). We then executed SSCG for 10 hours and report in Table 5.12 the average optimality gap reported by SSCG for each value of F. We note that SSCG was able to produce a feasible solution for every instance and each value for F. We see that the optimality gap grows as F grows and then plateaus. We surmise that the gaps increase as F increases because the gap values are derived from the value of the linear programming relaxation of an instance, which likely weakens as fixed costs increase.

F	500	1,000	2,000	3,000	4,000	5,000
Opt gap	2.82%	4.41%	5.90%	7.36%	7.82%	7.79%

Table 5.12: Results for different values of F

5.5.4 Understanding Why the Approach Works

Because many design choices were made when developing SSCG, we next seek to understand the impact of those choices. Specifically, we next study the impact of using the cycles generated by solving the linear relaxation of SNCRC with column generation, initializing the linearization factors based on the solution to the linear relaxation of SNDRC, and using the diversification and intensification procedures.

To study the impact of using the cycles generated with column generation on the quality of the solution produced by the approach we ran the approach on the instances from Andersen *et al.* (2011) but skipped solving the SNDRC with column generation. Instead we began the approach with the cycles generated through the multiple shortest paths procedure described at the end of Section 5.4.1. We found that using the cycles generated by column generation enabled the approach to find solutions that were 5.16% better on average.

We next focus on understanding whether setting $\rho_{ij}^{k\tau}(0) = 1$ when τ appears in the solution to the linear relaxation of SNDRC enables the approach to find higher quality solutions. To do so, we ran the approach on the instances from Andersen *et al.* (2011) but set $\rho_{ij}^{k\tau}(0) = \left(F + \sum_{(i,j)\in S} r_{ij}^{\tau} f_{ij}\right) \Big/ \sum_{(i,j)\in S} r_{ij}^{\tau} u_{ij} \quad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K}, \tau \in \theta$. We found that initializing the linearization factors based on the solution to the linear relaxation of SNDRC enabled the approach to find solutions that were 3.84% better on average.

Lastly, we focus on understanding the impact of using the diversification and intensification procedures described in Section 6.5.4. To do so, we ran the algorithm on each of the 35 instances from Andersen *et al.* (2011) two more times. The first time, neither the diversification or intensification procedures are used. The second time, the diversification procedure is used. Comparing the quality of the solutions produced when the diversification procedure is used with when it is not, we saw that solutions were 3.58% better on average when the diversification procedure is used. Next, comparing the quality of the solutions produced when only the diversification procedure is used with when both procedures are used, we saw that solutions were 6.41% better on average when both the diversification and intensification procedures were used.

5.6 Conclusions and Future Work

We extended existing service network design models that recognize the need to manage facility-based resources when routing commodities to also recognize that there are limits on the number of resources available at each terminal. In this paper we focused on a setting wherein the only rule governing the movements of a resource is that it must ultimately return to the terminal to which it is assigned. However, because the rules governing the movements of such resources may vary greatly from one transportation setting to another, we proposed a model and solution approach that are adaptable.

Due to the cycle-based nature of the model proposed, there can be a huge number of variables for instances of even modest size, too many to enumerate in a reasonable period of time. Thus, for a solution approach to produce good solutions it must both generate cycles that appear in high-quality solutions and use them in an effective manner. We presented a solution approach that combines techniques from multiple research areas to perform both of those tasks. We next presented an extensive computational study to demonstrate that the method can produce high quality solutions.

We demonstrated the effectiveness of the approach by comparing the quality of the solutions it produced with those produced by a state-of-the art commercial solver. To understand whether the approach is capable of generating "good" cycles, we first considered instances that were small enough that all the cycle-based variables could be enumerated and that a commercial solver was able to (usually) solve to optimality. Our experiments indicated that our approach, while only generating a small fraction of the possible number of cycles, was still able to produce high quality solutions (as measured against the near-optimal solutions produced by the commercial solver).

We next benchmarked our approach on a set of realistically-sized instances from the literature. Here, because all variables could not be enumerated, we developed a column generation-based heuristic against which we benchmarked our approach. We found that our approach produced both high-quality solutions (as measured against the dual bound our approach produces) and those solutions were typically better than those produced by the other heuristic. We conclude from these experiments that our approach is capable of producing the cycles necessary to produce a high quality solution and of using those cycles in an effective manner. Finally, like many solution methods, ours relies on various parameters and design choices. We

finished our computational study with experiments that showed how our choices are critical to the success of the approach.

The primary focus of our next work in this area is to extend the model to also allocate resources to facilities. Whereas the model we have studied focuses more on integrating operational-type decision-making into tactical planning formulations, this new model will instead recognize operational realities when making strategic (fleet dimensioning) and tactical decisions. While similar to location routing and location-allocation problems, we believe such a model has not yet been studied and could have many practical uses, such as helping a trucking company determine driver hiring strategies.

Acknowledgments

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grant program and by the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through their infrastructure grants and of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

6. Resources Location Issues Service Network Design

Summary of the Chapter 6

This chapter of the thesis presents our third work which will be presented at XII International Symposium On Locational Decision, 2014 with the following reference information:
T. G. Crainic, M. Hewitt, M. Toulouse, D. M. Vu: Integration Location - Service Network Design, XIII ISOLDE Symposium - Naples/Capri (Italy) 2014.

Abstract

In this chapter, we study the integration of resources allocation in service network design. We address the problem setting where a single unit of resource is required to operate a service, resources are assigned to terminals to which they must ultimately return, there is a finite number of resources assigned to each terminal, and the length of resource circuits is restricted. Moreover, a number of fleet utilization issues must be addressed at the beginning of the season: 1) repositioning resources among terminals to account for shifts in demand patterns; 2) acquire (buy or long-term rent) new resources and assign them to terminals; 3) outsource particular services. We present an integrated formulation combining these selection-location and scheduled service design decisions. The mixed-integer formulation is defined over a time-space network, the initial period modeling the location decisions on resource acquisition and positioning, while the decisions on service selection and scheduling, resource assignment and cycling routing, and demand satisfaction being modeled on the rest of the network. We also present a matheuristic solution method combining slope scaling and column generation, discuss its algorithmic performance, and explore the impact of combining the location and design decisions in the context of consolidation carrier service design.

Keywords: *Discrete location, service network design, tactical planning, consolidation transportation, matheuristics.*

6.1 Introduction

Service network design issues (Crainic, 2000, 2003; Crainic & Kim, 2007; Crainic & Laporte, 1997) which are part of tactical planning for consolidation-based carriers, have been studied intensively to provide solutions for tactical issues. Service network design issues aim to determine which services should be used and how carriers allocate resources to these services in the most efficient and profitable ways. Before, resources management issues were not considered jointly with the determination of services. That means the company first needs to determine a set of services and then the carrier will assign resources to selected services to find a feasible transportation planning for a specific demand pattern. This could lead to the increase of the operation cost of the carrier.

To obtain a more efficient transportation plan and optimize the operation of resources, resources management issues are then integrated into service network design model. Let us recall how resources management was integrated into service network design. The most basic resources management issue that was integrated into service network design model is the consideration of the design-balance constraints and the maximum number of available resources. Design-balance constraints ensure that the number of resources entering a terminal and leaving the same terminal is the same. The design-balance constraints issue has been considered in Smilowitz *et al.* (2003) in addressing the vehicle routing problem and in Lai & Lo (2004) in addressing a ferry transportation problem. This type of constraints is first considered formally in the article Pedersen *et al.* (2009). The maximum number of available resources of a carrier, i.e. a carrier has 20 unit of resources, restricts the number of resources used for each transportation plan (see Lai & Lo (2004)). Service network design models were then extended to capture more complicated resources management issues such as route-length constraints, coordination of fleet, service frequency. The route-length constraint limits on the duration of resource routes and it reflects practice in several transportation firms. For example, work rules for truck drivers require the satisfaction of hours-of-services rules described in US - Hour of Service rules (see zHO (2013)) and the carrier may set the route-length following these rules. The coordination of multiple fleets (see Andersen *et al.* (2009b)) refers to the management of many vehicles with different characteristics so that these vehicles could operate efficiently together to transport shipments from their sources to their destinations. The consideration of service/resource frequency (see Andersen *et al.* (2009a, 2011)) aims to tradeoff between the running with free space in low-peak periods and not serving all the potential demand in high-

peak periods. Most recently, Crainic *et al.* (2013) introduced a new service network design model with resources management by considering the resource limitation issues at terminals. Previous works (e.g. Andersen *et al.* (2011)) consider the total number of available resources at all terminals. However, resources, especially high-valued resources such as locomotives, are usually associated with terminals. Crainic *et al.* (2013) enhances resources management issues by considering the limitation of resources at each terminal.

This chapter enriches the resources management issues by introducing resource acquisition and positioning in service network design models. We consider some questions:

- How should the carrier reposition its resources among terminals to adapt to the change of demand pattern (e.g. in the end of the season) and to optimize the objective function?
- How should the carrier buy or rent resources and allocate them to terminals so that all demands are transported while optimizing the objective value?
- How should the carrier outsource the operation of services to another carrier so that all demands are transported while optimizing the objective value?

We present some scenarios in which these questions are relevant. For the first question, the reposition may happen when the demand pattern changes, so the carrier needs to reallocate resources to prepare for these changes. With the new demand pattern information, the question is what is the best way to reallocate these resources?

For the second question, we consider a scenario in which the carrier wants to increase its capacity. In order to achieve its goal, the carrier needs to acquire (buy or rent) some resources. The installment of resources to terminals requires some costs (i.e. cost to buy and install new resources). The question is that where the carrier should allocate these resources so that they can efficiently service all forecast demands while the cost is minimum.

The last question is the scenario in which the carrier wants to outsource the transportation of shipments to another carrier. The question about the information of the outsourced shipments (volume, origin, destination, cost, etc.) should be considered.

We highlight the difference with respect to location issues between our current problem and location-routing problems. In location routing, the locations of terminals decide the location of resources and then decide the routes of these resources. In our problem, locations of terminals

are fixed and we decide the locations of resources through allocation operations. Another difference is that the underlying transportation operation is network service, i.e. goods are transported through a network of terminals as opposed to vehicle routing where a truck departs from a depot, delivers goods to the customer and then returns the depot from which it departs. This leads to differences in the application of the two problems. The location-routing is usually applied to a strategic problem where one wants to build facilities and establish links and routes connecting facilities and nodes to satisfy customers demand. The application of our problem is in the strategic-tactical domain where one wants to allocate resources among a set of terminals and then establish a schedule to transport demands among terminals. Additionally, in our problem, we need to select services (with associated information such as origin of service, destination of service, capacity of service, fixed cost pay for the use of service, etc.) to route goods from terminals to terminals while in location-routing, routes are defined on links/arcs without additional information.

The contributions of this chapter are two-fold. First, we propose a model, named LWSND, that integrates resource acquisition and positioning operations into the service network design model. The model allows to reposition resources among terminals to account for shifts in demand patterns; to acquire (buy or long-term rent) new resources and assign them to terminals; to outsource particular services. We present an integrated formulation combining these selection-location and scheduled service design decisions. We present a mixed-integer formulation defined over a time-space network, the initial period modeling the location decisions on resource acquisition and positioning, while the decisions on service selection and scheduling, resource assignment and cycling routing, and demand satisfaction being modeled on the rest of the network. Second, we present a matheuristic solution method combining slope scaling and column generation, discuss its algorithmic performance, and explore the impact of combining the location and design decisions in the context of consolidation carrier service design.

6.2 Problem Description

To start, let us examine the transport issues of a company. The company processes a family of products. Each part of a product is produced and handled in one of its plants and then to be moved to another plant for further processing. To avoid the overload of storage and work at each plant, the manufacturer identifies the volume of each material that the plant can process at each period. To make the manufacturing process function perfectly, the manufacturer needs

to guarantee the transportation of products between plants. This task includes the allocation of vehicles to plants and the identification of routes for vehicles materials between plants with the consideration of the volume of materials, the available and due time of each material, the capacity of each vehicle, etc. The assignment of vehicles to plants may require the repositioning of vehicles from plants to other plants due to the demand level at each plant. In the case of lacking vehicle capacity, i.e. vehicles are busy or in the maintenance/broken status, the company may acquire or rent vehicles from an external supply company to have enough capacity for the transportation tasks. In addition to transport insourcing, the company also considers transport outsourcing in which parts of materials, products are delivered by external carriers. The company will identify from where to where the transport outsourcing is done and the volume and kind of material moved by outsourced services. Combining the transportation problem and the positioning of resources, the company wants to identify a transportation plan which delivers materials in time and minimizes the total cost while allowing an efficient use of vehicles. The plan should indicate how to reposition vehicles between plants, how to acquire or rent additional vehicles if needed and how to outsource part of the transport to a carrier.

We model the above problem as a scheduled service network design with resource capacity management. Briefly, a transportation network has a set of terminals (plants in the above description) in which many kind of goods are needed to be transported between them. The transport of goods requires resources (vehicles in the previous case) which include resources of the company or resources acquired from external suppliers. To simplify the problem, we assume that resources are homogenous and share same set of characteristics (volumes, speeds, etc.). We also assume resources can operate any kind of goods, materials. The positioning and acquisition of capacity include:

- Resource repositioning. The plan models the reposition existing resources among terminals that need to be worked into the current plan.

- Resources acquisition. The plan models the buy/rent of resources and we want to know where we should allocate these resources to terminals to efficiently transport forecast shipments.

- Services outsourcing. The plan models the outsources the transportation of some shipments to carriers.

The term *services* is used to describe a transportation activity between two terminals. Each

service is defined in terms of origin and destination of the service, the departure time, the arrival time and the volume of goods that the service can transport. We assume each unit of resource is needed to operate a service.

Demands which are set of shipments are described through a set of *commodities*. Information of each commodity includes its origin terminal, its destination terminal, and its associated volume. Commodities also have specific time when they are available to transport at their origins and deadline time when they need to be available at their destinations.

We define a *route* as a series of services, waiting and repositioning activities related to a resource. Resources management constraints are enforced on routes to reflect carrier's rules. For example, *route-length requirement* ensures that the length (in duration) of each route cannot be longer than a predefined length; we set this value to the schedule length (i.e. a day, a week, etc.).

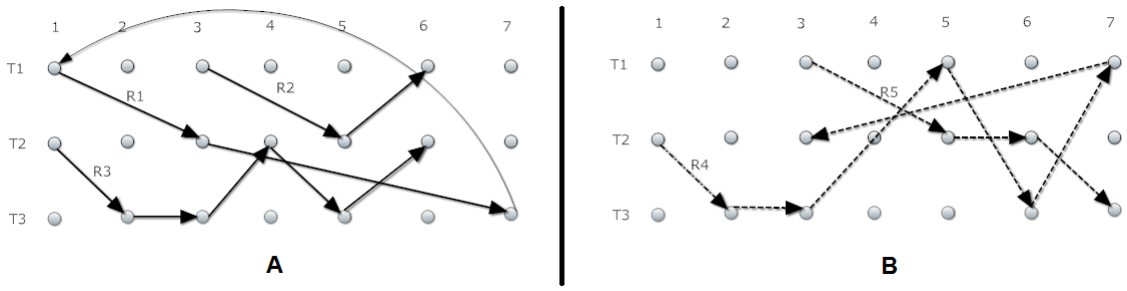


Figure 6.1: Valid and invalid routes.

Figure 6.1 presents an example of valid routes (A) and invalid routes (B). A route is valid if it has a same original and destination terminal, and its length is smaller than the length of the planning schedule. There are three valid routes in Figure 6.1A. Invalid routes in Figure 6.1B are due to the violation of the route-length constraints and the violation of the condition that resources must begin and end at the same terminal.

We want to obtain a plan which is a set of selected services to deliver shipments and a set of resource routes covering these services to identify the utilization of resources. The plan needs to satisfy all shipments, service and resource management constraints.

To compare and evaluate plans, we assume that costs for resource reposition and acquisition/renting activities, costs for the operations of resources, costs for using services and costs

for transporting commodities. The objective is to find a plan that minimizes the total cost which includes the cost of positioning and acquiring resources, the cost of operating resources, the cost of routing commodities, while satisfying all the shipments and resources management constraints.

Many costs are associated with a schedule. To simplify the problem, we only assume costs for main factors. For the reposition and acquisition of resources, the cost depends on the type of operation, its origin and its destination. For high value resources such as vehicles, locomotives, reposition cost is much smaller than buying or renting cost. Whereas, for low value resources such as containers, reposition over a long distance may cost more than buying or renting new ones. The cost paid for the use of each resource is considered. If resources are vehicles, this cost may present the value of vehicles, or expenses related to the utilization of vehicles (e.g. depreciation cost of resources, maintenance fee, etc.). We assume a cost for the operation of each service by resources. This cost may present expenses such as fuel, ticket, etc. needed for the operation of resources on this service. Related to shipments and commodities, the cost may present the damage of commodities transported over a service if we take into account the damage regarding to commodities or the penalty value in time regarding to each commodity if we want to minimize of the delivery time for each commodity.

6.3 Literature Review

The research problem relates two different aspects: the allocation of resources to terminals and the routing of commodities using allocated resources. The allocation of resources can be seen as a location issues of resources, and the routing of commodities in our problem is a tactical service network design problem. We review related problems in the facility location domains and service network design domains.

The review of Contreras & Fernandez (2012) provides a unified view of combined location - network design problems. Melkote & Daskin (2001b) describe an uncapacitated model that optimizes facility locations and the design of the underlying transportation network. The model describes the need to open several facilities and several design arcs to route the demands from origins to the nearest open facility. The goal of the model is to optimize the cost of opening facilities, opening services and routing cost. The problem is then transformed into an uncapacitated fixed cost network design problem and then can be solved efficiently using a mixed

integer programming solver. Later, Melkote & Daskin (2001a) introduce a combined facility location/capacitated network design problem in which facilities have constraining capacities on the amount of demand they can serve. We would notice that all these two models only consider the facility location issues with an underlying transportation network without any resource information.

To the best of our knowledge, resources management has not been considered in location - network design problems. We return our attentions to service network design problems with resources management. First papers such as Smilowitz *et al.* (2003); Lai & Lo (2004) described the most basic requirement named the design-balance constraints which ensure the number of services entering and leaving a node to be equal. Each service is assumed to be operated by one unit of resource and this characteristic leads to the balance of resources at each node. This property is also held for many other papers which consider resources management in their formulation. The proposed solution methods were complicated approaches, e.g. adhoc rounding techniques, adhoc heuristics. Pedersen *et al.* (2009) studied the compact formulation of the network design formulation with design-balance and stated that even the search for feasible solutions in this case is "far from trivial". They proposed the first two-phase tabu-search method to solve this problem. The first phase explores solutions that satisfy flow constraints but not design-balance constraints. The second phase converts a solution of the first phase into a feasible solution by a path-based neighborhood heuristic search. The quality of the solution much depends on the second phase and usually, it requires many iterations to obtain a feasible from a candidate infeasible solution. Then, Vu *et al.* (2013); Chouman & Crainic (2013) proposed more efficient solution methods to solve this problem. Vu *et al.* (2013) proposed an approach that shows how one can efficiently convert a non-feasible solution (which satisfies the flow constraints but not design-balance constraints) into a feasible solution using a minimum cost maximum flow procedure. This procedure solves the difficulty that Pedersen *et al.* (2009) met in their solution method. The procedure is integrated into a three-phase matheuristic which combines tabu-search, path-relinking and exact optimization and the new solution approach is proved to be efficient in finding good feasible solutions. In particular, the minimum cost maximum flow model which is proved to be efficient was then used in their other papers to solve design-balance constraints for much larger service network design problem instances (contain 300 or 400 nodes - Crainic *et al.* (2013)). Chouman & Crainic (2013) proposed a cutting plane approach which could obtain good feasible solutions with short running time. But there is still a question whether such method can handle design-balance constraints for a large network. The design balance property offers cycle-based formulations in addition to arc-based formu-

lations of service network design problems having these constraints. Andersen *et al.* (2009a) compared cycle-based formulations with arc-based formulations and pointed out the promising of cycle-based formulations in the sense of algorithmic development. The experiments also showed that the cycle-based formulations could provide better lower bound than the arc-based formulations. Later, Andersen *et al.* (2011) presented the first cycle-based branch-and-price solution method to solve this problem. Crainic *et al.* (2013) added the study of the resources management issues by introducing a new service network design model which considers limitations on how many resources are available at each terminal. The authors present a solution approach that includes column generation, slope-scaling, heuristics and exact optimization, and that presents a new solution approach for service network design problems. The experiment showed that the solution approach is able to find good feasible solutions and demonstrates the feasibility of using minimum cost maximum flow problem to solve design-balance constraints for large networks.

6.4 Problem Formulation

The problem is modeled over a time-space network. We suppose that the planning horizon is divided into TMAX periods; let $\mathcal{T} = \{1, 2, ..., \text{TMAX}\}$ be the set of time-periods. Let $\mathcal{L}$ be the set of possible terminals; for each terminal $l \in \mathcal{L}$, we denote l_t as the copy of terminal l at time period t for each $l \in \mathcal{L}$. We define $\mathcal{N}$ as the set of terminals with time-period information, $\mathcal{N} = \mathcal{L} \times \mathcal{T} = \{l_t | l \in \mathcal{L}, t \in \mathcal{T}\}$. Services are provided among terminals with specific time. For two nodes u_i, $v_j \in \mathcal{N}$, a potential *service arc* connects nodes u_i and v_j if there exists a service that would depart from terminal u at time-period i to arrive at terminal v at time-period j. Let us define S as the set of possible services defined over $\mathcal{N}$. For each $s \in \mathcal{S}$, u_s denotes the capacity offered by service s. We present the capacity of each service through the capacity of the associated resource but the formulation will be more complicated. In this study, we assume the capacity of service is fixed. The length or duration of the service s which departs from terminal u at time period i and arrives at terminal v at time period j is given by $(j - i + \text{TMAX}) \bmod \text{TMAX}$. We denote t_s (or t_{ij}) as the duration (in periods) of service s (or service between two node (i, j)).

The problem is defined by using a time space network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N} = \mathcal{L} \times \mathcal{T}$ and $\mathcal{A} = \mathcal{S} \cup \mathcal{H}$, where $\mathcal{H}$ denotes the set of *holding arcs*. A holding arc $a \in \mathcal{H}$ is an arc connecting nodes of the same terminal with consecutive time-periods or the last period

and the first period. In other words, a holding arc a must have the form $a = (l_t, l_{t+1})$ or $a = (l_{\text{TMAX}}, l_1)$ with $l \in \mathcal{L}$. We assume holding arcs have an infinite capacity for both flows and resources. Figure 6.2 shows an example of a time-space diagram for a four-terminal system and a seven-period planning horizon. The four terminals are named T1, T2, T3 and T4. In Figure 6.2, dotted arcs connecting different terminals represent all possible services that can be provided; dotted arcs between nodes of the same terminals are *holding arcs* representing the choice of keeping vehicles and loads at terminals for one time period. They present waiting activities of commodities and resources at terminal.

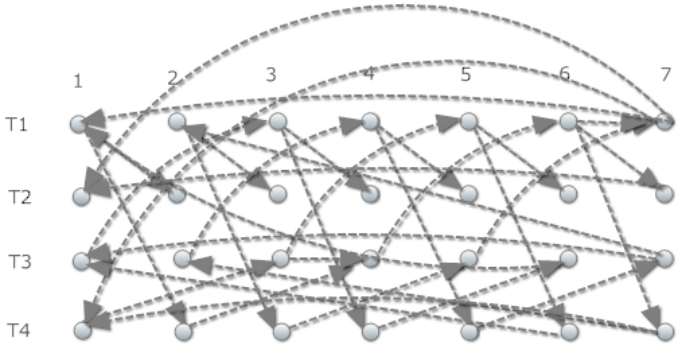


Figure 6.2: A time-space network with potential services.

We extend the classic time-space network design to represent the allocation activities of resources. Reposition activities can be represented through arcs connecting terminal l at an artificial time-period 0 and terminal l' at time-period 1 for any l, $l' \in \mathcal{L}$. Acquisition or renting of resources can be seen as a special reposition activity from an external supply company to local terminals. We use the notation e to represent an external supplier company. The arcs between the external supply node e at time period 0 and terminals in $\mathcal{L}$ at time period 1 represent buying/renting resources. We name the part of the extended time-space network related to the allocation of resources the ***resource allocation layer***, and the part related to the set of potential services over the planning horizon the ***network design layer.*** We define I_l with $l \in \mathcal{L}$ as the number of available resources at terminal $l \in \mathcal{L}$. We model buying/renting resource options as a reposition operation from external suppliers to terminals in $\mathcal{L}$. We define $\mathcal{L}^+$ as the union set of all terminals in $\mathcal{L}$ and the set of external suppliers.

Figure 6.3 is the extension of the time-space network in Figure 6.2 to represent allocation activities. In Figure 6.3, we assume only one external supply node e (rectangle notation) and four terminals T1, T2, T3 and T4. Note that the model can conceptually handle multiple

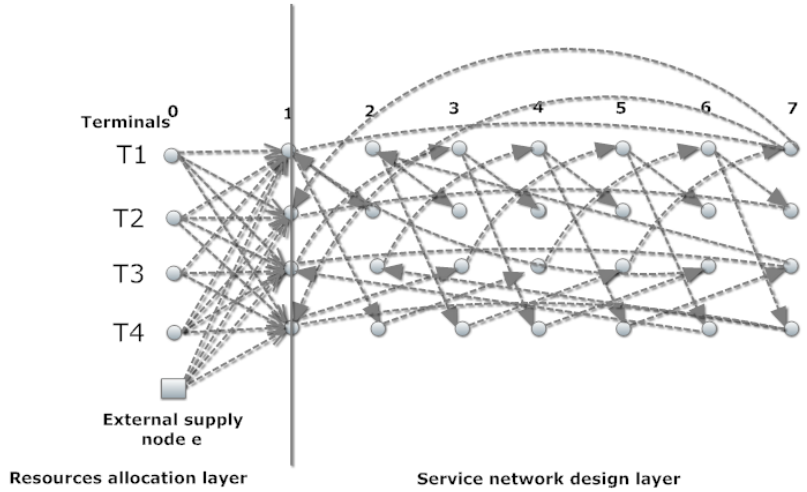


Figure 6.3: Time-space service network with the allocation of resources.

external suppliers. The arcs between e at time-period 0 and terminal T1, T2, T3 and T4 at time period 1 represent acquisition activities and the arcs between period 0 and period 1 of terminals T1, T2, T3 and T4 represent reposition activities.

We denote by $\mathcal{K}$ the set of commodities that represents possibly different products that need to be moved at particular times between particular origin and destination terminals. Each commodity $k \in \mathcal{K}$ requires that a certain quantity w^k be moved from origin node $o(k)$ to destination node $d(k)$ in the time-space network.

We model five types of costs. The first is the *resources allocation fixed cost*, $H_{ll'}$, of resource from the terminal l to terminal l' in $\mathcal{L}$ or from an external supplier $l \in \mathcal{L}^+$ to a terminal $l' \in \mathcal{L}$. The second is the fixed cost associated with each terminal, denoted as F_l - *operating resources fixed cost*. F_l may present depreciation cost of each used resource. The third is the *services fixed cost* associated with the execution of a service between two nodes i and j in the time-space network denoted by f_{ij}. The fourth is the fixed cost f^e_{ij} associated with the outsource of the service (i, j) to an external company. Finally, the fifth is the variable cost associated with the commodity k traveling on arc $(i, j) \in \mathcal{A}$, it is denoted by c^k_{ij} - *unit flow cost*.

Because of the design-balance condition that requires the same number of services, and thus resources in our problem, coming into and getting out of terminals, these conditions are fulfilled by resources traveling in cycles (which also reflects the usage of resources returning to base); therefore, we will model the problem based on cycles. We have no special rules on cycles apart from the route-length requirement. Each cycle can visit a terminal many times

while the route-length constraint is not violated. Following Crainic *et al.* (2013), we assume that the length of each cycle equals to TMAX by adding holding arcs if necessary.

Let θ_{lt} represent the set of cycles corresponding to the routes of resources departing from terminal $l \in \mathcal{L}$ at period $t \in [1, \text{TMAX}]$ in the time space network $\mathcal{G}$ and satisfying the route-length constraints. Define θ_l as the set of cycles τ corresponding to the routes of resources departing from terminal $l \in \mathcal{L}$, or $\theta_l = \cup_{t=1}^{\text{TMAX}} \theta_{lt}$. Let $\theta = \cup_{l \in \mathcal{L}} \theta_l$ denote the set of possible cycles in the network $\mathcal{G}$ for all terminal $l \in \mathcal{L}$. We define r_{ij}^{τ} as a parameter that indicates whether service arc (i, j) belongs to cycle τ.

We use four different sets of variables to model decision information of the problem. The first, denoted $h_{ll'}$, is the number of resources repositioned between two terminals l and l' with $l, l' \in \mathcal{L}^+$. If $l = l'$, h_{ll} is the number of resources staying at its original terminal. The second, z_τ, is a binary variable that indicates whether cycle $\tau \in \theta$ is selected. The third, y_{ij}^e, is a binary variable indicating whether the service (i, j) is operated by an external resource. Lastly, the fourth, x_{ij}^k, is a continuous variable that indicates the amount of commodity $k \in \mathcal{K}$ that flows on arc $(i, j) \in \mathcal{A}$.

The formulation of the service network design with resource allocation, named LWSND, is:

$$\begin{aligned} \text{minimize} \sum_{l,l' \in \mathcal{L}^+} H_{ll'} h_{ll'} + \sum_{l \in \mathcal{L}} F_l \left(\sum_{\tau \in \theta_l} z_\tau \right) \\ + \sum_{(i,j) \in \mathcal{A}} f_{ij} \left(\sum_{\tau \in \theta} r_{ij}^{\tau} z_\tau \right) + \sum_{(i,j) \in \mathcal{A}} f_{ij}^e y_{ij}^e + \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k \end{aligned} \quad (6.1)$$

subject to

$$\sum_{l' \in \mathcal{L}^+} h_{ll'} = I_l, \quad \forall l \in \mathcal{L}^+, \tag{6.2}$$

$$\sum_{\tau \in \theta_l} z_\tau - \sum_{l' \in \mathcal{L}^+} h_{l'l} \leq 0, \quad \forall l \in \mathcal{L}, \tag{6.3}$$

$$\sum_{\tau \in \theta} r_{ij}^\tau z_\tau + y_{ij}^e \leq 1, \quad \forall (i,j) \in S, \tag{6.4}$$

$$\sum_{k \in \mathcal{K}} x_{ij}^k \leq u_{ij} \left(\sum_{\tau \in \theta} r_{ij}^\tau z_\tau + y_{ij}^e \right), \quad \forall (i,j) \in S, \tag{6.5}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^k = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k) \\ -w^k & \text{if } i = d(k) \end{cases} \quad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \tag{6.6}$$

$$h_{ll'} \in \mathbb{N}, \quad \forall l, l' \in \mathcal{L}^+, \tag{6.7}$$

$$z_\tau \in \{0,1\}, \quad \forall \tau \in \theta, \tag{6.8}$$

$$y_{ij}^e \in \{0,1\}, \quad \forall (i,j) \in \mathcal{S}, \tag{6.9}$$

$$x_{ij}^k \geq 0, \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{K}. \tag{6.10}$$

The objective function (6.1) minimizes the cost of the allocation of resources to terminals $\sum_{l,l' \in \mathcal{L}^+} H_{ll'} h_{ll'}$ through the reposition, buying/renting operations, the cost of using resources $\sum_{l \in \mathcal{L}} F_l \left(\sum_{\tau \in \theta_l} z_\tau \right)$, the cost of renting services $\sum_{(i,j)\in\mathcal{A}} f_{ij}^e y_{ij}^e$, the cost of operating services to route commodities $\sum_{(i,j)\in\mathcal{A}} f_{ij} \left(\sum_{\tau \in \theta} r_{ij}^\tau z_\tau \right)$, and finally the cost of routing commodities $\sum_{k \in \mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k$ from their origins to their destinations.

Constraint set (6.2), named ***initial resources constraints***, ensures that the total number of resources allocated from a terminal to other terminals in $\mathcal{L}$ equals to the initial number of resources of this terminal. Constraints set (6.3), named ***resource bound constraints***, ensures that the number of cycles departing from each terminal does not exceed the number of resources at this terminal after the reposition. Constraint set (6.4), named ***service constraints***, ensures that there is at most one resource operating a service at a same moment. Constraint set (6.5), named ***weak forcing constraints***, ensures that the total volume of all commodities routed over a service arc cannot exceed the total capacity provided by resources operating that service. Constraint set (6.6), named ***flow balance constraints***, ensures that each commodity is routed

from its origin node to its destination node. Lastly, constraint sets (6.7), (6.8), (6.9) and (6.10) define the domains of the variables.

The formulation LWSND addresses what we described in Section 2. The transportation of commodities is enforced through flow variables x_{ij}^k and constraint set (6.6). The capacity of each selected service is satisfied by constraint set (6.5). The balance of services at each node is ensured through the design-cycle variables z_t and the constraint set (6.4). The reposition of resources between terminals are modeled through $h_{ll'}$ with $l, l' \in \mathcal{L}$ while we model the acquisition or renting of resources from external supplier to terminals through variables $h_{ll'}$ with $l \in \mathcal{L}^+$ and $l' \in \mathcal{L}$. Constraint set (6.2) ensures that the number of resources repositioned from each terminal does not exceed the number of resources available at that terminal. The outsource of services is modeled by y_{ij}^e variables.

Reformulation of the LWSND We now present another formulation of LWSND, named rwLWSND, in which there are no resources allocation variables $h_{ll'}$; and cycles are extended to include repositioning information. This formulation is used to develop the approximation model in our proposed solution approach described in the following section.

The new binary variables, $z_{ll'}^\tau$, denote that the cycle $\tau \in \theta_{l'}$ is operated by a unit of resources initially located at terminal l where $l' \in \mathcal{L}$ and $l \in \mathcal{L}^+$. It means this resource departs from terminal l, arrives at terminal l' and then performs its tasks following cycle τ. Note that $z_{ll'}^\tau = 0$ if $\tau \notin \theta_{l'}$. We have the constraint $z_\tau = \sum_{l' \in \mathcal{L}^+} z_{ll'}^\tau$ for each $\tau \in \theta_l$ or $z_\tau = \sum_{l,l' \in \mathcal{L}^+} z_{ll'}^\tau$ because $z_{ll'}^\tau = 0$ if $\tau \notin \theta_{l'}$ by definition.

We now define the relation between $h_{ll'}$ and $z_{ll'}^\tau$. Because the reposition cost $H_{ll} = 0$, so $\sum_{l \in \mathcal{L}^+} H_{ll} h_{ll} = \sum_{l \in \mathcal{L}^+} H_{ll} \left(\sum_{\tau \in \theta} z_{ll}^\tau\right) = 0$, and $\sum_{l,l' \in \mathcal{L}^+} H_{ll'} h_{ll'} = \sum_{l,l' \in \mathcal{L}^+} H_{ll'} \left(\sum_{\tau \in \theta} z_{ll'}^\tau\right)$. The reformulation for LWSND, rwLWSND, is:

$$\min \sum_{l,l' \in \mathcal{L}^+} H_{ll'} \left(\sum_{\tau \in \theta} z_{ll'}^\tau\right) + \sum_{l' \in \mathcal{L}} F_{l'} \left(\sum_{l \in \mathcal{L}^+} \sum_{\tau \in \theta} z_{ll'}^\tau\right) + \sum_{(i,j) \in \mathcal{A}} f_{ij} \left(\sum_{l,l' \in \mathcal{L}^+} \sum_{\tau \in \theta} r_{ij}^\tau z_{ll'}^\tau\right) + \sum_{(i,j) \in \mathcal{A}} f_{ij}^e y_{ij}^e + \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k \quad (6.11)$$

subject to

$$z_\tau = \sum_{l,l' \in \mathcal{L}^+} z^\tau_{ll'}, \quad \forall \tau \in \theta, \tag{6.12}$$

$$h_{ll'} = \sum_{\tau \in \theta} z^\tau_{ll'}, \quad \forall l \in \mathcal{L}^+, l' \in \mathcal{L}, l \neq l', \tag{6.13}$$

$$\sum r^\tau_{ij} z_\tau + y^e_{ij} \leq 1, \quad \forall \tau \in \theta, \tag{6.14}$$

$$\sum_{\tau \in \theta_{l'}} z_\tau - \sum_{l \in \mathcal{L}^+} h_{ll'} \leq 0, \quad \forall l' \in \mathcal{L}, \tag{6.15}$$

$$\sum_{l \in \mathcal{L}^+} h_{ll'} = I_{l'}, \quad \forall l' \in \mathcal{L}^+, \tag{6.16}$$

$$\sum_{k \in \mathcal{K}} x^k_{ij} \leq u_{ij} \left(\sum_{\tau \in \theta} r^\tau_{ij} z_\tau + y^e_{ij} \right), \quad \forall (i,j) \in S, \tag{6.17}$$

$$\sum_{j:(i,j) \in \mathcal{A}} x^k_{ij} - \sum_{j:(j,i) \in \mathcal{A}} x^k_{ji} = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k) \\ -w^k & \text{if } i = d(k) \end{cases} \quad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \tag{6.18}$$

$$x^k_{ij} \geq 0, \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{K}, \tag{6.19}$$

$$z^\tau_{ll'},\ z_\tau \in \{0,1\}, \quad \forall \tau \in \theta,\ l, l' \in \mathcal{L}^+, \tag{6.20}$$

$$y^e_{ij} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{S}. \tag{6.21}$$

The objective function (6.11) equals to (6.1) where we replace z_τ by $\sum_{l,l' \in \mathcal{L}^+} z^\tau_{ll'}$ in any possible term. Constraint sets (6.12) and (6.20) ensure the relation between design cycle variables z_τ and binary variables $z^\tau_{ll'}$. Constraint set (6.13) defines the relation between the design cycle variables $z^\tau_{ll'}$ and the number of resources repositioned from terminal l to terminal l'. Constraint sets (6.14), (6.15), (6.16), (6.17), (6.18) correspond respectively to constraints (6.2), (6.3), (6.4), (6.5), (6.6).

6.5 Algorithmic Approach

We propose a slope-scaling based approach for LWSND which is named SSCG. Slope-scaling has been used to address network design problems, e.g., Kim & Pardalos (1999),

Crainic *et al.* (2004) and recently Crainic *et al.* (2013). The algorithm includes a column generation procedure which generates cycles to develop approximation model in slope-scaling procedure. During the loop of the slope-scaling procedure, the approximation is solved and updated iteratively. The algorithm then derives feasible solutions for LWSND from the solutions of the approximation problems.

Algorithm 6.1 details our solution approach. This algorithm calls iteratively a slope-scaling procedure which is executed over a set of cycles generated by a column generation procedure. During the initialization phase of the algorithm, a basic set of cycles $\bar{\theta}$ is generated by using column generation to solve the linear relaxation of LWSND. Then, we use cycles in $\bar{\theta}$ to build AP(LWSND), an approximation of LWSND. Because AP(LWSND) includes many flow variables, if the network is large, the solver may require much memory and time to solve AP(LWSND). To avoid this disadvantage, we use only cycles τ if τ belongs to a solution to the linear relaxation of LWSND during the column generation and the value $z_\tau > \alpha$ (α is a parameter which controls the selection of cycles). We assign initial values to the linearization factors ρ and ϕ for all variables in AP(LWSND). From the solution to AP(LWSND), we obtain a (possibly infeasible) solution to LWSND and then we convert this solution into a feasible solution. If the intensification or diversification conditions are met, the corresponding operation is activated. Finally, we update the linearization factors to continue the procedure until a stopping condition is reached.

The current algorithm and the algorithm described in Crainic *et al.* (2013) share a similar approach but have some differences in each component. The first difference is in the column generation procedure. We use big M idea (Barnhart *et al.*, 1998) instead of generating a set of initial cycles to find an initial feasible solution to the relaxation model of the problem. We only use a subset of $\bar{\theta}$ instead all the cycles in $\bar{\theta}$. The second difference is in the development of the approximation model. We rewrite the formulation to establish a relation between flow variables and reposition variables, and then the approximation model is built from the rewritten formulation. The third difference is in the feasibility scheme in which the current algorithm solves the restricted model to check whether we can obtain a feasible solution and how resources are allocated.

Algorithm 6.1 The solution approach SSCG for LWSND.

1. Solve the linear relaxation of LWSND, with column generation to derive an initial set of cycles $\bar{\theta}$ for the slope-scaling.
2. Reduce the size of $\bar{\theta}$ by keeping only τ with $z_\tau > \alpha$.
3. Set up initial values for linearization factors ρ and ϕ for each cycle $\tau \in \bar{\theta}$.
4. Repeat
5. Create the linear approximation problem AP(LWSND(ρ, ϕ)) of LWSND based on rwLWSND parametrized by linearization factors ρ and ϕ.
6. Solve AP(LWSND(ρ, ϕ)) to obtain a (possibly infeasible) solution α to LWSND.
7. Solve the feasibility problem for α to get a feasible solution to LWSND.
8. If *intensification condition* is met
9. Active the intensification procedure.
10. If *diversification condition* is met
11. Activate the diversification procedure.
12. Update linearization factors ρ.
13. Until *stopping condition* is satisfied.
14. Return the best found solution.

6.5.1 Column Generation

Column generation is an approach that solves a linear optimization problem by iteratively generating and adding new variables to the model until no new improving variables are found (see Barnhart *et al.* (1998); Desaulniers *et al.* (2005) for more details on column generation). In our solution approach, column generation is used to solve the linear relaxation of LWSND and explicitly generate a set of cycles used to construct an approximate model, AP(LWNSD), of LWSND.

The column generation in our problem starts with an empty set $\bar{\theta}$ (because no cycle has been created yet). If all resources allocation operations are allowed, we can execute column generation procedure because the linear relaxation of the LWSND has a feasible solution, e.g., by setting $y^e_{ij} = 1 \forall (i,j) \in \mathcal{S}$. However, if $y^e_{ij} = 0, \ \ \forall (i,j) \in S$ (the service outsourcing option is not available), the column generation cannot start because we have no feasible solutions. In this case, we use big M approach (a popular technique in solving linear systems - see Barnhart *et al.* (1998)), i.e. assigning a high value for f^e_{ij}, e.g. f^e_{ij} =M, $\forall (i,j) \in S$, to start the procedure. The reason is with a high value of M, for any $y^e_{ij} > 0$ in a solution of the linear relaxation, this variable will be 0 (i.e. replaced by some cycles) to reduce the objective function. When the column generation stops, all variables y^e_{ij} will be 0 and we will obtain a solution to the linear relaxation of LWSND.

We associate dual variables $\alpha_{ij} \ \ (\leq 0)$ with each constraint in set (6.5), $\beta_{ij} \ \ (\leq 0)$ with each constraint in set (6.4), and $\gamma_l \ \ (\leq 0)$ with each constraint in set (6.3). Then, the reduced cost associated with variable z_τ is $F_l - \gamma_l + \sum_{(i,j)\in\tau}(f_{ij} + \alpha_{ij}u_{ij} - \beta_{ij})$. Given these dual values, we search for cycles with negative reduced costs to add to the set $\bar{\theta}$ in a weighted acyclic graph in which cost of each arc is defined through the function $f_{ij} + \alpha_{ij}u_{ij} - \beta_{ij}$ of dual values. We implemented a dynamic programming algorithm to find the smallest reduced cost cycle. For the detail of how to define the graph and the algorithm to find negative reduced cost cycles, we refer to Crainic *et al.* (2013) for the similar procedure.

6.5.2 Slope-scaling Framework

In this section, we develop the slope-scaling framework for the LWSND. The development of slope-scaling requires an approximation of LWSND. Crainic *et al.* (2013) presents how we

can define an approximation model for service network design problems with cycle-based formulation. In LWSND, in addition to design-cycle variables z_τ, we also have position variables $h_{ll'}$, and renting variables y^e_{ij}. The approximation of LWSND also needs to approximate these positions variables and parts of the objective function related to them. To achieve this goal, we develop the approximation model based on the formulation rwLWSND in Section 6.4.

Approximation model development The approximation problem AP(LWSND(ρ,ϕ)) of LWSND, which is a pure multi-commodity network flow problem, is defined such to identify a set of cycles/services that can form a (possibly infeasible) solution of LWSND. (ρ,ϕ) are the parameters which are introduced in the following.

We present flow variables and linearization factors which are used to generate the approximation of LWSND. Flow variables determine how commodities flow in the network. Linearization factors are parameters used to reflect the fixed cost part in the objective function of LWSND. The linearization factors parameters are updated based on flow information of the approximate problem. The insourcing cycle flow variable $x^{k\tau}_{ijll'}$ denotes the amount of the commodity k transported on arc (i,j) of cycle τ by an own resource initially located at terminal l and being repositioned to terminal l'. The outsourcing flow variable x^{ek}_{ij} denotes the amount of commodity k routed on arc (i,j) by an external resource (outsourcing to another carrier). We have the relation

$$\sum_{l,l' \in \mathcal{L}^+} \sum_{\tau \in \theta} r^\tau_{ij} x^{k\tau}_{ijll'} + x^{ek}_{ij} = x^k_{ij}, \quad \forall (i,j) \in S, k \in \mathcal{K} \tag{6.22}$$

Let $\rho^{k\tau}_{ijll'}$ denote the linearization factors associated with the insourcing flow variable $x^{k\tau}_{ijll'}$. Let ϕ^{ek}_{ij} denote the linearization factors associated with the outsourcing flow variable x^{ek}_{ij}. The sum $\sum_{ij \in \mathcal{A}} \sum_{k \in \mathcal{K}} x^{k\tau}_{ijll'} \rho^{k\tau}_{ijll'}$ approximates the fixed cost part $H_{ll'} z^\tau_{ll'} + F_{l'} z^\tau_{ll'} + \sum_{(i,j) \in \mathcal{A}} f_{ij} r^\tau_{ij} z^\tau_{ll'}$ associated to each design cycle variable $z^\tau_{ll'}$ appearing in the objective function (6.11). The sum $\sum_{k \in \mathcal{K}} x^{ek}_{ij} \phi^{ek}_{ij}$ approximates the fixed cost part $f^e_{ij} y^e_{ij}$ associated to each service-outsourced design variable y^e_{ij} appearing in the objective function (6.11). The AP(LWSND(ρ,ϕ)) is therefore defined as follows:

$$\text{minimize} \sum_{l,l' \in \mathcal{L}^+} \sum_{\tau \in \theta} \sum_{ij \in \mathcal{A}} \sum_{k \in \mathcal{K}} x^{k\tau}_{ijll'} \rho^{k\tau}_{ijll'} + \sum_{ij \in \mathcal{A}} \sum_{k \in \mathcal{K}} x^{ek}_{ij} \phi^{ek}_{ij} + \sum_{ij \in \mathcal{A}} \sum_{k \in \mathcal{K}} c^k_{ij} x^k_{ij} \tag{6.23}$$

subject to:

$$\sum_{l,l' \in \mathcal{L}^+} \sum_{\tau \in \theta} r_{ij}^{\tau} x_{ijll'}^{k\tau} + x_{ij}^{ek} = x_{ij}^{k}, \forall (i,j) \in \mathcal{S},\ \forall k \in \mathcal{K}, \tag{6.24}$$

$$\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^{k} - \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{k} = \begin{cases} w^k & \text{if } i = o(k) \\ 0 & i \neq o(k),\ d(k) \\ -w^k & \text{if } i = d(k) \end{cases} \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}, \tag{6.25}$$

$$\sum_{k \in \mathcal{K}} x_{ij}^{k} \leq u_{ij}, \quad \forall (i,j) \in \mathcal{S}, \tag{6.26}$$

$$0 \leq x_{ijll'}^{k\tau}, x_{ij}^{ek} \leq u_{ij} \quad \forall (i,j) \in \mathcal{S},\ k \in \mathcal{K},\ \forall l, l' \in \mathcal{L}^+,\ \tau \in \theta. \tag{6.27}$$

Active cycle/service and (infeasible) solution of LWSND Once AP(LWSND(ρ,ϕ)) is solved, we have two vectors $\{\tilde{x}_{ijll'}^{k\tau}\}$ and $\{\tilde{x}_{ij}^{ek}\}$ of the flow distribution. From these variables, we can identify which cycles are operated by resources and which services are outsourced. A cycle is assumed to be active and would be operated by a resource if at least one of its cycle flow variables $\tilde{x}_{ijll'}^{k\tau}$ is positive:

$$\tilde{z}_{ll'}^{\tau} = \begin{cases} 1 & \text{if } \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} \tilde{x}_{ijll'}^{k\tau} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.28}$$

A service is assumed to be active and to be outsourced if at least one of its outsourced flow variables $\tilde{x}_{ij}^{ek}$ is positive:

$$\tilde{y}_{ij}^{e} = \begin{cases} 1 & \text{if } \sum_{k\in\mathcal{K}} \tilde{x}_{ij}^{ek} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.29}$$

The set of active cycles and active services defines a (possibly infeasible) solution of LWSND. The objective value of the LWSND based on the solution to the approximation problem is given by:

$$\text{LWSND}(\tilde{x}, \tilde{y}, \tilde{z}) = \sum_{l,l' \in \mathcal{L}^+} H_{ll'} \left(\sum_{\tau \in \theta} \tilde{z}_{ll'}^{\tau} \right) + \sum_{l,l' \in \mathcal{L}^+} F_{l'} \left(\sum_{\tau \in \theta} \tilde{z}_{ll'}^{\tau} \right)$$

$$+ \sum_{(i,j)\in\mathcal{A}} f_{ij} \left(\sum_{l,l' \in \mathcal{L}^+} \sum_{\tau \in \theta} r_{ij}^{\tau} \tilde{z}_{ll'}^{\tau} \right) + \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{K}} f_{ij}^{e} \tilde{y}_{ij}^{e} + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^{k} \tilde{x}_{ij}^{k} \tag{6.30}$$

Update of the linearization factors Let t be the iteration counter, for any solution $\tilde{x}$ of AP(LWSND(ρ,ϕ)(t)), the next value $\rho(t+1)$ and $\phi(t+1)$ are determined to reflect the fixed cost. For each design cycle variable $z^{\tau}_{ll'}$, we have the relation:

$$\sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}} \tilde{x}^{k\tau}_{ijll'}\rho^{k\tau}_{ijll'}(t+1) = H_{ll'}\tilde{z}^{\tau}_{ll'} + F_{l'}\tilde{z}^{\tau}_{ll'} + \sum_{(i,j)\in\mathcal{A}} r^{\tau}_{ij}f_{ij}\tilde{z}^{\tau}_{ll'}, \tag{6.31}$$

If $\sum_{k\in\mathcal{K}}\sum_{ij\in\mathcal{A}}\tilde{x}^{k\tau}_{ijll'} > 0$, then $\tilde{z}^{\tau}_{ll'} = 1$, otherwise $\tilde{z}^{\tau}_{ll'} = 0$. We can set

$$\rho^{k\tau}_{ijll'}(t+1) = \begin{cases} \dfrac{F_{l'} + \sum\limits_{ij\in\mathcal{A}} r^{\tau}_{ij}f_{ij} + H_{ll'}}{\sum\limits_{k\in\mathcal{K}}\sum\limits_{ij\in\mathcal{A}}\tilde{x}^{k\tau}_{ijll'}} & \text{if } \sum\limits_{k\in\mathcal{K}}\sum\limits_{ij\in\mathcal{A}}\tilde{x}^{k\tau}_{ijll'} > 0 \\ \rho^{k\tau}_{ijll'}(t) & \text{otherwise} \end{cases} \tag{6.32}$$

Similarly, we have:

$$\sum_{k\in\mathcal{K}} \tilde{x}^{ek}_{ij}\phi^{ek}_{ij}(t+1) = f^{e}_{ij}\tilde{y}^{e}_{ij} \tag{6.33}$$

If $\sum_{k\in\mathcal{K}}\tilde{x}^{ek}_{ij} > 0$, $\tilde{y}^{e}_{ij} = 1$, otherwise, $\tilde{y}^{e}_{ij} = 0$. We can set

$$\phi^{ek}_{ij}(t+1) = \begin{cases} \dfrac{f^{e}_{ij}}{\sum_{k\in\mathcal{K}}\tilde{x}^{ek}_{ij}} & \text{if } \sum_{k\in\mathcal{K}}\tilde{x}^{ek}_{ij} > 0 \\ \phi^{ek}_{ij}(t) & \text{otherwise} \end{cases} \tag{6.34}$$

The update of the linearization factors (6.32) and (6.34) satisfies relations (6.31) and (6.33), i.e., it maintains the same interpretation of reflecting the objective value function if the solution to AP(LWSND(ρ,ϕ)(t)) is $\tilde{x}$.

Initialization of the linearization factors To start the slope-scaling procedure, we need to assign initial values for the linearization factors $\rho^{k\tau}_{ijll'}$ and ϕ^{ek}_{ij}. The initial value $\rho^{k\tau}_{ijll'}(0)$ is the ratio $\left(F_{l'} + \sum_{(i,j)\in S} r^{\tau}_{ij}f_{ij} + H_{ll'}\right) \big/ u_{\tau}$ where $u_{\tau} = \sum_{(i,j)\in\mathcal{S}} r^{\tau}_{ij}u_{ij}$. The initial value $\phi^{ek}_{ij}(0)$ is the ratio f^{e}_{ij}/u_{ij}. As demonstrated in Crainic *et al.* (2013), assigning special values for the linearization factors related to the solution of the linear relaxation of LWSND could provide

better feasible solutions. If the cycle τ or the service arc (i, j) belongs to the solution of the linear relaxation formulation of LWSND, we assign smaller values to them, i.e. $\rho^{k\tau}_{ijll'}(0) = 1$ and $\phi^{ek}_{ij}(0) = 1$, to favor these cycles/services in the search for the solution to AP(LWSND).

6.5.3 Creating a Feasible Solution to LWSND

Because the approximation problem AP(LWSND) does not include all the constraints (constraint sets (6.2) - (6.5)), and because rounding procedures are not sufficient to construct a feasible solution to the LWSND from a solution to the AP(LWSND), we execute another procedure to construct a feasible solution to the LWSND. Our procedure creates a subgraph, $\bar{\mathcal{G}}$ of $\mathcal{G}$ that contains the nodes $\mathcal{N}$ and a subset, $\bar{\mathcal{A}}$, of the arcs, $\mathcal{A}$, that can be decomposed into cycles. Then, the procedure attempts to extract cycles from this subgraph that can be used to construct a feasible solution to the LWSND.

We create $\bar{\mathcal{G}}$ by first adding to $\bar{\mathcal{A}}$ all service arcs (i, j) in $\mathcal{A}$ such that either these arcs carry some flow, or they belong to a cycle used in the AP(LWSND) solution. We also add all holding arcs to $\bar{\mathcal{A}}$. We solve an optimization problem to add service arcs in $\mathcal{A} \setminus \bar{\mathcal{A}}$ to $\bar{\mathcal{A}}$ to ensure that $\bar{\mathcal{G}}$ can be decomposed into cycles in such a way that each service arc appears in at most one cycle but holding arcs may appear in multiple cycles. The objective of this optimization problem is to minimize the total cost of the arcs added with respect to the cost coefficient f_{ij}. We note that we formulate and solve this optimization problem as a minimum cost, maximum flow problem Ahuja *et al.* (1994). See Vu *et al.* (2013) for details of how a similar procedure was done for a network design problem with Eulerian-type constraints.

After creating $\bar{\mathcal{G}}$, the cycle extraction algorithm in Crainic *et al.* (2013) is used to extract a set of cycles from this network that are guaranteed to satisfy the 0-1 service constraints (6.4). Finally, we define and solve restricted LWSND to find a feasible solution of the LWSND in t_{MIP} seconds based on extracted cycles and set of possible services.

6.5.4 Intensification and Diversification

For intensification, we introduce an exact optimization step into the search, wherein LWSND($\tilde{\theta}$, $\tilde{y}$) is solved (for what is presumably a set of cycles, $\tilde{\theta}$, of very limited cardinality) with a com-

mercial mixed integer programming solver. To create the set of cycles $\tilde{\theta}$, our intensification procedure first chooses the cycles/services that appear in the last q solutions to AP(LWSND). Then, a subgraph of $\mathcal{G}$ is created based on the arcs that appear in those solutions. We first add to the subgraph to make it Eulerian with the procedure presented in Vu *et al.* (2013). $\tilde{y}$ is defined as the set of service arcs in this graph. A depth-first search-type approach is performed on this subgraph to extract cycles which are TMAX periods length, and we add these cycles to $\tilde{\theta}$ which is initially empty. The restricted LWSND($\tilde{\theta}$,$\tilde{y}$) is then solved with the value of the best-known solution as an upper bound on the objective function value and time limit t_{MIP} (in seconds). The intensification procedure is executed when an improved solution has not been found after a predefined number of iterations.

For diversification, because it is the cycles/services in the solution to AP(LWSND) at a given iteration that dictates the structure of the resulting solution to LWSND, we periodically modify the objective function of AP(LWSND) to avoid frequently used cycles/services. To do so, we collect the number of times, fre_τ, in which each cycle τ appears in a solution to AP(LWSND). We also collect the number of times, fre_{ij}, in which each outsourced service arc (i,j) appears in a solution to AP(LWSND). Then, if the diversification condition is met, for each cycle τ and outsource service arc (i,j) in the current solution of the approximated problem AP(LWSND), we set the linearization factor $\rho^{k\tau}_{ijll'}(t)$ to $\rho^{k\tau}_{ijll'}(t)(1+\epsilon * fre_\tau)$ and $\phi^{ek}_{ij}(t)$ to $\phi^{ek}_{ij}(t)(1+\epsilon * fre_{ij})$ where ϵ is an algorithm parameter. We continue to update $\rho(t)$ and $\phi(t)$in this manner for a fixed number of iterations that is dictated by the algorithm parameter I^{diver}_{max}. Lastly, it is possible for two consecutive solutions of AP(LWSND) to be the same, a case in which the slope-scaling procedure will terminate. To continue the execution of slope-scaling when this occurs, we add a penalty value P to the linearization factors as seen in (6.35) and (6.36). In our experiments, P is set to the objective value of the current solution to AP(LWSND).

$$\rho^{k\tau}_{ijll'}(t+1) = \begin{cases} \dfrac{P + F_{l'} + \sum_{ij\in\mathcal{A}} r^{\tau}_{ij} f_{ij} + H_{ll'}}{\sum_{k\in\mathcal{K}} \sum_{ij\in\mathcal{A}} \tilde{x}^{k\tau}_{ijll'}} & \text{if} \sum_{k\in\mathcal{K}} \sum_{ij\in\mathcal{A}} \tilde{x}^{k\tau}_{ijll'} > 0 \\ \rho^{k\tau}_{ijll'}(t) & \text{otherwise} \end{cases} \tag{6.35}$$

and

$$\phi_{ij}^{ek}(t+1) = \begin{cases} \dfrac{P + f_{ij}^{e}}{\sum_{k \in \mathcal{K}} \tilde{x}_{ij}^{ek}} & \text{if } \sum_{k \in \mathcal{K}} \tilde{x}_{ij}^{ek} > 0 \\ \phi_{ij}^{ek}(t) & \text{otherwise} \end{cases} \tag{6.36}$$

6.6 Computational Experiments

The purpose of the computational experiments is to explore how the proposed method addresses the issues identified in the problem description and to understand the impact these issues might have on the resulting plan and to verify the performance of the proposed algorithm. Experiments are conducted on a cluster of computers that have 2 Intel Xeon 2.6 GHz processors and 48 GB of RAM running Scientific Linux 6.1. We use CPLEX 12.4 to solve linear and mixed integer programs.

Experiments are divided into two parts. In the first part, we evaluate the performance of the algorithm by comparing the solutions found by the algorithm and the solutions found by CPLEX for small instances, and the solutions found by an MIP heuristic for medium- and large-size instances. Next, we examine the performance of the algorithm on instances used in Crainic *et al.* (2013) to show how we can improve solutions when we introduce reposition operations. In the second part, we examine different resource allocation approaches and compare them with LWSND. We also demonstrate how we can obtain a better schedule plan by comparing two plans with and without repositions. Finally, we examine the number of empty moves (moves without flow), the number of outsourced services regarding to different values of resource fixed cost and indicate that we can reduce empty moves by taking into account service outsourcing.

6.6.1 Problem Instances and Parameters

Our experiments are based on the 35 medium and large-size rail-based instances in Crainic *et al.* (2013); Andersen *et al.* (2011) and 14 small-size instances derived from these 35 instances. For the completeness of the document, we represent the information of 35 of medium and large-size instances (grouped into 7 classes, each class has 5 instances) in Table 6.1 and the information of small-size instances in Table 6.2. The number of periods (TMAX) of each small-size instance is changed from its origin value (30, 40 or 50) to new values 20 and 25

while we keep identical other values such as the number of terminals, services and commodities. Considering that the number of periods is limited to 20 or 25, we were able to generate all the cycles for each problem instance and then solve each instance with CPLEX (see Table 6.17 for the number of cycles of each 20- and 25-period instance).

Instances ID	Terminals	Services	Periods	Service+Holding arcs	Commodities
6	5	15	40	600+200	200
7	5	15	50	750+250	400
8	7	30	30	900+210	200
9	7	30	30	900+210	400
10	7	30	50	1500+350	300
11	10	40	30	1200+300	200
12	10	50	30	1500+350	100

Table 6.1: Information of each class of medium and large-size rail-based instances.

Instances	Terminals	Services	Commodities	Service + Holding Arcs	
				20 periods	25 periods
1-4	5	15	200	300+100	375+125
5-6	7	30	400	600+140	750+175
7-8	7	30	200	600+140	750+175
9-10	7	30	300	600+140	750+175
11-12	10	40	200	800+200	900+250
13-14	10	50	100	1000+200	1250+250

Table 6.2: Information of the small-size instances.

The costs are as follows. The reposition cost of own resources is generated randomly using the function $(\text{rand}()\%5+1)*100$ in which rand()%5 generates a random number in the interval [0..4]. The rented resource reposition fixed cost is 1000 to indicate that we need to pay more for rented resources than own resources. The service fixed cost f_{ij} is 50 times the length of the service (as in Crainic *et al.* (2013)) while the renting service fixed cost f^e_{ij} is 4 times of service fixed cost f_{ij}. The resource utilization fixed cost F_l will vary in order to generate different scenarios that favor or balance the use of own resources and rented resources/services.

Instances ID	Terminals	Services	Periods	Service+Holding arcs	Commodities
1	5	10	15	150+75	20
2	5	15	20	300+100	25
3	5	15	25	375+125	25
4	5	15	15	225+75	100
5	5	15	15	225+75	200
6	5	15	40	600+200	200
7	5	15	40	600+200	200

Table 6.3: Instances for the calibration.

We calibrated the four parameters shown in Table 6.4. The parameter q is fixed to 1 as in Crainic *et al.* (2013). Each parameter was tested with two values, as indicated in the "Values tested" column. We tested the 16 different combinations of values for the 4 parameters above, for the 7 instances in Table 2. Column "Value selected" displays the selected values, which were then used for the complete set of test instances.

Parameter	Description	Values tested	Value selected
I_{max}^{diver}	Number of diversification iterations	3, 5	3
ϵ	The effect of frequency on linearization factor	0.5, 1	1
t_{MIP}	Time limit t_{MIP} for solving a LWSND by the solver	600, 900	600
α	Parameter for cycle's selection	0.01, 0.1	0.1

Table 6.4: Parameter values.

To make the selection, we computed for each combination the average gap and the average execution time. Table 6.5 shows the combinations corresponding to the top ten best average gaps with their corresponding average execution times. The difference between the best and the worst among the ten top average gaps is about 0.7%, indicating that these combinations could provide about the same average solution quality. We have thus assigned a score to each combination of values, a score of 10 being assigned to the best combination, 9 to the second best, etc. Table 6.5 scores each parameter value by summing the score of each combination where this value appears.

Results in Table 6.5 show that the values in the combination (3,1,900,0.01) (same order of parameters as in Table 6.4) have the best scores. This combination provides solutions with 4.6% average gap. However, we take the combination (3,1,600,0.1), which provides solutions

with 4.89% average gap. The two combinations share two first parameters but the combination (600,0.1) allows the reduction of the computation time for large instances as well as memory required by the AP(LWSND) (see Table 6.6 for an example of number of cycles selected with two different values of α and number of cycles in the solutions).

Combination	Gap
3,1,900,0.01	4.60%
3,1,600,0.01	4.70%
3,1,600,0.1	4.89%
3,0.5,900,0.1	4.92%
5,0.5,900,0.01	4.95%
5,1,900,0.01	5.02%
3,0.5,600,0.1	5.03%
3,0.5,900,0.01	5.04%
5,1,600,0.01	5.18%
5,0.5,600,0.01	5.23%

Parameter	Value	Score
I_{diver}^{max}	3	41
	5	14
ϵ	0.5	21
	1	34
t_{MIP}	600	23
	900	32
α	0.01	36
	0.1	19

Table 6.5: Ten top average results sorted by gap and the score of each parameter

Instance ID	$\lvert\bar{\theta}\rvert$	$\alpha = 0.01$	$\alpha = 0.1$	#Cycles in the solutions
6	1800	301	237	12
7	2000	382	55	4

Table 6.6: Number of selected cycles with two values of α

6.6.2 Performance Comparisons

This section first compares the performance of our proposed solution SSCG with CPLEX. Next we run SSCG with instances used in Crainic *et al.* (2013) to show how we can get improved plans by introducing reposition operations.

6.6.2.1 Comparisons of CPLEX and SSCG

Table 6.7 reports the average gap of solutions found by SSCG and CPLEX for 20 and 25-period instances. It is assumed that the resource utilization fixed cost F_l is the same at all terminals but varies among these three values 500, 1000 or 2000, while other costs such as

reposition cost and renting cost are kept constant. The costs are chosen to favor/balance the utilization of owned resources and the utilization of external resources and services. A bigger value of F_l, e.g. 3000, leads to a solution in which all the services are outsourced (because in this case the total operation cost of a cycle τ, $F_l + \sum_{(i,j)\in\tau} f_{ij}$, is greater than $\sum_{(i,j)\in\tau} f^e_{ij}$, the cost of outsourcing this cycle) while a small value of F_l, e.g. 500, favor solutions that use only local resources (if they are enough). We consider two initial resources distribution patterns: enough resources and not enough resources. In total, we have 6 scenarios; CPLEX solves 20-period instances within 1 hour of execution; and 25-period instances within 1 hour and 10 hours of execution. SSCG solves 20 and 25-period instances using 1 hour of execution.

	Enough resources			Not enough resources		
F_l	500	1000	2000	500	1000	2000
20 periods - CPLEX 1H gap	2.16	2.54	3.23	4.38	4.79	3.44
20 periods - SSCG 1H gap	3.94	4.12	5.08	5.13	4.61	5.59
25 periods - CPLEX 1H gap	4.95(2)	6.58(1)	7.69(2)	7.90(3)	12.29(2)	12.15(3)
25 periods - CPLEX 10H gap	3.00(1)	3.69	4.57	4.35	4.58	5.35
25 periods - SSCG 1H gap	4.15	5.04	6.25	5.15	5.25	6.30

Table 6.7: CPLEX and SSCG regarding to small instances.

The first two rows of the table show that CPLEX and SSCG find good feasible solutions for 20-period instances in all scenarios. For 25-periods instances, CPLEX within 1 hour of execution can not find feasible solutions for some instances (numbers in parentheses in the third row give the number of instances for which CPLEX has failed to provide feasible solutions). Results obtained by CPLEX in 1 hour for 20 and 25-period instances indicate that CPLEX may not perform well for large networks because of the number of design variables. In average, changing the period from 20 to 25 increases the number of design cycle variables 10 times (which are in the interval [10^5,10^7], see Table 6.17). CPLEX needs to run for 10 hours in order to obtain good feasible solutions for 25-period instances. However, the last row shows that SSCG provides solutions of good quality with much less execution time. These comparisons show that SSCG is competitive with CPLEX for small instances.

Now we compare the performances of CPLEX and our heuristic for medium- or large-size instances in which not all cycles can be generated and stored. We also have six scenarios in this experiment with three values of F_l, $1000, 3000, 5000$ and two resources distribution scenarios (enough and not enough resources). We let the slope-scaling part of the SSCG run within 5

hours or 100 iterations, whichever comes first. On the other hand, CPLEX solves within 5 hours the restricted LWSND problem using a set of cycles provided by column generation.

	Enough resources			Not enough resources		
Resource utilization fixed cost	1000	3000	5000	1000	3000	5000
SSCG optimal gap	5.25%	6.65%	5.40%	6.31%	7.20%	5.67%
CPLEX gap	15.70%(1)	15.25%(2)	10.31%(1)	16.46%(2)	14.64%(2)	8.73%(1)

Table 6.8: CPLEX and SSCG for medium- and large-size instances.

Results in Table 6.8 show that SSCG found feasible solutions for all instances with acceptable average gaps while CPLEX fails to find feasible solutions for some instances and has average gaps which are quite high. Table 6.9 provides more details on the results in Table 6.8. It shows that SSCG outperforms CPLEX for instances having many commodities (e.g. 300, 400 commodities). Moreover, all the instances for which CPLEX fails to find feasible solutions belong to this set. For instances with fewer commodities (100, 200 commodities), SSCG is competitive with CPLEX and is only beaten by CPLEX for scenarios where there are not enough resources and high resource utilization fixed cost. CPLEX performs better with these instances but SSCG is still competitive.

	Enough resources			Not enough resources		
Resource utilization fixed cost	1000	3000	5000	1000	3000	5000
SSCG optimal gap - 300/400 commodities	4.51%	5.38%	4.22%	4.38%	5.49%	4.14%
CPLEX gap - 300/400 commodities	27.15%	24.96%	15.77%	29.65%	25.97%	13.20%
SSCG optimal gap - 100/200 commodities	5.73%	7.60%	6.25%	7.4%	8.2%	11.4%
CPLEX gap - 100/200 commodities	7.68%	9.12%	5.13%	7.89%	7.27%	5.08%

Table 6.9: CPLEX and SSCG for medium- and large-size instances.

6.6.2.2 Performance of SSCG on SNDRC's Problem Instances

This section reports the results of running SSCG on problem instances in Crainic *et al.* (2013) for the SNDRC model. The SNDRC is derived from LWSND by fixing all the resources allocation variables to 0. In this part, we limit LWSND with reposition operations. We use 3 sets of resources distribution in Crainic *et al.* (2013) for the goal of comparison. Two set of resources distribution scenarios are named "Random distribution" and "Proportional distribution" with

the service fixed cost $f_{ij} = 0$. In the last set, the number of resources is in the interval [5,10]; and $f_{ij} \neq 0$.

First, we verify whether the algorithm for LWSND can solve the SNDRC well? To do so, we limit the number of cycles extracted at each terminal so that each terminal l does not use more than I_l resources. We run the modified algorithm on SNDRC instances. The results show that we obtain worse and infeasible solutions than those in Crainic *et al.* (2013), especially instances with Random distribution. We note that, in SNDRC, there is a procedure that handles the resource bound violation, and in LWSND, this issue is solved by the reposition of resources. The modified algorithm, which fix reposition variables to 0, cannot handle resource issues and this leads to downgraded and infeasible solutions.

	Proportional Distribution	Random Distribution
Ave. Improvement Gap	-1.33%	-1.23%
#Improvement	26	27
Average optimal LWSND gap	7.70%	7.69%

Table 6.10: Comparison of the solutions to LWSND and the solutions to SNDRC with respect to two different distributions of resources.

Otherwise, if we do not fix allocation variables to 0, we will obtain improved solutions comparing to those in Crainic *et al.* (2013). Table 6.10 reports results for the first two sets of resource distribution. SSCG obtained feasible solutions for all instances with -1.33% and -1.23% average improved gaps respectively with the proportional resources distribution and random resources distribution scenarios. We also obtain 26 and 27 better solutions respectively for the proportional and random distributions. The average lower bound gaps of LWSND are about 7.70% for these two distributions. This indicates the algorithm is not so sensitive to the resource distribution because of the support of the reposition operations.

We present the results related to the last set of resources distribution. Compared with the solutions provided in Crainic *et al.* (2013), Table 6.11 (see also Table 6.21 in the Annex) indicates that the solution approach provides solutions with improved quality, in which we have 32/35 better solutions found with 1.50% improvement gap.

	LWSND/SNDRC	#Improvement
$f_{ij} \neq 0$	-1.50%	32

Table 6.11: Comparison of the solutions to LWSND and the solutions to SNDRC with respect to the case $f_{ij} \neq 0$.

6.6.3 Analysis of the Formulation

In this section, we examine some basic questions using CPLEX. First, we examine the question whether we should consider resources management jointly with service network design problems. We want to know it is better if we consider the reposition/acquisiton of resources together with the design of service network or we will consider these two problems separately. Second, we examine the question of how we can save with "reposition" operations. Finally, we examine the number of empty moves by considering two cases, balanced and unbalanced demand, and point out that services outsourcing is an option to avoid empty moves.

6.6.3.1 How We Should Position Resources

We examine whether we solve the resource position together with the transportation problem (the LWSND) or we should consider these two problems separately. To answer this question, we consider two other reposition approaches. In the first approach, we solve the transportation problem (named SND) and then we solve the positioning of resources (named POSITION). In the second one, we position resources first and then we solve the transportation problem. In the first approach, regarding to SND, we assume that there are many resources at each terminal, and then from the solution to SND, we will determine how we allocate resources to each terminal. In the second approach, we assume that each terminal will need a number of resources which is proportional to the total volumes of commodities departed from this terminal. We compare these two approaches with the LWSND.

F_l	LWSND / SND + REPOSITION	
	Improvement Gap	#Improvement
1000	-0.58%	12/14
2000	-0.31%	10/14

Table 6.12: Compare the LWSND and the first alternative approach.

F_l	LWSND / REPOSITION+SND	
	Improvement Gap	#Improvement
500	-0.26%	12/14
1000	-0.35%	10/14

Table 6.13: Compare the LWSND and the second approach.

We report the experiments related to 14 small instances solved by CPLEX within 1 hour with two different values of resource fixed cost F_l. Tables 6.12 and 6.13 indicate that the integrated model provides better solutions than the solutions found by solving the transportation of commodities and the problem of the resources allocation separately. The column "Improvement gap" indicates how much better the solutions to LWSND give, and the column "#Improvement" presents the number of improved solutions that LWSND found. These number indicate quantitatively the advantages of the integration of reposition into service design models.

In addition to quantitative value as we saw before, we examine how the reposition operation can provide a better schedule. We consider an example in which one seeks a plan which minimizes the number of resources used to transport all the demands and minimizes the total handling/waiting time at intermediate terminals (representing through holding arcs which do not belong to destination terminals). We obtain this goal by assigning appropriate values to the objective function of the LWSND. We assign high fixed costs to resources and high routing costs to commodities at holding arcs. Other costs are set to zero. The reposition cost is assumed to be much smaller than other costs and we assign 0 to this cost.

First we fix the reposition variables to zero to obtain initial solutions without reposition. Second, we allow the repositions and check how much we can “save” when the reposition is enabled. Table 6.14 presents the resources and the total number of waiting/handling periods extracted from solutions with the instances where TMAX = 20 and after 10 hours of execution. The long execution time is to guarantee that the best found solutions are optimal or close to optimal and that these solutions are reasonable to understand how much we can save with the integration of the reposition operation.

Instances	Reposition		No reposition	
	Resources	#Handling/waiting periods	Resources	#Handling/waiting periods
1	30	3,530	30	3,620
2	27	2,770	27	3,150
3	14	2,720	14	2,930
4	19	5,140	19	5,210
5	41	6,180	41	6,520
6	43	4,790	43	5,470
7	26	5,920	26	6,130
8	39	7,870	39	10,720
9	28	7,470	28	7,640
10	35	4,390	35	5,090
11	28	2,850	29	2,470
12	34	1,950	34	1,990
13	26	6,400	26	6,780
14	24	7,310	25	7,450

Table 6.14: Results regarding to two scenarios: with and without reposition.

Table 6.14 indicates that solutions with repositions are better than those without repositions, i.e. they need fewer resources and reduce total waiting/handling time (in period) of shipments. For example, instance 2 shows that we can save $3150 - 2770 = 380$ handling/waiting periods through the integration of the reposition; in average, the reposition allows saving 420 periods of handling/waiting activity at terminals.

From the results of the experiment, the contributions of the reposition activities could be:

- Resources & services: the reposition operation allows a better utilization of resources. If the reposition is integrated in the formulation, all resources can be repositioned appropriately from terminals to terminals before the execution of planning schedule. This will improve the total performance of the system.

- Commodities: the integration of reposition operation also allows better transportation plans for demands. The reposition allows us to obtain plans with less total waiting/handling times or total routing times or their combinations.

Resource operation fixed cost	Resources	Rented Services	Empty moves
High demand patterns			
500	29.0	2.1	4.2%
1000	25.9	5.8	2.3%
2000	19.8	18.0	0.0%
Low demand patterns			
500	8.9	1.0	4.6%
1000	7.9	2.2	0.9%
2000	5.5	6.6	0.0%

Table 6.15: Average number of resources, services and empty moves regarding to different values of resource operation fixed cost.

6.6.3.2 Examine of Empty Moves and Service Outsourcing

In this section, we want to examine in which cases we could consider service outsourcing. To do that, we take into account the number of empty moves, resources, and service outsourcing regarding different values of resource fixed costs and demand patterns. Table 6.15 presents the aggregated number regarding two demand patterns scenarios (named high and low demand pattern). The number of commodities of the low demand pattern is equal to 1/4 the number of commodities of the high demand pattern. The value of F_l will balance the choice between the use of own/rented resources and the outsources of services. A small value of F_l will favor the use of own resources rather than the outsource of services. When F_l increases, the greater number of services will be outsourced. Table 6.15 indicates that the number of empty moves is small compared to the number of used resources even when the number of demands is low (from 50 to 100 demands in total for each instance) but balanced (each terminal both sends and receives shipments).

If the demand is unbalanced, e.g., by assuming that there is a terminal which only sends or receives shipments from other terminals, the number of empty moves will be high. Table 6.16 indicates that the number of empty moves of an unbalanced demand is much higher than that of a balanced demand. These empty moves presents the unprofitable utilization of resources. To avoid these empty moves, the carrier may rely on service outsourcing choices and save resources for other tasks.

	Balance demand	Imbalance demand	
		Receive	Send
Total number of empty moves	22	137	116
Average number of empty move per instance	5.5%	36.0%	31.4%

Table 6.16: The number of empty moves regarding balance and unbalance demands.

6.7 Conclusion

This chapter presents a class of service network design problem that considers resources allocation issues. We propose a service network design model named LWSND that integrates resources allocation operations which include the reposition/acquisition of resources and the outsource. We introduce a solution approach that includes column generation, slope-scaling and exact method to solve LWSND.

The experiment for the problem is divided into two parts. The first part verifies the performance of the proposed algorithm. We compare outputs with the lower bounds and the solutions found by CPLEX. The experiment shows that the algorithm is able to find good feasible solutions. The algorithm is competitive and even better than CPLEX regarding different test instances. We also show that we can obtain improved solutions when we introduce resources repositions into the model.

In the second part of the experiment, we examine some basic questions using CPLEX. We examine the question whether we should consider resources management jointly with service network design problems. We also examine the question of how we can save with "reposition" operations. Finally, we examine the number of empty moves by considering two cases, balanced and unbalanced demand, and point out that services outsourcing is an option to avoid empty moves.

We present some future research directions related to the problem. First, we can extend the model to allow multiple resources on a service which happens in truck-based delivery. One of them is an improvement of cycle-extraction procedure which takes into account flow information, resources information at each terminal, reposition cost, etc. This will provide a better utilization of resources while taking into account side information. Taking into account company's rules such as short cycles, e.g, cycles with length of 2 days on a 5 days time-space network should be considered, a situation in truck-based transportation.

6.8 Annex

In the Annex, we include the detail tables of the results presented in the previous sections.

Instance	20-period	25-period
1	109,600	1,431,250
2	126,040	1,655,800
3	119,600	1,625,375
4	531,420	10,137,475
5	21,180	191,150
6	35,500	336,525
7	19,920	170,550
8	29,980	293,950
9	24,220	255,800
10	19,880	167,250
11	23,020	227,600
12	14,160	165,450
13	67,960	851,075
14	104,340	1,142,925

Table 6.17: The number of cycles of each 20-period and 25-period instance.

Instance	20-period instances 1h			25-period instances 1h			25-period instances 10h		
	Obj	Bound	Gap	Obj	Bound	Gap	Obj	Bound	Gap
1	195,646	193,429	1.13%	188,476	186,465	1.07%	188,476	186,473	1.06%
2	170,300	169,484	0.48%	168,148	166,752	0.83%	168,148	166,752	0.83%
3	89,834	88,233	1.78%	n/a	n/a	n/a	85,834	84,696	1.33%
4	136,863	135,187	1.22%	n/a	n/a	n/a	133,297	131,557	1.31%
5	261,443	257,941	1.34%	250,480	245,977	1.80%	250,284	245,977	1.72%
6	288,056	285,161	1.01%	263,061	259,434	1.38%	263,041	259,434	1.37%
7	154,781	147,344	4.80%	144,672	134,413	7.09%	143,282	134,417	6.19%
8	254,240	251,363	1.13%	240,744	232,726	3.33%	238,815	232,726	2.55%
9	195,224	191,375	1.97%	189,019	183,364	2.99%	188,584	183,364	2.77%
10	225,533	221,920	1.60%	208,006	202,784	2.51%	207,049	202,784	2.06%
11	115,966	108,013	6.86%	107,634	84,987	21.04%	104,338	91,620	12.19%
12	126,564	123,131	2.71%	109,445	81,608	25.43%	101,255	89,338	11.77%
13	156,842	152,167	2.98%	150,471	140,540	6.60%	148,663	142,951	3.84%
14	143,699	138,627	3.53%	n/a	n/a	n/a	138,045	132,074	4.32%

Table 6.18: Objective value and optimal gap of solutions found for 20 and 25-period instances by CPLEX.

Instance	20-period instances 1h			25-period instances 1h			25-period instances 10h		
	Obj	Bound	Gap	Obj	Bound	Gap	Obj	Bound	Gap
1	198,172	195,011	1.6%	186,670	184,736	1.0%	186,670	184,736	1.0%
2	170,900	169,944	0.6%	170,642	168,754	1.1%	169,900	168,754	0.7%
3	91,132	89,241	2.1%	n/a	n/a	n/a	85,332	83,667	2.0%
4	139,394	136,825	1.8%	n/a	n/a	n/a	133,120	131,359	1.3%
5	259,666	255,306	1.7%	247,852	242,540	2.1%	246,539	242,540	1.6%
6	275,180	271,437	1.4%	261,601	257,744	1.5%	260,781	257,839	1.1%
7	162,522	145,512	10.5%	168,117	134,408	20.1%	152,535	134,611	11.8%
8	255,854	245,140	4.2%	236,976	231,106	2.5%	236,717	231,472	2.2%
9	207,203	196,114	5.4%	192,287	183,890	4.4%	190,363	183,890	3.4%
10	229,613	219,180	4.5%	206,455	202,916	1.7%	206,121	202,916	1.6%
11	120,321	110,315	8.3%	221,220	99,549	55.0%	106,980	99,549	6.9%
12	122,791	113,052	7.9%	106,894	81,322	23.9%	110,707	89,197	19.4%
13	158,950	149,950	5.7%	161,452	138,455	14.2%	149,559	141,495	5.4%
14	149,081	138,709	7.0%	137,309	129,066	6.0%	135,015	130,853	3.1%

Table 6.19: Objective value and optimal gap of solutions found for 20 and 25-period instances by CPLEX.

Instances	LWSND		SNDRC		LWSND/SNDRC	
	Proportion	Random	Proportion	Random	Proportion	Random
6.1	175,585	175,319	174,051	173,911	0.87%	0.80%
6.2	51,389	51,504	51,385	51,385	0.01%	0.23%
6.3	187,072	183,038	182,612	187,372	2.38%	-2.37%
6.4	214,046	213,926	216,534	216,922	-1.16%	-1.40%
6.5	201,211	205,612	188,952	n/a	6.09%	n/a
7.1	349,679	349,951	369,159	374,199	-5.57%	-6.93%
7.2	454,903	455,293	n/a	470,245	n/a	-3.28%
7.3	93,732	93,298	93,334	93,374	0.42%	-0.08%
7.4	144,879	144,324	139,409	140,146	3.78%	2.89%
7.5	369,959	369,736	395,041	387,780	-6.78%	-4.88%
8.1	370,503	370,372	379,144	375,434	-2.33%	-1.37%
8.2	82,092	81,401	82,714	82,672	-0.76%	-1.56%
8.3	362,495	365,911	366,825	365,689	-1.19%	0.06%
8.4	312,423	317,252	321,594	317,304	-2.94%	-0.02%
8.5	329,578	321,988	330,922	333,002	-0.41%	-3.42%
9.1	171,649	170,891	176,850	171,717	-3.03%	-0.48%
9.2	301,424	300,962	303,059	309,467	-0.54%	-2.83%
9.3	691,423	687,578	n/a	n/a	n/a	n/a
9.4	716,515	719,385	n/a	n/a	n/a	n/a
9.5	325,413	323,082	330,236	324,321	-1.48%	-0.38%
10.1	396,986	399,421	406,820	n/a	-2.48%	n/a
10.2	202,464	201,110	198,663	200,155	1.88%	0.47%
10.3	221,703	219,764	220,325	220,315	0.62%	-0.25%
10.4	440,755	440,452	451,124	467,769	-2.35%	-6.20%
10.5	451,209	455,305	445,302	n/a	1.31%	n/a
11.1	235,421	242,199	246,217	238,203	-4.59%	1.65%
11.2	519,447	515,235	520,668	n/a	-0.24%	n/a
11.3	110,010	109,762	113,379	109,465	-3.06%	0.27%
11.4	119,827	119,488	121,941	119,836	-1.76%	-0.29%
11.5	111,442	112,226	115,434	112,831	-3.58%	-0.54%
12.1	88,400	89,436	90,680	87,257	-2.58%	2.44%
12.2	105,773	104,328	107,718	108,047	-1.84%	-3.56%
12.3	54,314	54,553	58,207	55,221	-7.17%	-1.22%
12.4	185,580	184,532	186,337	185,308	-0.41%	-0.42%
12.5	171,676	173,263	177,942	176,835	-3.65%	-2.06%
Average					-1.33%	-1.23%

Table 6.20: Comparision of the objective function with regard to reposition and without reposition operation

Instances	LWSND	SNDRC	Bound	LWSND/Bound	LWSND/SNDRC
6.1	198,025	204,097	184,536	6.81%	-3.07%
6.2	58,254	58,763	53,089	8.87%	-0.87%
6.3	209,320	215,578	198,220	5.30%	-2.99%
6.4	245,646	255,134	229,972	6.38%	-3.86%
6.5	228,109	214,350	206,387	9.52%	6.03%
7.1	395,463	411,259	380,384	3.81%	-3.99%
7.2	516,229	539,258	480,593	6.90%	-4.46%
7.3	106,696	108,723	99,973	6.30%	-1.90%
7.4	165,133	172,897	157,817	4.43%	-4.70%
7.5	427,167	427,349	404,529	5.30%	-0.04%
8.1	426,243	441,804	401,946	5.70%	-3.65%
8.2	92,385	92,929	80,737	12.61%	-0.59%
8.3	404,578	419,310	384,497	4.96%	-3.64%
8.4	359,702	370,847	341,288	5.12%	-3.10%
8.5	373,160	376,121	350,188	6.16%	-0.79%
9.1	195,342	191,967	179,669	8.02%	1.73%
9.2	352,658	354,780	317,498	9.97%	-0.60%
9.3	817,845	819,869	743,430	9.30%	0.00%
9.4	830,975	834,529	758,914	8.67%	0.00%
9.5	369,098	371,984	344,144	6.76%	-0.78%
10.1	453,961	469,833	433,409	4.53%	-3.50%
10.2	231,471	232,485	211,523	8.62%	-0.44%
10.3	254,395	257,824	234,249	7.92%	-1.35%
10.4	518,868	545,125	478,755	7.73%	-5.06%
10.5	514,469	572,349	489,662	4.82%	-11.25%
11.1	266,790	272,329	252,991	5.17%	-2.08%
11.2	591,801	590,043	551,047	6.89%	0.30%
11.3	127,248	123,165	111,135	12.66%	3.21%
11.4	135,074	135,811	121,612	9.97%	-0.55%
11.5	127,117	131,578	114,376	10.02%	-3.51%
12.1	102,481	101,211	84,877	17.18%	1.24%
12.2	123,255	123,929	108,080	12.31%	-0.55%
12.3	63,256	62,663	50,130	20.75%	0.94%
12.4	216,242	213,667	194,302	10.15%	1.19%
12.5	203,477	201,500	181,247	10.93%	0.97%
Average				8.31%	-1.39%
Improvement					27/35

Table 6.21: Results with respect to the integration of reposition where service cost $f_{ij} \neq 0$.

Instance	$F_l = 1000$			$F_l = 3000$			$F_l = 5000$		
	CPLEX	SSCG	LB	CPLEX	SSCG	LB	CPLEX	SSCG	LB
6.1	146,603	147,084	143,871	165,903	169,417	163,494	184,123	187,979	181,382
6.2	41,504	41,540	40,049	47,717	47,311	45,288	53,522	53,016	49,900
6.3	155,648	156,570	154,084	176,967	179,880	174,714	196,967	201,330	194,332
6.4	178,396	180,670	176,606	202,272	204,376	199,178	224,196	227,046	220,960
6.5	155,900	158,578	154,582	180,048	184,848	178,467	209,899	204,100	207,492
7.1	n/a	313,845	309,733	360,203	351,947	343,501	379,081	389,517	373,915
7.2	408,775	399,664	390,651	n/a	451,749	433,538	n/a	493,111	474,596
7.3	134,160	82,600	79,324	96,522	92,609	87,720	106,208	101,695	95,972
7.4	130,851	127,165	124,198	140,586	145,595	137,877	152,750	155,730	149,965
7.5	440,291	332,883	327,534	379,788	374,943	363,968	454,581	412,551	398,594
8.1	296,038	293,115	286,776	342,817	347,087	336,053	371,131	371,660	365,546
8.2	63,248	63,485	57,717	74,789	76,520	67,375	82,921	82,976	74,179
8.3	279,987	284,847	276,062	332,878	338,673	325,823	363,333	364,338	355,709
8.4	242,043	241,806	236,751	286,590	288,088	276,958	308,278	308,800	302,091
8.5	254,491	256,692	251,180	298,980	302,253	292,578	326,959	326,959	321,079
9.1	133,873	135,989	129,457	158,729	163,780	151,150	171,924	172,899	165,469
9.2	234,172	234,223	227,102	421,037	282,819	265,479	298,841	299,276	291,376
9.3	695,777	774,221	536,508	888,121	652,556	626,459	697,843	702,895	692,297
9.4	559,809	563,784	551,872	n/a	678,342	645,194	715,748	716,514	708,982
9.5	504,555	255,352	249,211	310,738	305,694	291,015	325,841	329,158	320,049
10.1	626,022	355,448	347,562	690,072	406,418	384,081	789,422	437,634	419,279
10.2	461,634	178,690	171,808	474,634	206,965	190,608	315,734	223,874	208,407
10.3	468,960	196,876	190,222	340,632	230,216	211,112	515,210	250,919	229,083
10.4	623,387	398,512	386,754	740,887	459,306	428,503	755,287	499,310	467,975
10.5	483,444	404,962	395,621	719,883	461,617	438,929	783,983	506,241	480,161
11.1	192,507	186,081	180,618	260,593	224,901	213,074	237,981	238,749	230,305
11.2	429,456	408,840	397,635	668,043	484,515	463,268	512,950	514,910	504,659
11.3	86,531	87,225	79,120	103,754	104,459	92,879	110,547	110,538	100,344
11.4	89,090	89,554	81,692	104,572	107,604	94,823	114,772	114,772	103,472
11.5	139,080	86,536	78,766	113,854	103,028	92,133	110,572	111,109	99,477
12.1	67,856	66,697	59,659	80,738	80,863	69,295	86,354	86,449	75,859
12.2	80,422	79,667	70,906	92,109	93,834	81,619	99,887	100,021	89,032
12.3	43,047	40,872	32,285	51,625	51,113	37,825	54,308	54,594	41,091
12.4	141,343	141,567	135,233	170,363	168,752	156,745	180,876	180,808	170,854
12.5	132,240	131,379	125,874	160,149	155,228	146,271	165,947	165,947	159,434

Table 6.22: Solutions found by SSCG and CPLEX and the lower bound regarding to enough resource scenario.

Instance	$F_l = 1000$		$F_l = 3000$		$F_l = 5000$	
	CPLEX	SSCG	CPLEX	SSCG	CPLEX	SSCG
6.1	1.86%	2.18%	1.45%	3.50%	1.49%	3.51%
6.2	3.51%	3.59%	5.09%	4.28%	6.7%	5.88%
6.3	1%	1.59%	1.27%	2.87%	1.34%	3.48%
6.4	1%	2.25%	2%	2.54%	1.44%	2.68%
6.5	0.67%	2.52%	0.88%	3.45%	1.1%	-1.66%
7.1	n/a	1.31%	4.64%	2.40%	1.36%	4.01%
7.2	4.43%	2.26%	n/a	4.03%	n/a	3.75%
7.3	40.87%	3.97%	9.12%	5.28%	9.64%	5.63%
7.4	5.08%	2.33%	1.93%	5.30%	1.82%	3.70%
7.5	25.61%	1.61%	4.17%	2.93%	12.32%	3.38%
8.1	3.13%	2.16%	1.97%	3.18%	1.33%	1.65%
8.2	8.34%	9.09%	9.91%	11.95%	8.47%	10.60%
8.3	1.4%	3.08%	1.99%	3.79%	1.84%	2.37%
8.4	2.19%	2.09%	3.36%	3.86%	1.78%	2.17%
8.5	1.3%	2.15%	2.14%	3.20%	1.22%	1.80%
9.1	3.3%	4.80%	4.77%	7.71%	3.75%	4.30%
9.2	3.02%	3.04%	36.95%	6.13%	2.5%	2.64%
9.3	22.89%	30.70%	29.47%	4.00%	0.79%	1.51%
9.4	1.42%	2.11%	n/a	4.89%	0.9%	1.05%
9.5	50.61%	2.40%	6.35%	4.80%	1.48%	2.77%
10.1	44.48%	2.22%	44.35%	5.50%	46.89%	4.19%
10.2	62.78%	3.85%	59.84%	7.90%	33.99%	6.91%
10.3	59.44%	3.38%	38.02%	8.30%	55.54%	8.70%
10.4	37.96%	2.95%	42.16%	6.71%	38.04%	6.28%
10.5	18.17%	2.31%	39.03%	4.91%	38.75%	5.15%
11.1	6.18%	2.94%	18.24%	5.26%	3.23%	3.54%
11.2	7.41%	2.74%	30.59%	4.39%	1.21%	1.99%
11.3	8.56%	9.29%	10.48%	11.09%	8.84%	9.22%
11.4	8.3%	8.78%	9.14%	11.88%	7.1%	9.85%
11.5	43.37%	8.98%	19.08%	10.57%	9.89%	10.47%
12.1	12.08%	10.55%	14.17%	14.31%	10.97%	12.25%
12.2	10.8%	11.00%	10.36%	13.02%	9.99%	10.99%
12.3	23.38%	21.01%	24.17%	26.00%	16.6%	24.73%
12.4	4.28%	4.47%	7.99%	7.12%	4.49%	5.51%
12.5	4.81%	4.19%	8.67%	5.77%	3.61%	3.92%

Table 6.23: Optimal gaps regarding to enough resource scenario.

Instance	$F_l = 1000$			$F_l = 3000$			$F_l = 5000$		
	CPLEX	SSCG	LB	CPLEX	SSCG	LB	CPLEX	SSCG	LB
6.1	149,161	152,159	146,895	168,123	172,825	165,780	185,979	188,729	182,983
6.2	42,717	42,793	40,855	49,100	48,687	45,632	53,134	53,707	50,169
6.3	159,695	163,494	157,700	180,067	183,520	177,523	199,655	202,629	197,140
6.4	180,926	184,296	178,288	203,470	207,726	200,069	225,370	228,896	221,850
6.5	159,928	163,300	158,752	183,935	188,600	182,182	209,900	205,300	
7.1	n/a	317,482	309,733	380,231	358,648	343,501	381,238	389,303	373,915
7.2	n/a	404,100	390,651	477,684	449,967	433,538	507,121	491,509	474,596
7.3	134,260	83,982	79,557	93,766	94,482	87,851	100,162	101,822	96,092
7.4	132,450	132,296	128,203	287,457	146,103	141,021	163,018	160,543	152,909
7.5	337,280	343,177	330,834	n/a	384,386	366,281	n/a	421,093	400,206
8.1	303,484	300,546	293,805	345,060	348,498	338,772	371,678	371,858	365,546
8.2	65,789	67,598	59,745	78,038	78,052	68,969	82,921	82,921	74,179
8.3	289,057	292,873	282,859	337,367	343,533	328,805	363,333	363,646	355,709
8.4	261,505	250,835	240,036	285,182	288,531	278,568	308,278	310,216	302,091
8.5	262,336	264,018	253,099	307,746	307,828	293,768	326,959	327,521	321,079
9.1	143,931	147,537	132,638	160,812	167,886	153,197	172,117	172,053	165,469
9.2	318,341	252,006	230,770	278,401	285,478	268,380	298,841	301,474	291,376
9.3	542,227	564,349	536,508	n/a	655,517	626,459	698,085	699,427	692,297
9.4	562,061	574,580	553,267	899,444	669,373	645,194	715,250	720,967	708,982
9.5	419,865	265,703	252,367	299,164	307,333	293,057	325,841	328,205	320,049
10.1	597,872	368,725	354,024	673,772	422,678	389,683	746,628	449,720	424,875
10.2	358,134	183,969	174,411	465,684	210,715	192,555	339,231	225,558	210,210
10.3	1,693,468	205,200	193,910	419,167	230,892	213,111	302,010	249,544	231,049
10.4	683,837	405,653	389,354	714,787	459,424	429,239	706,687	497,091	468,711
10.5	648,647	422,928	405,543	666,701	480,699	447,604	689,757	514,632	488,718
11.1	201,672	193,842	185,534	223,643	227,366	214,850	238,061	239,684	230,305
11.2	410,510	420,136	405,773	496,700	487,846	465,980	512,820	519,691	504,659
11.3	90,851	92,003	82,330	105,688	105,566	94,528	110,547	111,150	100,344
11.4	93,674	93,272	82,776	110,653	109,827	95,169	114,772	114,772	103,472
11.5	108,984	88,848	80,180	104,695	105,045	92,331	110,572	110,297	99,477
12.1	71,822	69,660	59,917	82,018	81,696	69,749	86,186	86,270	75,859
12.2	85,236	84,350	72,490	94,113	95,471	82,832	100,021	99,887	89,032
12.3	45,995	44,328	33,334	53,449	52,484	38,303	54,316	54,249	41,091
12.4	147,795	150,152	139,749	171,067	172,973	159,801	180,983	180,808	170,854
12.5	134,590	136,019	128,139	155,967	159,302	146,440	166,069	165,771	159,434

Table 6.24: Solutions found by SSCG and CPLEX and the lower bound regarding to enough resource scenario.

Instance	$F_l = 1000$		$F_l = 3000$		$F_l = 5000$	
	CPLEX	SSCG	CPLEX	SSCG	CPLEX	SSCG
6.1	1.52%	3.46%	1.39%	4.08%	1.19%	3.04%
6.2	4.36%	4.53%	6.98%	6.27%	5.58%	6.59%
6.3	1.25%	3.54%	1.41%	3.27%	1.23%	2.71%
6.4	1.46%	3.26%	1.67%	3.69%	1.56%	3.08%
6.5	0.74%	2.79%	0.95%	3.40%	1.15%	100.00%
7.1	n/a	2.44%	9.66%	4.22%	1.92%	3.95%
7.2	n/a	3.33%	9.24%	3.65%	6.41%	3.44%
7.3	40.74%	5.27%	6.31%	7.02%	4.06%	5.63%
7.4	3.21%	3.09%	50.94%	3.48%	6.2%	4.76%
7.5	1.91%	3.60%	n/a	4.71%	n/a	4.96%
8.1	3.19%	2.24%	1.82%	2.79%	1.44%	1.70%
8.2	8.6%	11.62%	11.62%	11.64%	8.87%	10.54%
8.3	2.14%	3.42%	2.28%	4.29%	1.55%	2.18%
8.4	8.21%	4.31%	1.86%	3.45%	1.72%	2.62%
8.5	3.52%	4.14%	4.54%	4.57%	1.12%	1.97%
9.1	7.85%	10.10%	4.74%	8.75%	3.69%	3.83%
9.2	27.51%	8.43%	3.6%	5.99%	2.5%	3.35%
9.3	1.05%	4.93%	n/a	4.43%	0.83%	1.02%
9.4	1.56%	3.71%	28.27%	3.61%	0.88%	1.66%
9.5	39.89%	5.02%	2.04%	4.65%	1.32%	2.49%
10.1	40.79%	3.99%	42.16%	7.81%	43.09%	5.52%
10.2	51.3%	5.20%	58.65%	8.62%	38.03%	6.80%
10.3	88.55%	5.50%	49.16%	7.70%	23.5%	7.41%
10.4	43.6%	4.02%	39.95%	6.57%	33.67%	5.71%
10.5	37.48%	4.11%	32.86%	6.88%	29.15%	5.04%
11.1	8%	4.29%	3.86%	5.50%	3.26%	3.91%
11.2	1.15%	3.42%	6.18%	4.48%	1.24%	2.89%
11.3	9.38%	10.51%	10.07%	10.46%	8.68%	9.72%
11.4	10.04%	11.25%	13.57%	13.35%	7.31%	9.85%
11.5	26.43%	9.76%	11.81%	12.10%	9.92%	9.81%
12.1	16.57%	13.99%	14.68%	14.62%	10.44%	12.07%
12.2	14.09%	14.06%	11.58%	13.24%	10.14%	10.87%
12.3	27.52%	24.80%	26.42%	27.02%	16.43%	24.25%
12.4	5.29%	6.93%	6.59%	7.62%	5.01%	5.51%
12.5	4.36%	5.79%	6.11%	8.07%	3.8%	3.82%

Table 6.25: Optimal gaps regarding to not enough resource scenario.

7. CONCLUSION

Service network design refers to problems at tactical level of a transportation system. These problems focus on the selection of services to route shipments, the allocation of resources to operate these services (explicitly or implicitly) while satisfying a set of side constraints on resources, services and shipments. Our literature review shows that there exists several publications on service network design but the contributions concerning resource managements are still limited. Additionally, many of the papers addressing resource management focus on specific problems in particular domains, for example, Lai & Lo (2004) for ferry transportation, Smilowitz *et al.* (2003) for deferred items in air express, etc., while the contributions to generic settings are limited. This thesis contributes of studies on service network design with resource managements, presenting a new approach to existing problems, proposing new service network design models that capture new resource constraints and developing efficient solution methods for these problems.

The thesis focuses on service network design problems with capacitated multi-commodity fixed cost structure. This structure represents different commodities which need to be routed over a capacitated service network. Resources are used to operate services that transport all the commodities from their origins to their destinations. In these problems, it is typically assumed that a single unit of resource is required to operate a service, resources are assigned to terminals to which they must ultimately returned, there is a finite number of resources assigned to each terminal, and the length of the resource circuits is restricted. In the thesis, we study three problems with the mentioned characteristics. The first problem is in fact a basic constituent of many problems in the field. Research regarding this problem may contribute of important tools and results to future research, which is why we select this problem as our first research problem. The second problem extends a generic setting described in literature which is used to capture the limitation of resources at each terminal. The third problem further extends this generic setting to capture resource reposition and capacity management issues.

In more details, the first research problem addresses the capacitated multi-commodity fixed cost network design problem with design-balance constraints (DBCMND). This problem appears as a basic sub-structure of many service network design problems including in our last two research problems. However, efficient solution methods for this problem have not been identified. Obtaining feasible solutions are “far from trivial” - Pedersen *et al.* (2009). Existing approaches for the DBCMND have difficulties in handling design-balance constraints - Smilowitz *et al.* (2003), or require many iterations to obtain feasible solution from an infeasible solution via neighborhood explorations - Pedersen *et al.* (2009). We have handled

design-balance constraints by modeling this problem as a graph problem and a minimum cost, maximum flow model. We proposed a three-phase matheuristic which consists of a tabu-search, a path-relinking procedure and an exact optimization. The matheuristic we also enhance the exploration of the search by taking into account the weaknesses of existing procedures in literature, e.g. (Pedersen *et al.*, 2009) and propose ways to improve them. This matheuristic use the path-relinking procedure with a simple neighborhood to enrich the set of feasible solutions. Finally, we show how we can further improve solution quality by employing the power of mixed integer solver on a small restricted DBCMND by exploiting information from feasible solutions and applying simple weighted fixing arc schemes. Experimentations show that the proposed algorithm is able to find good feasible solutions and that all the components contribute to the success of the algorithm. Comparing to the state-of-art tabu search presented in Pedersen *et al.* (2009), our study presents a more simple and more efficient method to address the design-balance constraints as well as to obtain feasible solutions in general. The method is easy to apply to other problems, e.g., the one described in Smilowitz *et al.* (2003), we show its application in our second and third study.

In the second research problem, named SNDRC, we extend existing service network design models that manage facility-based resources to route commodities to include limitations on the number of resources available at each terminal. Based on different optimization techniques, we present a matheuristic solution framework that includes column generation, slope-scaling, heuristic and exact optimization. We introduce a cycle-based formulation and a corresponding solution approach for this problem. The cycle-based solution approach generates cycles that appear in high-quality solutions and use them in an effective manner.

Solution methods for cycle-based formulations have been developed based on exact-based approach such as branch-and-price (Andersen *et al.*, 2011) or Benders decomposition (Agarwal & Ergun, 2008). These approaches are often complex to implement and depend on adhoc branching techniques to find feasible solutions. Moreover, when dealing with network design problems with multi-commodities, these approaches have several limitations, including large computational times. In our work, we rely rather on slope-scaling, a technique that has been proved promising for classical network design models despite of its simplicity. Our idea for the solution framework is that from a solution to the approximation problem of the formulation, which is a set of cycles, we will convert these cycles into feasible solutions. The difficulties in implementing this idea did not only came from the complexity of the service network design problem but also from the lack of work regarding cycle-based formulations with complex

constraint sets. For example, we had to obtain good subsets of cycles in order to develop the approximation problem. We solved this issue by using column generation to generate cycles. Then to obtain feasibility, we proposed heuristics which inherit ideas based on minimum-cost maximum flow model and depth-first search strategy.

The contribution from our work on this research problem includes not only a new problem setting that enriches the range of resources management constraints but also a generic solution framework that can be exploited to develop solution methods for other service network design problems. The solution method proposed in this second research problem also extends the application of cycle-based formulations for service network design problems. We showed that the set of cycles generated by column generation is useful to define the approximation problem. Next, we demonstrate how to define, initialize and update the cycle-based approximation formulation using the set of of cycles generated by column generation. We also show how to customize the initialization of the linearization factors using solution of the linear relaxation problem. Finally, we show how we can obtain solution of a cycle-based service network design formulation by a set of heuristic steps. The experiments indicated that the method is able to generate good feasible solutions even for large-size network instances with many commodities.

Finally, we develop a model that supports the integration of resource locations and capacity management in service network design. The goal of the problem, named LWSND, is to answer questions related to the allocation of resources and outsourcing of services. For example, how should resources be repositioned between terminals to prepare for the next transportation season, or how to buy/rent resources and allocate them to terminals. To answer these questions, we propose a problem setting and a model which address 1) repositioning of resources among terminals to account for shifts in demand patterns; 2) acquire (buy or long-term rent) new resources and assign them to terminals; 3) outsource particular services. We extend the time-space network to model resource acquisition and reposition decisions while decisions on cycle and service selections, demand satisfaction are modeled in the rest of the network. Based on the algorithm developed in our second study, we propose a solution approach which includes slope-scaling, heuristics and exact optimization for this new problem.

The contribution of this problem not only includes a new problem setting and formulation that integrates resources allocation issues but also again highlights the contribution of the second research problem by showing how we can exploit the solution framework introduced in the second problem to develop a solution framework for this problem. We are continuing to work on this problem to enhance the obtained results as well as to understand more the effect

of the resources allocation operations to service network design models.

In summary, regarding the subject and the scope of the thesis, our contributions enrich the current research literature on service network design with resources management by, first, proposing an efficient solution method for the capacitated multi-commodity fixed cost network design with design balance constraints (DBCMND), a basic and important problem belonging to the class of service network design with resource management problems, and second, proposing two new problem settings, named SNRDC and LWSND, with corresponding mathematical models and solution methods. SNDRC take into accounts the bound of resources at each terminal while LWSND covers the reposition of resources, acquisition of capacity and the outsourcing of services. The solution method for DBCMND is efficient not only for the problem itself but also for other problems having the design-balance property. In particular, the method demonstrates its applications in our studies for SNDRC and LWSND. Regarding SNDRC and LWSND, we demonstrate the first time the combination of slope-scaling, column generation, and heuristics to find feasible solution method for cycle-based formulations with complex constraint sets. The solution methods are proved to be efficient for our research problems. In addition, these methods present new techniques to find feasible solutions regarding cycle-based formulations. This will extend the application of our research to other problems of this class.

To conclude, we present some research directions that can be developed based on the results of thesis. These research subjects will extend the contributions of the thesis as well as enrich the knowledge related to service network design problems.

For DBCMND, one direction is to integrate the feasibility handling scheme into other search frameworks such as slope-scaling, guided local search to develop a more efficient algorithm for DBCMND. For SNDRC and LWSND, one goal is to improve the execution of cycle-extraction procedure, e.g., by integrating heuristic ideas and cost functions into the cycle-extraction procedure. We can further improve the algorithm by considering several cycle-extraction procedures which work together in a solution framework. Other research issues is to develop a more efficient solution method for these problems, especially regarding instances with thousand of commodities.

Other direction which will further strengthen the research on service network design is to extend the models to represent other practical resource management issues. We could extend the model to allow multiple resources operating a service which appears in truck-based carriers.

We can also extend the model to reflect work rules in some carriers in which a resource need to return frequently to its base during its journey, e.g. short route-length constraints. Integrating stochasticity into these studies would also support the robustness of the model regarding the randomness of future demands. These researches will not only enrich the research related to service network design with resource management but also extend its application to the operation of carriers.

BIBLIOGRAPHY

2006. *POLCORRIDOR project*. http://www.eurekanetwork.org/project/-/id/2727.

2013. *US Hour-of-Service Rules*. http://www.fmcsa.dot.gov/rules-regulations/topics/hos/index.htm.

Agarwal, R., & Ergun, Ö. 2008. Ship Scheduling and Network Design for Cargo Routing in Liner Shipping. *Transportation Science*, **42**(2), 175–196.

Ahuja, R.K., Magnanti, T.L., & Orlin, J.B. 1994. Network Flows: Theory, Algorithms, and Applications. *Journal of the Operational Research Society*, **45**(11), 1340–1340.

Andersen, J., & Christiansen, M. 2009. Designing New European Rail Freight Services. *Journal of the Operational Research Society*, **60**, 348–360.

Andersen, J., Crainic, T. G., & Christiansen, M. 2009a. Service Network Design with Asset Management: Formulations and Comparative Analyses. *Transportation Research Part C: Emerging Technologies*, **17**(2), 197 – 207.

Andersen, J., Crainic, T. G., & Christiansen, M. 2009b. Service Network Design with Management and Coordination of Multiple Fleets. *European Journal of Operational Research*, **193**(2), 377 – 389.

Andersen, J., Christiansen, M., Crainic, T. G., & Grønhaug, R. 2011. Branch and Price for Service Network Design with Asset Management Constraints. *Transportation Science*, **45**(1), 33–49.

Archetti, C., Speranza, M.G., & Savelsbergh, M.W.P. 2008. An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem. *Transportation Science*, **42**(1), 22–31.

Armacost, A.P, Barnhart, C., Ware, K.A, & Wilson, A.M. 2004. UPS Optimizes Its Air Network. *Interfaces*, **34**(1), 15–25.

Atamtürk, A. 2000. On Capacitated Network Design Cut-Set Polyhedra. *Mathematical Programming*, **92**(3), 425–437.

Atamtürk, A., & Rajan, D. 2002. On Splittable and Unsplittable Flow Capacitated Network Design Arc-set Polyhedra. *Mathematical Programming*, **92**(2), 315–333.

Avella, P., Mattia, S., & Sassano, A. 2007. Metric Inequalities and the Network Loading Problem. *Discrete Optimization*, **4**(1), 103 – 114.

Bai, R., Wallace, S. W., Li, J., & Chong, A. Yee-Loong. 2014. Stochastic Service Network Design with Rerouting. *Transportation Research Part B: Methodological*, **60**(0), 50 – 65.

Balakrishnan, A., Magnanti, T. L., & Wong, R. T. 1989. A Dual-Ascent Procedure for Large-Scale Uncapacitated Network Design. *Operations Research*, **37**(5), pp. 716–740.

Balakrishnan, A., Magnanti, T.L., & Mirchandani, P. 1997. Network Design. *Pages 311–334 of:* Dell'Amico, M., Maffioli, F., & Martello, S. (eds), *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons, New York, NY.

Barahona, F. 1996. Network Design Using Cut Inequalities. *SIAM J. on Optimization*, **6**(3), 823–837.

Barnhart, C., & Schneur, R. R. 1996. Air Network Design for Express Shipment Service. *Operations Research*, **44**(6), pp. 852–863.

Barnhart, C., & Shen, S. 2005. Logistics Service Network Design for Time-Critical Delivery. *Pages 86–105 of:* Burke, Edmund, & Trick, Michael (eds), *Practice and Theory of Automated Timetabling V*. Lecture Notes in Computer Science, vol. 3616. Springer Berlin Heidelberg.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., & Vance, P.H. 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, **46**(3), 316–329.

Barnhart, C., Krishnan, N., Kim, D., & Ware, K. 2002. Network Design for Express Shipment Delivery. *Computational Optimization and Applications*, **21**(3), 239–262.

Bartolini, E., & Mingozzi, A. 2009. Algorithms for the Non-bifurcated Network Design Problem. *Journal of Heuristics*, **15**(3), 259–281.

Bektaş, T., & Crainic, T. G. 2008. A Brief Overview of Intermodal Transportation. *Chap. 28, pages 1–16 of:* Taylor, G.D. (ed), *Logistics Engineering Handbook*. Taylor and Francis Group, Boca Raton, FL, USA.

Benders, J.F. 1962. Partitioning Procedures for Solving Mixed-variables Programming Problems. *Computational Management Science*, **2**(1), 3–19.

Berger, D., Gendron, B., Potvin, J.Y., Raghavan, S., & Soriano, P. 2000. Tabu Search for a Network Loading Problem with Multiple Facilities. *Journal of Heuristics*, **6**(2), 253–267.

Büdenbender, K., Grunert, T., & Sebastian, H.J. 2000. A Hybrid Tabu Search/Branch-and-Bound Algorithm for the Direct Flight Network Design Problem. *Transportation Science*, **34**(4), 364–380.

Chouman, M., & Crainic, T.G. 2010. *A MIP-Tabu Search Hybrid Framework for Multicommodity Capacitated Fixed-Charge Network Design*. Tech. rept. CIRRELT-2010-31. Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et les Transports, Université de Montréal, Montréal, QC, Canada.

Chouman, M., & Crainic, T.G. 2013. Cutting-plane Matheuristic for Service Network Design with Design-balanced Requirements. *Transportation Science*. Forthcoming.

Chouman, M., Crainic, T.G., & Gendron, B. 2003. *A Cutting-Plane Algorithm Based on Cutset Inequalities for Multicommodity Capacitated Fixed Charge Network Design*. Tech. rept. CRT-2003-16. Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et les Transports, Université de Montréal, Montréal, QC, Canada.

Chouman, M., Crainic, T.G., & Gendron, B. 2009. *A Cutting-Plane Algorithm for Multicommodity Capacitated Fixed Charge Network Design*. Tech. rept. CIRRELT-2009-20. Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et les Transports, Université de Montréal, Montréal, QC, Canada.

Christiansen, M., Fagerholt, K., & Ronen, D. 2004. Ship Routing and Scheduling: Status and Perspectives. *Transportation Science*, **38**(1), 1–18.

Christiansen, M., Fagerholt, K., Nygreen, B., & Ronen, D. 2007. Maritime Transportation. *Pages 189–284 of:* Barnhart, C., & Laporte, G. (eds), *Transportation*. Handbooks in Operations Research and Management Science, vol. 14. North-Holland, Amsterdam.

Contreras, I., & Fernandez, E. 2012. General Network Design: A Unified View of Combined Location and Network Design Problems. *European Journal of Operational Research*, **219**(3), 680 – 697.

Cordeau, J. F., Toth, P., & Vigo, D. 1998. A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science*, **32**(4), 380–404.

Costa, A. M. 2005. A Survey on Benders Decomposition Applied to Fixed-Charge Network Design Problems. *Computers & Operations Research*, **32**(6), 1429 – 1450.

Crainic, T. G. 2000. Service Network Design in Freight Transportation. *European Journal of Operational Research*, **122**(2), 272–288.

Crainic, T. G., & Bektas, T. 2008. *Logistics Engineering Handbook*. Taylor and Francis Group. Chap. A brief overview of intermodal transportation, pages 1–16.

Crainic, T. G., & Kim, K. H. 2007. Intermodal Transportation. *Transport, Handbooks in Operations Research and Management Science.*, **14**, 467–537.

Crainic, T. G., & Laporte, G. 1997. Planning Models for Freight Transportation. *European Journal of Operational Research*, **97**(3), 409–438.

Crainic, T. G., Gendron, B., & Hernu, G. 2004. A Slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design. *Journal of Heuristics*, **10**(5), 525–545.

Crainic, T. G., Hewitt, M., Toulouse, M., & Vu, D. M. 2013. Service Network Design with Resources Constraints. *Transportation Science*. To appear.

Crainic, T.G. 2003. Long-Haul Freight Transportation. *Pages 451–516 of:* Hall, R.W. (ed), *Handbook of Transportation Science*, second edn. Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G. 2005. Parallel Computation, Co-operation, Tabu Search. *Pages 283–302 of:* Sharda, Ramesh, Voay, Stefan, Rego, Ceasar, & Alidaee, Bahram (eds), *Metaheuristic Optimization via Memory and Evolution*. Operations Research/Computer Science Interfaces Series, vol. 30. Springer US.

Crainic, T.G., & Gendreau, M. 2002. Cooperative Parallel Tabu Search for Capacitated Network Design. *Journal of Heuristics*, **8**(6), 601–627.

Crainic, T.G., & Gendreau, M. 2007. A Scatter Search Heuristic for the Fixed-Charge Capacitated Network Design Problem. *Pages 25–40 of: Metaheuristics*. Operations Research/-Computer Science Interfaces Series, vol. 39. Springer US.

Crainic, T.G., & Nourredine, H. 2005. Parallel Meta-Heuristics Applications. *Pages 447–494 of:* Alba, E. (ed), *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, Hoboken, NJ.

Crainic, T.G., & Toulouse, M. 1998. Parallel Metaheuristics. *Pages 205–251 of:* T.G. Crainic, & G. Laporte (eds), *Fleet Management and Logistics*. Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G., & Toulouse, M. 2003. Parallel Strategies for Meta-heuristics. *Pages 475–513 of:* F. Glover, & G. Kochenberger (eds), *Handbook in Metaheuristics*. Kluwer Academic Publishers, Norwell, MA.

Crainic, T.G, & Toulouse, M. 2010. Parallel Meta-Heuristics. *Pages 497–541 of:* Gendreau, M., & Potvin, J.-Y. (eds), *Handbook of Metaheuristics (2nd Edition)*. Springer.

Crainic, T.G., Ferland, J.A., & Rousseau, J.M. 1984. A Tactical Planning Model for Rail Freight Transportation. *Transportation Science*, **18**(2), 165–184.

Crainic, T.G., Gendreau, M., & Farvolden, J.M. 2000. A Simplex-Based Tabu Search Method for Capacitated Network Design. *INFORMS Journal on Computing*, **12**(3), 223–236.

Crainic, T.G., Frangioni, A., & Gendron, B. 2001. Bundle-Based Relaxation Methods for Multicommodity Capacitated Network Design. *Discrete Applied Mathematics*, **112**(1–3), 73–99.

Crainic, T.G., Gendreau, M., & Potvin, J.Y. 2005. Parallel Tabu Search. *Pages 298–313 of:* Alba, E. (ed), *Parallel Metaheuristics*. John Wiley & Sons, Hoboken, NJ.

Crainic, T.G., Li, Y., & Toulouse, M. 2006. A First Multilevel Cooperative Algorithm for the Capacitated Multicommodity Network Design. *Computers & Operations Research*, **33**(9), 2602–2622.

De Franceschi, R, Fischetti, M., & Toth, P. 2006. A New ILP-based Refinement Heuristic for Vehicle Routing Problems. *Mathematical Programming B*, **105**(2-3), 471–499.

Desaulniers, G., Desrosiers, J., & Solomon, M.M. (eds). 2005. *Column Generation*. Springer-Verlag, New York.

Erera, A., Hewitt, M., Savelsbergh, M., & Zhang, Y. 2012. Improved Load Plan Design Through Integer Programming Based Local Search. *Transportation Science*, **to appear**(3).

Ergun, Ö., Kuyzu, G., & Savelsbergh, M. W. P. 2007. Reducing Truckload Transportation Costs Through Collaboration. *Transportation Science*, **41**(2), 206–221.

Favolden, J.M., & Powell, W.B. 1994. Subgradient Methods for the Service Network Design Problem. *Transportation Science*, **28**(3), pp. 256–272.

Fischetti, M., & Lodi, A. 2003. Local Branching. *Mathematical Programming*, **98**(1-3), 23–47.

Frangioni, A., & Gendron, B. 2009. 0-1 Reformulations of the Multicommodity Capacitated Network Design Problem. *Discrete Applied Mathematics*, **157**(6), 1229 – 1241. Reformulation Techniques and Mathematical Programming.

Gendron, B., & Crainic, T.G. 1994. *Relaxations for Multicommodity Network Design Problems*. Publication CRT-965. Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et les Transports, Université de Montréal, Montréal, QC, Canada.

Gendron, B., & Crainic, T.G. 1996. *Bounding Procedures for Multicommodity Capacitated Network Design Problems*. Publication CRT-96-06. Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et les Transports, Université de Montréal, Montréal, QC, Canada.

Ghamlouche, I., Crainic, T. G., & Gendreau, M. 2003. Cycle-Based Neighbourhoods for Fixed-Charge Capacitated Multicommodity Network Design. *Operations Research*, **51**(4), 655–667.

Ghamlouche, I., Crainic, T.G, & Gendreau, M. 2004. Path Relinking, Cycle-based Neighbourhoods and Capacitated Multicommodity Network Design. *Annals of Operations Research*, **131**(1-4), 109–133.

Glover, F. 1989. Tabu Search – Part I. *ORSA Journal on Computing*, **1**(3), 190–206.

Glover, F. 1990. Tabu Search – Part II. *ORSA Journal on Computing*, **2**(1), 4–32.

Glover, F. 1997. A Template for Scatter Search and Path Relinking. *Pages 13–54 of:* Hao, J.K., Lutton, E., Ronald, E., Schoenauer, M., & Snyers, D. (eds), *Artificial Evolution*. Lecture Notes in Computer Science, vol. 1363. Springer Verlag, Berlin.

Glover, F., & Laguna, M. 1997. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA.

Glover, F., Laguna, M., & Martí, R. 2000. Fundamentals of Scatter Search and Path Relinking. *Control and Cybernetics*, **39**(3), 653–684.

Haghani, A.E. 1989. Formulation and Solution of a Combined Train Routing and Makeup, and Empty Car Distribution Model. *Transportation Research Part B: Methodological*, **23**(6), 433 – 452.

Hewitt, M., Nemhauser, G.L., & Savelsbergh, M.W.P. 2010. Combining Exact and Heuristic Approaches for the Capacitated Fixed-Charge Network Flow Problem. *INFORMS Journal on Computing*, **22**(2), 314–325.

Hoff, A., L., A.-G., Lokketangen, A., & Crainic, T.G. 2010. A Metaheuristic for Stochastic Service Network Design. *Journal of Heuristics*, **16**(5), 653–679.

Holmberg, K., & Hellstrand, J. 1998. Solving the Uncapacitated Network Design Problem by a Lagrangian Heuristic and Branch-and-Bound. *Operations Research*, **46**(2), 247–259.

Holmberg, K., & Yuan, D. 2000. A Lagrangean Heuristic Based Branch-and-Bound Approach for the Capacitated Network Design Problem. *Operations Research*, **48**(3), 461–481.

Kim, D., & Pardalos, P.M. 1999. A Solution to the Fixed Charge Network Flow Problem Using a Dynamic Slope Scaling Procedure. *Operations Research Letters*, **24**(4), 195–203.

Kim, D., Barnhart, C., Ware, K., & Reinhardt, G. 1999. Multimodal Express Package Delivery: a Service Network Design Application. *Transportation Science*, **33**(4), 391–407.

Lai, M. F., & Lo, H. K. 2004. Ferry Service Network Design: Optimal Fleet Size, Routing, and Scheduling. *Transportation Research Part A: Policy and Practice*, **38**(4), 305–328.

Lin, C.-C.and Chen, Y.C. 2003. The Integration of Taiwanese and Chinese Air Networks for Direct Air Cargo Services. *Transportation Research Part A: Policy and Practice*, **37**(7), 629 – 647.

Lium, A.-G., Crainic, T.G., & Wallace, S.W. 2007. Correlations in Stochastic Programming: A Case from Stochastic Service Network Design. *Asia-Pacific Journal of Operational Research*, **24**(2), 161–179.

Lium, A.G., Crainic, T.G, & Wallace, S.W. 2009. A Study of Demand Stochasticity in Service Network Design. *Transportation Science*, **43**(2), 144–157.

Magnanti, T.L., & Wong, R.T. 1984. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science*, **18**(1), 1–55.

Magnanti, T.L., Mireault, P., & Wong, R.T. 1986. Tailoring Benders Decomposition for Uncapacitated Network Design. *Mathematical Programming Study*, **26**, 112–154.

Melkote, S., & Daskin, M.S. 2001a. Capacitated Facility Location/Network Design Problems. *European Journal of Operational Research*, **129**(3), 481 – 495.

Melkote, S., & Daskin, M.S. 2001b. An Integrated Model of Facility Location and Transportation Network Design. *Transportation Research Part A: Policy and Practice*, **35**(6), 515 – 538.

Minoux, M. 1989. Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications. *Networks*, **19**(3), 313–360.

Ortega, F., & Wolsey, L. 2000. *A Branch-and-cut Algorithm for the Single Commodity Uncapacitated Fixed Charge Network Flow Problem*. CORE Discussion Papers 2000049. Université catholique de Louvain, Center for Operations Research and Econometrics (CORE).

Pedersen, M. B., Crainic, T. G., & Madsen, O. B. G. 2009. Models and Tabu Search Metaheuristics for Service Network Design with Asset-Balance Requirements. *Transportation Science*, **43**(2), 158–177.

Powell, W.B. 1986. A Local Improvement Heuristic for the Design of Less-than-Truckload Motor Carrier Networks. *Transportation Science*, **20**(4), 246–257.

Powell, W.B., & Sheffi, S. 1983. The Load Planning Problem of Motor Carriers: Problem Description and a Proposed Solution Approach. *Transportation Research Part A: General*, **17**(6), 471 – 480.

Powell, W.B., & Sheffi, Y. 1989. OR Practice Design and Implementation of an Interactive Optimization System for Network Design in the Motor Carrier Industry. *Operations Research*, **37**(1), 12–29.

Rei, W., Cordeau, J.F., Gendreau, M., & Soriano, P. 2009. Accelerating Benders Decomposition by Local Branching. *INFORMS Journal on Computing*, **21**(2), 333–345.

Resende, M.G.C., Ribeiro, C.C., Glover, F., & Marti, R. 2010. Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications. *Pages 87–107 of:* Gendreau,

Michel, & Potvin, Jean-Yves (eds), *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, vol. 146. Springer US.

Sebbah, S., & Jaumard, B. 2012. An Efficient Column Generation Design Method of P-cycle-based Protected Working Capacity Envelope. *Photonic Network Communications*, **24**(3), 167–176.

Smilowitz, K.R., Atamtürk, A., & Daganzo, C.F. 2003. Deferred Item and Vehicle Routing within Integrated Networks. *Transportation Research Part E: Logistics and Transportation Review*, **39**(4), 305–323.

Sridhar, V., & Park, J. S. 2000. Benders-and-cut Algorithm for Fixed-charge Capacitated Network Design Problem. *European Journal of Operational Research*, **125**, 622–632.

Sun, M., Aronson, J.E., McKeown, P.G, & Drinka, D. 1998. A Tabu Search Heuristic Procedure for the Fixed Charge Transportation Problem. *European Journal of Operational Research*, **106**(2-3), 441 – 456.

Tang, C.H., Yan, S., & Chen, Y.H. 2008. An Integrated Model and Solution Algorithms for Passenger, Cargo, and Combined Flight Scheduling. *Transportation Research Part E: Logistics and Transportation Review*, **44**(6), 1004 – 1024.

Teypaz, N., Schrenk, S., & Cung, V.D. 2010. A Decomposition Scheme for Large-scale Service Network Design with Asset Management. *Transportation Research Part E: Logistics and Transportation Review*, **46**(1), 156 – 170.

Thapalia, B. K., Crainic, T. G., Kaut, M., & Wallace, S.W. 2012. Single-commodity Network Design with Random Edge Capacities. *European Journal of Operational Research*, **220**(2), 394–403.

Vu, D.M., Crainic, T.G., & Toulouse, M. 2013. A Three-Phase Matheuristic for the Capacitated Multi-commodity Fixed-Cost Network Design with Design-Balance Constraints. *Journal of Heuristics*, **19**(5), 757–795.

Wang, D.Z.W., & Lo, H.K. 2008. Multi-fleet Ferry Service Network Design with Passenger Preferences for Differential Services. *Transportation Research Part B: Methodological*, **42**(9), 798 – 822.

Wolsey, Laurence A. 1998. *Integer Programming*. Wiley.

Yan, S, & Chen, Hao-Lei. 2002. A Scheduling Model and a Solution Algorithm for Inter-city Bus Carriers. *Transportation Research Part A: Policy and Practice*, **36**(9), 805 – 825.

Yan, S., & Tseng, C.H. 2002. A Passenger Demand Model for Airline Flight Scheduling and Fleet Routing. *Computers & Operations Research*, **29**(11), 1559 – 1581.

Yan, S., & Young, H.F. 1996. A Decision Support Framework for Multi-fleet Routing and Multi-stop Flight Scheduling. *Transportation Research Part A: Policy and Practice*, **30**(5), 379 – 398.

Yen, J.Y. 1971. Finding the k Shortest Loopless Paths in a Network. *Management Science*, **17**(11), 712–716.

Zhu, E., Crainic, T. G., & Gendreau, M. 2011. *Sheduled Service Network Design for Freight Rail Transportation*. Tech. rept. CIRRELT-2011-38. Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et les Transports, Université de Montréal.