Université de Montréal

Scheduled Service Network Design for Integrated Planning of Rail Freight Transportation

par

Endong Zhu

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 Philosophiæ Doctor (Ph.D.)
en informatique et recherche opérationnelle

Août, 2010

Université de Montréal
Faculté des arts et des sciences

Cette thèse intitulée:

Scheduled Service Network Design for Integrated Planning of Rail Freight Transportation

présentée par:

Endong Zhu

a été évaluée par un jury composé des personnes suivantes:

Jacques Ferland,	président-rapporteur
Michel Gendreau,	directeur de recherche
Teodor Gabriel Crainic,	codirecteur
Bernard Gendron,	membre du jury
Ali Haghani,	examinateur externe
Jacques Ferland,	représentant du doyen de la FAS

RÉSUMÉ

Cette thèse étudie une approche intégrant la gestion de l'horaire et la conception de réseaux de services pour le transport ferroviaire de marchandises. Le transport par rail s'articule autour d'une structure à deux niveaux de consolidation où l'affectation des wagons aux blocs ainsi que des blocs aux services représentent des décisions qui complexifient grandement la gestion des opérations. Dans cette thèse, les deux processus de consolidation ainsi que l'horaire d'exploitation sont étudiés simultanément. La résolution de ce problème permet d'identifier un plan d'exploitation rentable comprenant les politiques de blocage, le routage et l'horaire des trains, de même que l'habillage ainsi que l'affectation du traffic.

Afin de décrire les différentes activités ferroviaires au niveau tactique, nous étendons le réseau physique et construisons une structure de réseau espace-temps comprenant trois couches dans lequel la dimension liée au temps prend en considération les impacts temporels sur les opérations. De plus, les opérations relatives aux trains, blocs et wagons sont décrites par différentes couches. Sur la base de cette structure de réseau, nous modélisons ce problème de planification ferroviaire comme un problème de conception de réseaux de services.

Le modèle proposé se formule comme un programme mathématique en variables mixtes. Ce dernier s'avère très difficile à résoudre en raison de la grande taille des instances traitées et de sa complexité intrinsèque. Trois versions sont étudiées : le modèle simplifié (comprenant des services directs uniquement), le modèle complet (comprenant des services directs et multi-arrêts), ainsi qu'un modèle complet à très grande échelle. Plusieurs heuristiques sont développées afin d'obtenir de bonnes solutions en des temps de calcul raisonnables.

Premièrement, un cas particulier avec services directs est analysé. En considérant une caractéristique spécifique du problème de conception de réseaux de services directs nous développons un nouvel algorithme de recherche avec tabous. Un voisinage par cycles est privilégié à cet effet. Celui-ci est basé sur la distribution du flot circulant sur les blocs selon les cycles issus du réseau résiduel.

Un algorithme basé sur l'ajustement de pente est développé pour le modèle complet, et nous proposons une nouvelle méthode, appelée *recherche ellipsoïdale*, permettant d'améliorer davantage la qualité de la solution. La recherche ellipsoïdale combine les bonnes solutions admissibles générées par l'algorithme d'ajustement de pente, et regroupe les caractéristiques des bonnes solutions afin de créer un problème élite qui est résolu de façon exacte à l'aide d'un logiciel commercial. L'heuristique tire donc avantage de la vitesse de convergence de l'algorithme d'ajustement de pente et de la qualité de solution de la recherche ellipsoïdale. Les tests numériques illustrent l'efficacité de l'heuristique proposée. En outre, l'algorithme représente une alternative intéressante afin de résoudre le problème simplifié.

Enfin, nous étudions le modèle complet à très grande échelle. Une heuristique hybride est développée en intégrant les idées de l'algorithme précédemment décrit et la génération de colonnes. Nous proposons une nouvelle procédure d'ajustement de pente où, par rapport à l'ancienne, seule l'approximation des coûts liés aux services est considérée. La nouvelle approche d'ajustement de pente sépare ainsi les décisions associées aux blocs et aux services afin de fournir une décomposition naturelle du problème. Les résultats numériques obtenus montrent que l'algorithme est en mesure d'identifier des solutions de qualité dans un contexte visant la résolution d'instances réelles.

Mots clés: conception de réseaux de services, transport ferroviaire de marchandises, conception du réseau en fonction du temps.

ABSTRACT

This thesis studies a scheduled service network design problem for rail freight transportation planning. Rails follow a special two level consolidation organization, and the car-to-block, block-to-service handling procedure complicates daily operations. In this research, the two consolidation processes as well as the operation schedule are considered simultaneously, and by solving this problem, we provide an overall cost-effective operating plan, including blocking policy, train routing, scheduling, make-up policy and traffic distribution.

In order to describe various rail operations at the tactical level, we extend the physical network and construct a 3-layer time-space structure, in which the time dimension takes into consideration the temporal impacts on operations. Furthermore, operations on trains, blocks, and cars are described in different layers. Based on this network structure, we model the rail planning problem to a service network design formulation.

The proposed model relies on a complex mixed-integer programming formulation. The problem is very hard to solve due to the computational difficulty as well as the tremendous size of the application instances. Three versions of the problem are studied, which are the simplified model (with only non-stop services), complete model (with both non-stop and multi-stop services) and very-large-scale complete model. Heuristic algorithms are developed to provide good feasible solutions in reasonable computing efforts.

A special case with non-stop services is first studied. According to a specific characteristic of the direct service network design problem, we develop a tabu search algorithm. The tabu search moves in a cycle-based neighborhood, where flows on blocks are re-distributed according to the cycles in a conceptual residual network.

A slope scaling based algorithm is developed for the complete model, and we propose a new method, called *ellipsoidal search*, to further improve the solution quality. Ellipsoidal search combines the good feasible solutions generated from the slope scaling, and collects the features of good solutions into an elite problem, and solves it with exact solvers. The algorithm thus takes advantage of the convergence speed of slope scaling and solution quality of ellipsoidal search, and is proven effective. The algorithm also presents an alternative for solving the simplified problem.

Finally, we work on the very-large-size complete model. A hybrid heuristic is developed by integrating the ideas of previous research with column generation. We propose a new slope scaling scheme where, compared with the previous scheme, only approximate service costs instead of both service and block costs are considered. The new slope scaling scheme thus separates the block decisions and service decisions, and provide a natural decomposition of the problem. Experiments show the algorithm is good to solve real-life size instances.

Keywords: service network design, rail freight transportation, time-dependent network design.

CONTENTS

RÉSUMÉ . . . . **iii**

ABSTRACT . . . . **v**

CONTENTS . . . . **vii**

LIST OF TABLES . . . . **x**

LIST OF FIGURES . . . . **xi**

ACKNOWLEDGMENTS . . . . **xii**

CHAPTER 1: INTRODUCTION **1**

1.1 Rail Freight Transportation 1

1.2 Rail Operating Plan 5

1.3 Research Problem 7

1.4 Thesis Outline 9

CHAPTER 2: NETWORK DESIGN **11**

2.1 Network Design Problems 12

2.1.1 Commodity and Demand 12

2.1.2 Capacity 13

2.1.3 Flow Cost and Design Cost 13

2.1.4 Static / Dynamic 14

2.1.5 Mono-Layer / Multi-Layer Network 14

2.1.6 Deterministic / Stochastic 15

2.2 Network Design Models 15

2.2.1 Link-based Formulation 17

2.2.2 Path-based Formulation 19

2.3 Solution Methods 20

2.3.1 Uncapacitated Network Design 22
2.3.2 Capacitated Network Design 23
2.4 Summary 30

CHAPTER 3: SERVICE NETWORK DESIGN IN FREIGHT TRANSPORTATION 31
3.1 Freight Transportation System 31
3.2 Tactical Planning 34
3.3 Service Network Design 37
3.3.1 Service Network Design with Frequency 38
3.3.2 Service Network Design with Schedule 40
3.4 Summary 43

CHAPTER 4: RAIL FREIGHT OPERATION PLANNING 44
4.1 Blocking Policy 45
4.2 Train Routing, Scheduling, and Make-up 46
4.3 Empty Car Reposition 49
4.4 Resource Allocation 50
4.5 Integrated Models 51
4.6 Summary 53

CHAPTER 5: 3-LAYER TIME-SPACE NETWORK STRUCTURE AND MODELS 55
5.1 3-Layer Time-Space Network Structure 56
5.1.1 Service Layer and Service 57
5.1.2 Block Layer and Block 58
5.1.3 Car Layer 59
5.2 Integrated Service Network Design Model 62
5.3 Direct Service Network Design Model 67
5.4 Summary 68

CHAPTER 6: HEURISTIC ALGORITHMS . . . 70
6.1 Tabu Search . . . 70
6.1.1 Initialization . . . 73
6.1.2 Residual Network and Local Search . . . 74
6.1.3 Intensification . . . 82
6.1.4 Diversification . . . 83
6.2 Slope Scaling with Ellipsoidal Search . . . 84
6.2.1 Slope Scaling Procedure . . . 85
6.2.2 Factor Perturbation . . . 89
6.2.3 Ellipsoidal Search . . . 90
6.2.4 Overall Procedure . . . 94
6.3 A Hybrid Heuristic . . . 95
6.3.1 Outer Loop . . . 96
6.3.2 Inner Loop . . . 99
6.3.3 Overall Procedure . . . 104

CHAPTER 7: COMPUTATIONAL RESULTS AND DISCUSSION . . . 106
7.1 Results for Tabu Search . . . 107
7.1.1 Parameter Calibration . . . 108
7.1.2 Result Analysis . . . 110
7.2 Results for SS+ES+LMP . . . 112
7.2.1 Parameter Calibration . . . 113
7.2.2 Results Analysis . . . 118
7.3 Results for BEST . . . 124

CHAPTER 8: CONCLUSION . . . 128

BIBLIOGRAPHY . . . 132

LIST OF TABLES

5.I Notation Summary 66

7.I Instance Parameters: Set D-S 107
7.II Instance Parameters: Set D-M 108
7.III Instance Parameters: Set D-L 108
7.IV Tabu Search Parameter Calibration 109
7.V Tabu Search Results on Instance Set D-S 111
7.VI Tabu Search Results on Instance Set D-M 111
7.VII Tabu Search Results on Instance Set D-L 112
7.VIII Instance Parameter, Set C-S 113
7.IX Instance Parameter, Set C-L 113
7.X Perturbation Parameter Calibration 115
7.XI Ellipsoid Search Parameter Calibration 118
7.XII SS+LMP Results on Instance Set C-S 119
7.XIII SS+LMP Results on Instance Set C-L 120
7.XIV SS+ES+LMP Results on Instance Set C-S 121
7.XV SS+ES+LMP Results on Instance Set C-L 121
7.XVI SS+ES+LMP Results on Instance Set D-S 122
7.XVII SS+ES+LMP Results on Instance Set D-M 123
7.XVIII SS+ES+LMP Results on Instance Set D-L 123
7.XIX BEST Results on Instance Set C-S 125
7.XX BEST Results on Instance Set C-L 125
7.XXI Instance Parameters and Results, Set C-XL 126

I.I Instance Parameters: Set C-C xiii

LIST OF FIGURES

3.1 Illustration of Physical Network 41
3.2 Illustration of Time-Space Network 41

5.1 A Simple Rail Network 55
5.2 3-Layer Structure 56
5.3 Service Layer 58
5.4 Block Layer 60
5.5 Car Layer 61
5.6 Yard Section of 3-Layer Time-Space Network 62
5.7 Block Layer of 2-Layer Time-Space Network 68

6.1 Cycle-based Neighborhood Specification 71
6.2 General Procedure of Tabu Search 72
6.3 Construction of γ-Residual Network 75
6.4 Extended Bellman-Ford Algorithm 77
6.5 Flows in 2-Layer Time-Space Network (Projection) 78
6.6 Solution in 2-Layer Time-Space Network (Projection) 79
6.7 A Lowest-Cost Cycle in γ-Residual Network 80
6.8 New Solution on Neighbor Block Design (Projection) 81
6.9 Local Search Procedure in Cycle-based Neighborhood 82
6.10 Ellipsoidal Search Procedure 91
6.11 Illustration of Ellipsoid within k-opt Neighborhood 93
6.12 Slope Scaling + Ellipsoidal Search + Long-term Memory Perturbation 95
6.13 Outer Loop Procedure 99
6.14 Inner Loop Procedure 105

7.1 SS+LMP, Comparison of R1-ISND Evolutions 116
7.2 SS+LMP, Comparison of ISND Evolutions 117

ACKNOWLEDGMENTS

This dissertation is the result of contributions from many people in different ways. I am deeply grateful to all of them.

First of all, I would like to thank Prof. Michel Gendreau for his patience and advice, and Prof. Theodore Gabriel Crainic for his support and guidance. Not only shall the operations research knowledge, also the research attitude I learned from them be remembered for the rest of my life. I also would thank for my jury members for their invaluable comments.

Sincere thanks to my friends in CIRRELT, their help, encouragement, entertainment have made my time spent in Montréal a wonderful experience. I really enjoyed the insightful conversations with Dr. Mervat Chouman, Dr. Nadia Lahrichi, Dr. Walter Rei, especially my officemate and good friend, Dr. Benoit Crevier. I also would thank Serge Bisaillon, Daniel Charbonneau and Luc Rocheleau for the help with programming and technical support.

I could not thank enough for my parents for their support and encouragement. Most importantly, I am heartfelt grateful to my beloved wife, Ding Yuan, who provides a shelter for me with affection and support. I will never reach this point without her.

CHAPTER 1

INTRODUCTION

Since the early 1960s, network design problems have received much attention in the operations research society because of their significance and variety of substantial applications. The range of network design applications keeps getting broader in recent years, including transportation networks [e.g. 34]; telecommunication networks [e.g. 84]; centralized teleprocessing networks [e.g. 76]; energy systems [e.g. 21], etc.

Service network design in transportation represents a major application of network design. The service network design problem is very concerned to transportation service providers who offer, for example air services, truck services, as it provides a very innate tool given the intuitive network presentation of most transportation paradigms.

In this research, we study a *service network design* problem, aiming at generating an outline of the operating plan for rail freight transportation. In this chapter, we first introduce the background of railway transportation, and present our research problem.

1.1 Rail Freight Transportation

As a major category of freight transportation in modern society, rail freight transportation plays an important role in the world's economy. Only in Canada, Mexico, and the United States, more than $3,600$ railroads operate over $270,000$km of track [69], and several top ones (Class I rails) generate a total annual revenue of 67.4 billion USD [1].

As a mainstay of transportation for many basic industries, rail distinguishes itself with other transportation ways (e.g. airlines, shipping lines) by its particular infrastructure and complex operations.

Railways operate on a physical rail network, consisting of terminals and tracks. Rail terminals include *stations* and *yards*. Stations are dedicated to car exchange with customers. At the origin station, customers take empty cars and load the cars with commodities to be transported; at the destination station, loaded cars are cleared by customers and empty cars are returned. Generally, the station where a customer takes empty/loaded cars and the station where loaded/empty cars are returned is the same, with exceptions if it benefits the transportation afterwards. In yards, rails perform major handling operations, for example, car classification, blocking, train assembly and disassembly, etc. Terminals in the physical rail network are linked by tracks. Tracks include main-line tracks connecting major yards, and branch (also known as local or secondary) lines which attach the stations with their nearby yard. Generally, given the considerable construction cost and land occupancy, most rail networks present a hub-and-spoke distribution paradigm.

The *demand* in rail freight transportation is usually expressed in terms of tonnage or cars of certain commodities to be moved from one terminal to another. Given the special commodity species, appropriate car type has to be employed. For example, hopper cars for bulk loads and tank cars for liquid. For a commodity type, there might be several types of cars which are feasible, e.g. double-stack and flat cars for containers. In a modern rail company, there exist around 20 types of cars which are commonly exploited. Car loads together with the appropriate car type employed are thus treated as the basic traffic flow unit on tracks.

Every loaded movement on a rail network leads to a supply of empty cars at destination. Therefore, if transportation demand is unbalanced, empty cars must be repositioned to avoid their accumulation in some parts of the network where more traffic is directed. Repositioning empty freight cars can thus help rails offer better transportation service to its customers by reducing the average time they have to wait for empty cars, and decrease the capital investment associated with equipment ownership.

Transportation services provided by rail carriers are implemented by trains. Each train consists of one or more locomotives, hauling a queue of cars (both loaded and empty). From ancient steam to advanced electric or hybrid power, locomotives provide motive power for moving cars on tracks. Multiple locomotives can be jointed to achieve a higher hauling capacity; however, the connection of locomotives is restricted by the engine type, sequence, or even the position in the car queue.

Hauled by locomotives, a train follows a physical route from its origin where the train assembles to its destination where the train disassembles. Feeder trains on branch lines are used to provide transportation between the customer's station and the yard designated, and long-haul trains provide transportation among yards. Due to the expensive crew charge and engine depreciation cost, dedicated trains are only offered for important customers with large and regular demands, and consolidation-based trains are broadly applied otherwise, commonly for long-haul transportation. Consolidation is a widely applied concept in many transportation industries, meaning service provider group shipments from different customers, load the shipments into the same vehicle, and transport them together in order to reduce the operating cost.

During the journey of a train, there might be stops where some cars are loaded to or unloaded from the train. Loading/unloading a car to a train is realized by attaching/detaching the car to/from the very end of the car queue. Operating on tracks, unloading a car somewhere in the middle of the car queue is a burden: all tail cars "after" the target car must be detached and temporarily stored on a siding track in the terminal, remove the target car, and re-attach the tail cars. Such unloading operations become a major source of inefficiency and unreliability for rail transportation when multiple discrete cars must be removed from a train. To reduce the redundant yard operations, rails build an intermediate formation of commodities, called *block*. A block is associated with an origin-destination pair and consists of a group of cars. The block origin and destination may or may not be the origin or destination of any car in the block. However, cars in a block are treated as a unit, and share the common portion of their trips during the block life cycle. As a result, trains are made of blocks, and blocks are made of cars. Rails thus work under a special double consolidation organization: loaded and empty cars are grouped to build blocks, and blocks are grouped to make up trains.

The first level consolidation is grouping cars into blocks. To build blocks, cars with various commodities and different origins and destinations are classified (or sorted) in the appropriate yards in the rail network, called *classification yards*. In general, a classification yard is equipped with a set of parallel tracks, called *classification tracks*. When a block is built, at least one classification track must be assigned to the block. It is possible to allot more than one classification track to the same block in order to increase the block capacity. After received in a yard (either from customer stations or other yards), cars are kept in receiving tracks. To enter a block, cars need to be moved from receiving tracks to the proper classification track which is designated to the block. The receiving-to-classification track movement is implemented according to the yard infrastructure. Two types of classification yards commonly present, *flat yard* and *hump yard*. In flat yard, cars are moved by shunters (or switch engines), and in hump yard cars are pushed to the top to a hump and slide to the proper classification track by gravity. At the block destination, the block is broken down and the component cars are either delivered to customer's station by feeder trains, or re-classified to shape another block. Therefore, each car may pass one or several blocks in its transportation journey.

The second level consolidation concentrates at transporting blocks with trains. Each train may take more than one block, and at each intermediate stop some blocks are loaded or unloaded. Two observations emerge: first, it is possible for a block to take only a section of the train between any two stops; second, from its origin to its destination, a block may be taken by a sequence of train sections. After unloaded from a train at an intermediate terminal, the block waits in a transfer track and later be loaded to another train, and this process is called *transfer* (or *swap*).

Other rail operations are related to the use of the rolling stock and the schedule of crew. Because of the high capital expenditure of locomotives, a major concern to every railway is to maximize the use of available engines. The locomotive assignment allots power engines to cover all planned trains while satisfying some constraints such as traction power requirements, compatibility restrictions and maintenance constraints. Crew is also necessary for operating trains, and an effective crew schedule should match the proposed trains, respecting the employ regulations, and maximize the working efficiency.

While looking into the railway transportation, another major issue one should not avoid is delay, as a considerable percentage of car-time ascribes to idling either on tracks or in yards. The track lines can be made of a single track, as is often the case in North America and in most developing countries, or may contain two or more tracks, as is common in Europe. To allow trains traveling in different directions on a single-track line to meet, sidings are built at regular intervals along the line. These short track sections allow one train to pull over and free the line for another. Sidings are also used to permit a fast train to pass a slower one. Meeting and passing usually generate delays on sidings. Yard delays derive from various yard operations, mainly from classifications and transfers. The accumulation delay from classification includes the car waiting in receiving tracks for classification, as well as the holding in classification tracks for more cars to be gathered. On average, each classification process results in approximately one day delay for the shipment. Transfers usually bring connection delays where blocks wait in transfer tracks for the next outbound train. In addition, on the same train route, comparing with multi-stop trains, apparently direct trains reduce the delays occurring at the intermediate stops.

The daily operation of freight rail transportation can be briefly summarized as follows. Empty cars are loaded at the customer's station, and transported to the rail yard nearby by feeder trains. The cars then wait for the classification process in receiving tracks. After the classification, cars are formed into blocks. Then the block waits and later taken by an outbound train. On its way to the block destination, the block may be unloaded at some intermediary yard, and transferred to another train after possible connection delay. At the block destination, the block is broken down, and the cars are reclassified and included in another block if they have not reached their destination yard. At the destination yard, the cars are sent to the consignee by feeder trains, and the empty cars are returned to rails later. The empty cars are then repositioned or wait for the next journey.

1.2 Rail Operating Plan

Rail freight transportation presents a complex system, and the performance and profitability of a rail system depend for a large part on efficient and coordinated terminal and long-haul transport operations.

In order to achieve the allocation of resources to activities that fulfills the economic objective and customers' service expedition, railroads must establish an operating (the terms "load" and "transportation" are also used) plan. The operating plan consists of two facets, exterior and interior. Exteriorly, rails propose to potential customers a service plan, which organizes a set of trains to be provided. A *service* represents a train characterized by the origin, destination, train route, speed, capacity, as well as service schedule either expressed in the form of frequency or timetable. Given the complex car-to-block and block-to-train consolidation operations, rails also need to regulate the interior operations. The interior operations support the service plan, and are invisible to customers. The interior operating policies indicate the tactical decision-makings, and answer the question such as, which blocks should be built, which cars are grouped to which block, which train takes which blocks, where the empty cars would be sent, which crew should be assigned to which train, etc. Apparently, the exterior service plan sets the guideline for the interior policies, and the interior policies affect the performance of the services in the service plan.

Rail operating plan concerns the main daily operations in the rail freight transportation industry. Generally, a rail operating plan consists of following policies.

Blocking Policy Blocking policy determines the block building decision, including the car classification policies for each yard. The blocking policy is usually updated with the demand pattern seasonally or with a major change of rail infrastructure.

Train Routing, Scheduling, and Make-up Train routing sets up the routing plan of trains. Scheduling specifies timing information for each possible occurrence of train during a given scheduling length. Scheduling may be indicative, e.g., the frequency is only specified assuming a more or less uniform distribution of departures over the time period. Alternatively, a timetable may be specified for each service indicating arrival and departure times for each terminal on the service route. The train make-up policy gives the assignment of blocks to trains.

Resource Allocation Resource allocation (or asset management) dispatches the necessary resource to each train to support the appropriate service performance. Two major resources, locomotives and crew are generally concerned. Locomotive assignment allocates to each train a fleet of engines to match the power requirement of the train. The locomotives sent to a train may be heterogeneous, and must respect the connection regulations of different type. Crew scheduling dispatches for each train a crew, according to the employee work hour and regulations.

Traffic Distribution Traffic distribution specifies for each particular demand the itinerary used from its origin to its destination: the sequence of blocks and train services, and the corresponding yard operations. As part of traffic routing, *empty car reposition* indicates the routing of empty cars in order to answer the future demands.

We notice that all these issues are interrelated, and particular operating policies in rail freight industry, especially blocking policy for the first consolidation process, and train routing, scheduling, and make-up policy for the second, as well as their interactions make the rail operation plan very complicated.

1.3 Research Problem

Tactical planning for freight rail carriers aims to select the train services to operate over the contemplated schedule length (e.g., week) together with their frequencies or schedules (timetables), the blocks that will make up each train, the blocks to be built in each yard, the routing of the cars loaded with the customers' freight using these services and blocks, and empty car movements. It involves many selections (services, blocks, schedules) and routing decision makings (loaded and empty cars), each with network-wide impacts and all strongly and complexly linked in economic terms and in their time-space dimension. Tactical operation planning is thus a complex problem in most cases.

Many researchers have dedicated to the railway field to develop optimization models and methods to assist rails to operate under an effective, smooth, and cost-efficient way. However, most of the previous efforts were focusing on a specific problem, or only a small portion of the rail transportation, or make significant simplifying hypotheses, as surveyed by [2, 31, 34, 92]. Generally speaking, the industry solution of rail operating plan is obtained by solving the two consolidation process separately, and on top of that, cars (both loaded and empty) are distributed and resources (e.g. crew and locomotives) are assigned.

To detail the relationship among the tactical planning issues, one needs to address this problem in a comprehensive way, and it requires an integrated model. However, no model currently available in the literature addresses in an integrated formulation all the tactical planning issues. Our goal is to answer this challenge and present an integrated tactical planning tool for rail freight transportation.

In this research, we assume the rail network characteristics and information for the facilities and infrastructure are available, and the transportation demand can be obtained from demand forecasting. Most origin and destination stations are on secondary lines. Even when this is not the case, and the station is located on a main trunk line, the movement of loaded and empty cars between stations and their respective designated yards is generally performed by feeder trains whose scheduling is usually not within the scope of the tactical planning process designing the long-haul service network. We follow this practice in this research and assume that all demands are specified at the appropriate origin and destination yards. Furthermore, focusing our study on the analysis of two consolidation process, we suppose the resource allocation policies are determined in a later time. That is, for all the train services provided, there is enough crew and locomotives to support the train movement.

By integrating the blocking policy, train routing, scheduling, and make-up policy and traffic distribution, our objective is to minimize the total operating cost, while meeting the transportation demands from various customers as well the empty demands from rails. Other constraints should be considered come from the rail operations, including the restrictions on number of trains running on each track, limitations on capacities on trains and blocks provided, constraints on number of cars to be handled in each yard, and number of blocks to be built in each yard, etc.

The tactical planning process producing a transportation plan is generally known as the *service network design problem* [34]. The corresponding service network design problem aims to making the most efficient use of the railroad's assets to achieve its economic and customer-service performance objectives. In this research, we integrate the major tactical planning issues in rails, and analyze the trade-offs among them by study a service network design model. The major difficulties we are facing to come from two aspects. First, how to describe the complex operations of rails. To address the various operations on trains, blocks, and cars, as well as their time-related decisions, we are obliged to develop a new network structure. Second, such service network design problems usually lead to very complicated formulations, which present the complication in both mathematical complexity and application scale.

To address the problem, we construct a special multi-layer time-space network structure. Analyzing both temporal and special aspects of operations, a time-space structure can be a useful tool for scheduling the operations in a network. Furthermore, multi-layer structure is incorporated to address the specific operations on trains, blocks, and cars respectively.

To solve such a problem which is complex and large-scale, rather than the exact algorithms which guarantee the optimal solution, we are particular interested in the heuristic/meta-heuristic solutions to achieve a balance between the solution qualities and computing efforts. In this research, several heuristics algorithms are elaborately developed according to different versions of the model.

1.4 Thesis Outline

The rest of the thesis is organized as follows.

The general network design is introduced in Chapter 2, where we first discuss the classification of network design problems, and present two traditional formulations. It is followed by a review of solution methods, with both exact and approximation approaches. Chapter 3 presents a general introduction of tactical planning and service network design in the freight transportation area. Some modeling methods applied in the transportation network design are also presented, together with a selective literature review. In Chapter 4, previous research in tactical planning of rail freight transportation is reviewed.

Chapter 5 proposes a new integrated model for the service network design in rail freight transportation. A complex 3-layer time-space structure is constructed in order to describe the various operations in rails, as well as their temporal information.

In Chapter 6, we study several heuristic algorithms. A tabu search solution, a slope scaling based algorithm, and a hybrid heuristic are developed, according to the characteristics of the simplified model (with only non-stop service), the complete model (with both non-stop and multi-stop services), and very-large-scale complete model.

Chapter 7 presents the numerical experiments. The heuristic algorithms proposed are compared with the state-of-the-art mathematical solver, and the computational results demonstrate the robust performance of our algorithms.

In Chapter 8, we summarize the research results, and conclude by discussing several interesting issues deserving further investigation and research.

CHAPTER 2

NETWORK DESIGN

Expressed in very general terms, the network design problem can be briefly stated as follows. We have a set of *nodes* (or *vertices*) $\mathcal{N}$ which, according to the specific context, may represent cities in a transportation network, switch centers in a computer or telecommunication network, etc. These nodes communicate with one another, exchanging, for instance, commodities of various types, telephone traffic, data traffic, electrical power or whatsoever. To that aim, a network has to be designed (or improved) which typically consists in building (or adding) *links* between some properly chosen pairs of nodes. These links describe resources to carry communication. Depending on the context of application, links could represent urban roads or highways in transportation networks, transmission facilities in telecommunication networks, electric lines in energy systems, etc. The links between nodes may be arcs (in the directed case) or edges (in the undirected case) of a graph. Various characteristics could be associated to the links in the network, and in general we have capacity and cost. Capacity regulates the resources on the link, and cost is the price one pays for installing or adding more resource on a link, as well as the price for routing flows. We use (i, j) to denote an arc (an edge if no orientation is addressed) connecting two nodes i and j. Let $\mathcal{A}$ be the set of all arcs (or edges) in a graph, the associated graph is $\mathcal{G} = (\mathcal{N}, \mathcal{A})$.

Besides the structural representation provided by the graph, a network design problem is also provided with information for the level of communication to be established between nodes. This information has long been identified as *flow requirements* or *demands*.

With all these information, network design is the selection of arcs (edges) in the graph to satisfy the flow requirements, under the constraints of the species and amount of resources. Examples of such constraints are flow following prescribed traffic requirements, or achieving a desired level of security, or satisfying specific technical restrictions, e.g. compatibility of various types of equipments. The objective is generally to minimize the total cost, including both cost for routing flows and cost for installing resources.

This chapter is organized as follows. Section 2.1 introduces the classifications of network design problems. Section 2.2 presents two kinds of general formulations, with flow variables basing on links and paths respectively. In Section 2.3, we review some solution efforts, with both exact and approximate approaches.

2.1 Network Design Problems

Network design problems could be categorized by many criteria. In this section, we introduce basic concepts and variants of classifications of network design problems.

2.1.1 Commodity and Demand

In general, a product or data to be moved in the network is described by a *commodity* with a single origin and destination. A commodity therefore is associated with an origin-destination (O-D) pair. In some applications of network design, a commodity may be shipped from several origins to fulfill the demand of several destinations. For example, the models where the supply is from several origins to satisfy a given demand are often used in the study of the distribution of raw materials. It is worth to notice that one may view the commodity of same type with different origins/destinations as different commodities. For example, for the demand of shipping coals from several origins to one destination, one could consider sending several commodities each from a different origin to the destination. Therefore, demands in the network can always be treated as multiple commodities each with an O-D pair.

Denote a commodity (or traffic) as p. Some previous researches are based on the flow of one single demand with commodity type $m(p)$ and amount (or volume) $w(p)$, which flows from its origin (source) $o(p) \in \mathcal{N}$ to its destination (target) $d(p) \in \mathcal{N}$. However, in numerous situations, the network is designed in order to allow communications between many pairs of nodes concurrently. Thus, one has to consider a number of individual (*single-commodity*) flows, collected in set $\mathcal{P}$. In the case where $|\mathcal{P}|$ distinct commodities have to be simultaneously routed through the network so as to share the resource on links, we have what is commonly referred a *multicommodity* problem.

2.1.2 Capacity

In most practical cases, we have parameters in the network specifying the amount of resource (e.g. number of communication facilities) available (or to be installed) on links or nodes in the network. We focus on the link resource as node resource can be translated by disaggregating each node with two conceptual vertices, and considering the resource on the artificial link connecting them.

The amount of resource existing on a link (i, j) in the network is characterized by the *capacity* of the link, denoted by u_{ij}. Apparently, it is assumed that the capacities are expressed in the same unit (or weight) as the flow requirements. One observes, some problems impose capacities on many individual resources, which can be modeled by multiple links between i and j with corresponding capacities $u_{ij}^1, u_{ij}^2, \cdots, u_{ij}^{|\mathcal{R}|}$, each for a type of resource $r \in \mathcal{R}$. A *capacitated* network design model describes the resource capacity on links, possibly including the constraint of resource on each link, or partial capacity for some resources, or both constraints.

When we consider a network design problem in which u_{ij} is at least the largest possible flow on link (i, j), that means the total flow on link (i, j) always respects its capacity. This case corresponds to the *uncapacitated* network design.

2.1.3 Flow Cost and Design Cost

Costs may be associated to some or all links in the network, including *flow cost* and *design cost*.

Flowing commodities on a link (i, j) generates a flow cost c_{ij}^p which is in most cases related to the type and volume of commodity p on the link. The flow cost is generally *linear* with the flow on the link, however, *nonlinear* flow cost is sometimes used to model flow congestion effects.

To depict the resource installation, the design cost, f_{ij} is normally associated to each link (i, j) to model the cost of constructing new facilities, or the cost of adding capacity to existing facilities. The design cost is generally described by a piece-wise cost function, modeling the unit of resource to be installed on the link. An interesting variant occurs when we restrict the design cost to a *fixed cost* (or *fixed charge*), where only one unit of resource can be associated to each link. In that case, links are either open (with one unit capacity) or close (with zero capacity) in the final design.

2.1.4 Static / Dynamic

The multicommodity problem introduced above can be viewed as a static case, and it should be carefully distinguished from another situation where a set of single-commodity demands present, but the network is designed to meet these requirements in different time. With non-simultaneous flow, one needs to explicitly consider the additional temporal factors and constraints, e.g., time delay cost in the objective, time-dependent resource availability on links. Such cases where commodities flow at different time, or the network presents a different form during different time, are referred as *dynamic* (or *time-dependent*) network design.

To model such non-simultaneous network flow, a time dimension is usually attached to the physical network to shape a *time-space* network, in which every physical node is duplicated at each time in the time horizon. Apparently, the time-space structure expends the network dramatically and makes the problem much larger.

2.1.5 Mono-Layer / Multi-Layer Network

As dynamic network design usually expands the network “horizontally” with a time dimension, one may further consider another expansion of the network by “vertically” delaminating the network into layers.

Multi-layer network design usually derives from telecommunication, where data are wrapped into packages and sent with different devices, e.g. optical fibers for bits and routers for packets. A multi-layer network is constructed with several parallel layers, and each layer carries the flow of commodities with a corresponding format. Demands are routed through layers until reaching the final destination. Focusing on one layer, the demand of commodity (with appropriate commodity format) is determined if one fixes the flows in other layers. In this case, each layer represents a general (mono-layer) network design.

Relative capacities representing the devices for different formats of the commodity are installed in each layer. To describe the resources serving different formats, design costs are usually associated in each layer, and the problem turns into a *multi-level* network design. Additional complexity is usually introduced with the interactions between flow costs and design costs in multiple layers.

2.1.6 Deterministic / Stochastic

Most of the network design models are deterministic where parameters (demands, capacities, costs, etc.) are either estimate-based, or from historical data, or from future predictions.

To describe the uncertainties in realistic, sometimes stochastic is introduced to the network design to address the probabilistic information, mainly on the undetermined demand expectations, uncertain costs, possibility of delays, etc. Stochastic version of the network design problem can be decomposed into several discrete stages, also called *scenarios*, where different probabilities on parameters are associated. The probabilities assigned to scenarios are deterministic and subjective to the application. They also are called weights reflecting the relative importance in an uncertain environment. In general, a stochastic solution is unnecessarily optimal for any individual scenario, but suitably balanced against various scenarios.

2.2 Network Design Models

Beyond the apparent diversity of practical applications involved, most of network design problems can be presented by a rather limited number of basic models.

Generally, a network design formulation has two types of variables, design variables and flow variables.

Design variables address the resource decision on links. Many models contribute to the fixed-cost version where the design variables y_{ij} are restricted to $\{0, 1\}$. $y_{ij} = 1$ only if link (i, j) is *open*, that is, link (i, j) is selected in the final network or for capacity expansion; the link is *closed* when $y_{ij} = 0$. If we extend the design variable definition to,

$$y_{ij} \geq 0, \text{integer}, \qquad (i, j) \in \mathcal{A}$$

meaning, for all the $(i, j) \in \mathcal{A}$, y_{ij} represents the number of facilities or units of resource, or the level of service offered. Nevertheless, as shown by [51], the variant with integer variables can be reformulated with $\{0, 1\}$ variables by variable disaggregation.

Other variables x^p_{ij} are continuous flow variables indicating the amount of flow of traffic p using link (i, j). Thus $0 \leq x^p_{ij} \leq w(p)$. A variant is to use x^p_{ij} to represent the percentage of demand volume $w(p)$ on link (i, j), $0 \leq x^p_{ij} \leq 1$. This variant is especially suitable for uncapacitated network design where each demand can be intuitively normalized to one unit. Some applications may require integer value for flow variables, for example $x^p_{ij} \in \{0, 1\}$ to regulate the non-fractional flow of each demand. Most methodological developments, however, have been dedicated to network design formulations with continuous flow variables.

It is also possible to define flow variables on each path instead of on each link. Let $\mathcal{L}^p$ be the set of paths for commodity p, connecting $o(p)$ to $d(p)$. We define h^p_l the flow of traffic p on path $l \in \mathcal{L}^p$. A parameter δ^{lp}_{ij} is associated, $\delta^{lp}_{ij} = 1$ if link (i, j) belongs to path $l \in \mathcal{L}^p$, and $\delta^{lp}_{ij} = 0$ otherwise. Thus we have,

$$x^p_{ij} = \sum_{l \in \mathcal{L}^p} h^p_l \delta^{lp}_{ij}. \tag{2.1}$$

Two mostly applied network design formulations are introduced as follows, which differ from the type of flow variables employed. The general link-based formulation will be discussed in detail to explain the different classes of network design problems, and an equivalent path-based formulation is briefly introduced later.

2.2.1 Link-based Formulation

The link-based *Fixed-charge Multicommodity Capacitated Network Design* (FM-CND) formulation is most commonly found in the literatures. A case with continuous flow variables on flow volume can be illustrated as follows.

$$\min \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{(i,j)\in\mathcal{A}}\sum_{p\in\mathcal{P}} c_{ij}^p x_{ij}^p \tag{2.2}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}} x_{ij}^p - \sum_{j\in\mathcal{N}} x_{ji}^p = w_i^p \qquad i\in\mathcal{N}, p\in\mathcal{P}; \tag{2.3}$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \le u_{ij}y_{ij} \qquad (i,j)\in\mathcal{A}; \tag{2.4}$$

$$(y,x)\in\phi \qquad (i,j)\in\mathcal{A}, p\in\mathcal{P}; \tag{2.5}$$

$$y_{ij}\in\{0,1\} \qquad (i,j)\in\mathcal{A}; \tag{2.6}$$

$$x_{ij}^p \ge 0 \qquad (i,j)\in\mathcal{A}, p\in\mathcal{P}. \tag{2.7}$$

The objective function as shown in (2.2) measures the total cost in the network, which includes the fixed costs for offering the resource (or opening the links), and flow costs of moving commodities (transport products).

The *flow conservation* constraint (2.3) expresses the usual demand satisfaction requirements imposed on each traffic p at each node $i\in\mathcal{N}$. w_i^p is the absolute demand for traffic p at node i,

$$w_i^p = \begin{cases} w(p) & \text{if node } i \text{ is the origin of demand } p,\ i=o(p); \\ -w(p) & \text{if node } i \text{ is the destination of demand } p,\ i=d(p); \\ 0 & \text{otherwise.} \end{cases}$$

Constraint (2.4), called *linking* (or *forcing*) constraint, states that the total flow on link (i,j) cannot exceed its capacity u_{ij} if the link is chosen in the design of the network ($y_{ij}=1$) and must be 0 if link (i,j) is not part of the selected network ($y_{ij}=0$). There are some other forms of linking constraint. An example with multiple link capacities corresponding to each commodity is shown in (2.8).

$$x_{ij}^p \leq u_{ij}^p y_{ij} \qquad (i,j) \in \mathcal{A}, p \in \mathcal{P} \tag{2.8}$$

where u_{ij}^p is the non-negative *partial capacity* restricting the amount of flow of p moving through link (i,j). One observes, if it is assumed that $u_{ij}^p \geq min(u_{ij}, w(p))$, (2.8) becomes redundant for the network design formulation, in that case, the inequality is denoted *strong linking inequality*. Formulation with additional (2.8) is also called *strong formulation*, comparing with the version otherwise which is denoted *weak formulation*. Another special form of linking constraints exists when we consider the network design problem in an *undirected* graph. Note that even if the graph is undirected, flows are directed, in which case, the edge capacity is shared between every commodity flowing through the edge regardless of the direction. The linking constraint in an undirected network is presented as,

$$\sum_{p \in \mathcal{P}} \left(x_{ij}^p + x_{ji}^p \right) \leq u_{ij} y_{ij} \qquad (i,j) \in \mathcal{A}.$$

Additional constraints related to the design of the network or relationships among variables are gathered up in (2.5). These constraints might model, for example, topological restrictions imposed upon the configuration of the network, or side constraints restricting the resources shared by several links, etc. An important type of such constraints reflects the financial limitation which applies in many cases.

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

The constraint states that the maximum expenditure for constructing the network is limited by a total budget **B**.

2.2.2 Path-based Formulation

The *Path-based Fixed-charge Multicommodity Capacitated Network Design* (P-FMCND) formulation is with flow variables basing on each path. The path-based formulation is equivalent to the link-based model according to equation (2.1). An example of path-based formulation is expressed below, where percentage flow variables are engaged.

$$\min \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{p\in\mathcal{P}}\sum_{l\in\mathcal{L}^p} k_l^p h_l^p \tag{2.9}$$

$$\text{s.t.} \quad \sum_{l\in\mathcal{L}^p} h_l^p = 1 \qquad p \in \mathcal{P}; \tag{2.10}$$

$$\sum_{p\in\mathcal{P}}\sum_{l\in\mathcal{L}^p} h_l^p \delta_{ij}^{lp} w(p) \leq u_{ij}y_{ij} \qquad (i,j)\in\mathcal{A}; \tag{2.11}$$

$$y_{ij} \in \{0,1\}, \quad (i,j)\in\mathcal{A}; \tag{2.12}$$

$$0 \leq h_l^p \leq 1 \qquad l \in \mathcal{L}^p, p \in \mathcal{P}. \tag{2.13}$$

In the objective, k_l^p is the flow cost for moving $w(p)$ unit of traffic p on path l,

$$k_l^p = \sum_{(i,j)\in\mathcal{A}} c_{ij}^p \delta_{ij}^{lp} w(p).$$

Constraint (2.10) is the flow conservation constraint corresponding to (2.3), and the linking constraint (2.11) is related to (2.4). Constraint (2.5) of FMCND does not exist in the P-FMCND formulation. It is usually addressed during the path generation process.

Several important problem classes may be derived from the general network design by the appropriate definition of the network $\mathcal{G}$. When fixed costs are associated to nodes, one obtains *network location* formulations. *Traveling salesman problem* which consists of finding a minimum cost cycle that traverses through each node exactly once is formed by proper setting of constraint (2.5). Different sets of constraints on the form of the optimal network design yield the *steiner* and the *spanning tree* problems. And the *vehicle routing problem* may be viewed as a special case of the spanning tree formulation. These major extensions further remark the significance of network design.

For more detail of the fixed-charge network design models, interested readers can refer to Crainic [34].

2.3 Solution Methods

The network design problem has long been recognized as one of the most difficult and challenging problems in combinatorial optimization. The generic fixed-charge network design models introduced in Section 2.1 is with non-concave objective, and is known as NP-hard [85]. The difficulties of solving this problem exist in the following ways. First, the trade-offs between fixed design costs and flow costs conduct the link selection in the network. Selecting more links gives opportunities for reducing the flow costs at the expense of higher fixed costs. On the other hand, with fewer links in the design, the fixed costs are lower but the routing costs increase. Second, in the capacitated version commodities compete for the limited resource on links. Furthermore, the problem size grows very quickly with the increase of the nodes, links, and commodities, especially when the time-space network or multi-layer network is concerned, which leads the real-sized applications extremely large.

With the mixed integer programming (MIP) formulations, network design problem may be approached by the methodologies available for this class of problems.

One intuitive method to solve network design problem is to restrict the convex hull of the problem by adding so-called *valid inequalities* (or *cuts*). Valid inequalities are in the form of linear constraint, which might be redundant for the network design formulation but not for its linear programming (LP) relaxation. For example, the strong linking inequality (2.8) presents a cut for the network design formulation. Apparently, if one can construct the convex hull of the problem by these cuts, the original problem could be solved by linear programming methods. Various cuts are implemented within the framework of *cutting plane* method, in which, the formulation is tightened by a succession of resolutions of the LP relaxation of the problem and cut generations.

Benders decomposition [56] is another approach used to solve the problem. The structure of network design problem presents a natural decomposition scheme for the

Benders approach. The fundamental idea behind this method is to decompose the formulation into two simpler subproblems. The first subproblem, called *master problem*, is generally associated with the design variables which present a tentative configuration of the network. The second part, called *auxiliary problem*, obtains the values corresponding to the multicommodity flow problem while keeping the design variables fixed, and generates cuts for the master problem. Therefore, the master problem gives a network pattern under which the auxiliary problem finds the optimal distribution of commodities. The master and auxiliary problems are solved iteratively, until no more cuts can be generated.

Implicit enumeration scheme or *branch-and-bound*, are widely applied in solving network design problems. Branch-and-bound extracts the network design problem within a branch-and-bound tree and each tree node represents a subproblem where some variables are restricted. Relaxation approaches have been applied to obtain the lower bound of the original problem. Lagrangian relaxations and LP relaxation are the two mostly applied approaches. In the Lagrangian relaxation, multipliers are adjusted by non-differentiable optimization techniques, for example, *subgradient* or *bundle* method. For the LP relaxation, *dual ascent* is often used to improve the lower bound. Dual ascent generates lower bounds by solving the LP dual problem approximately. Then the solution of the LP dual is strengthened by iteratively updating the dual variable in the dual problem. Other methods are also imposed to improve the lower bound. *Branch-and-cut* is a branch-and-bound algorithm in which cutting planes are generated throughout the branch-and-bound tree. Rather than re-optimizing fast at each tree-node, the branch-and-cut algorithm adds as much cuts as necessary to get a tight bound at each tree-node.

Given the NP-hard complexity of most network design problems, researchers have often used approximate (heuristic, meta-heuristic) algorithms as alternative solution methods for solving large-scale network design applications, or to give upper bounds in the branch-and-bound algorithm. Three commonly utilized heuristics are *add*, *delete*, and *interchange* procedures. The add heuristic starts with some feasible design and add links, one at a time, choosing at each step the link that gives the greatest decrease in cost. The delete heuristic is similar, but starts with an initial design containing all candidate links, and removes links one at a time. Starting with some initial design, the interchange heuristics add and/or delete a link at each step until no further improvement in cost is possible.

In the following, we review and analyze various solutions applied to the network design problem by the type of network design models.

2.3.1 Uncapacitated Network Design

The uncapacitated version of network design problem, with infinite capacity on links, has been quite well studied in several research reports.

Magnanti and Wong [86] applied Benders decomposition for the fixed-charge uncapacitated network design. A special progressing technique eliminating non-optimal variables is combined with acceleration techniques. This implementation of Benders decomposition has solved undirected network problems to optimality with up to 30 nodes and 90 edges.

Balakrishnan et al. [13] proposed an efficient dual ascent method for fixed-charge network design models with complete demand patterns. Initial from a feasible solution generated by the dual ascent algorithm, the traditional add/delete heuristic is used for further improvements. Computational results confirmed that on problem instances with up to 45 nodes and 595 edges, the approach provides good lower bounds within 1% to 4% of optimality.

In Holmberg and Hellstrand [66], a Lagrangian heuristic within the branch-and-bound scheme has been developed. The Lagrangian heuristic is based on the Lagrangian relaxation of flow conservation constraints. The Lagrangian dual is solved by the subgradient method from initial heuristic solutions. The performance of the approach depends on the setting of many parameters, and modifications might be required to tune the method according to a certain problem structure. However, on a given parameter setting, a larger problem with 40 nodes, $1,000$ arcs, and 600 commodities has been solved to optimality. With respect to problem size and solution time, this method outperforms the state-of-the-art mathematical solver at the time.

Sun et al. [112] developed a tabu search heuristic with simplex-based local search.

The procedure is guided by several searching memories, including immediate memory to find the local approximated optima, short-term memory to intensify the optimal search, and long-term memory to diversify the search among the least visited arcs. The algorithm is tested on some dense instances with $50,000$ demands.

For the uncapacitated version, the efficient solution methods exist partially because the good characteristic of the LP relaxation of the uncapacitated network design. As shown by Hellstrand et al. [64], the polytope of the LP relaxation of uncapacitated network design problem is *quasi-integral*. That is, any edge of the network design (integer) polytope is an edge of the LP relaxation polytope. However, such property is unlikely to hold for the capacitated problem.

2.3.2 Capacitated Network Design

While in some cases the network can be assumed to be uncapacitated, capacitated models are more general and often much more suitable for real-life applications. However, the limited link capacities make the problem much more complex.

Valid inequalities are generated to tighten up the formulation of the problem for the development of efficient solution methods. Other than the strong inequality (2.8), several well-known *cutset* inequalities derive from particular structures of the network. These inequalities are based on the idea that in any feasible network solution, the capacity of the links of any partition (cutset) that cuts off some origins from their destinations must be sufficient to carry the demands across the partition. Such inequalities include cover cuts, minimum cardinality cuts, flow cover cuts, and flow pack cuts, which are explained by [12, 20, 28, 111] in detail.

Stallaert [110] developed a cutting plane procedure for the fixed-charge capacitated network design. Properties of the fractional extreme points of the LP relaxation are used to construct a class of inequalities. Chouman et al. [29, 30] adapted to capacitated network design several important families of valid inequalities, and developed a cutting-plane algorithm. Efficient separation heuristics and lifting procedures are used in conjunction with the cut generation algorithm. These efforts improve the lower bounds through the identification of new valid inequalities. However, solving the resulting LP relaxation requires considerable computational efforts since the formulation is even larger because of the importation of valid inequalities.

Benders decomposition methods, based on the idea of partition and constraint generation, have been successfully applied to uncapacitated network design problems. However, only a few efforts exist for the capacitated network design, and are for the specially defined cases (as reviewed by [32]). Sridhar and Park [109] incorporated Benders cuts with some cutset cuts, and indicated that Benders cuts are more effective for the problems with heavy traffic demands. Rei et al. [103] applied local branching cuts [48] to accelerate the classical Benders decomposition algorithm. Costa et al. [33] analyzed Benders cuts, cutset inequalities and metric cuts (a special case of Benders cut) for the network design problem. The authors have shown that cutset inequalities are a subclass of Benders inequalities, but are not necessarily metric inequalities. Benders decomposition method is also applied by Fortz and Poss [50] to solve a 2-level network design problem.

Branch-and-bound algorithms are the most common tools to solve capacitated network design problems. An efficient branch-and-bound algorithm requires improving the lower bounds provided by relaxations. Gendron and Crainic [52] compared several relaxations (LP relaxation, Lagrangian relaxation on linking constraints, Lagrangian relaxation on flow conservation constraints) on both weak and strong formulations. The authors showed that both Lagrangian relaxations on the strong formulation yield the same theoretical lower bound. The strong LP relaxation also generates the same bound because the subproblem has the integrality property. On the weak formulation, Lagrangian relaxation on linking constraints outperforms LP relaxation and Lagrangian relaxation on flow conservation constraints. Nevertheless, experiments concluded that the gaps shown by the upper bounds provided by a resource-based decomposition are still large, especially when with high fixed costs on links.

Linear programming relaxation, especially the weak relaxation, as shown by previous researchers, generally does not provide good approximations to the capacitated network design problem. In order to obtain tight bounds, valid inequalities are usually incorporated, and branch-and-cut algorithms are developed. For example, Günlük [62] presented a branch-and-cut method for the capacitated network design problem, where not only valid inequalities but also branching strategies are studied. The proposed algorithm shows a competitive performance on small instances, and beats the classical branch-and-bound algorithm on larger ones.

Compared with the LP relaxation, Lagrangian relaxation approaches sound more promising. However, the two Lagrangian-based relaxations have pros-and-cons respectively. While relaxing the linking constraints, the subproblem can be decomposed into a number of shortest path problems. When the flow conservation constraints are relaxed with Lagrangian multipliers, it yields a series of very simple and separable knapsack problems; however, the network structure is vanished.

Gendron and Crainic [54] proposed a bounding procedure based on the Lagrangian relaxation on the linking constraints. The lower bound is derived by optimizing the Lagrangian dual with the traditional subgradient method. The computational effort required by the procedures is significantly reduced by designing parallel implementations that exploit a decomposition by commodity [53]. The authors also showed that the implementation and calibration of Lagrangian-based methods have a significant impact on the behavior and performance.

Holmberg and Yuan [67] proposed a Lagrangian approach by relaxing the flow conservation constraints, and solved the Lagrangian dual by the subgradient method. The solution method also includes techniques for finding primal feasible solutions to serve as the upper bound. The Lagrangian approach is then embedded into a branch-and-bound scheme for further improvements. To speed up the branch-and-bound procedure, a heuristic variable fixing method is used for choosing promising tree nodes, by the information from the Lagrangian dual solution. While variable fixing is applied, the method turns into a heuristic. Computational results showed that the method is efficient in generating near-optimal solutions.

In the process of solving the Lagrangian dual problem, researchers noticed that the traditional subgradient method performs unevenly, sometime shows a zigzag behavior or even worse, and it may stop far from the optimal solution. Crainic et al. [43] calibrated and compared bundle and subgradient methods applied to the optimization of Lagrangian duals arising from two Lagrangian relaxations (on linking constraints and flow conservation constraints respectively) of the capacitated network design. The authors showed the fact that the bundle method appears superior to subgradient approaches because they converge faster and are more robust relative to different relaxations, problem characteristics, and the selection of initial parameter values. The study also demonstrated that good lower bounds may be computed efficiently for large-scale instances of the network design problem.

Another way to improve the Lagrangian relaxation is to tighten the solution space by adding valid inequalities. This is commonly realized by appending the *dualized cuts* into the formulation to preserve the structure of Lagrangian subproblem. Sellmann et al. [107] presented a branch-and-bound approach. The authors focused on local tightening strategies, such as variable fixing and local cuts which are only valid for the current sub-tree in the branch-and-bound tree. Different variable fixing strategies are evaluated solitarily and in combined versions. This approach seems promising, but the success exists in trading tighter solution space with more difficulties in solving the Lagrangian dual problem because a large number of constraints are dualized. Kliewer and Timajev [80] also integrated Lagrangian-based branch-and-bound with valid inequalities. Two types of valid inequalities, the cover inequalities and local cuts, are carefully incorporated into Lagrangian relaxation, maintaining the structure of the subproblems. Experiments indicated that the incorporation of valid inequalities improves the overall performance significantly, and the proposed *relax-and-cut* algorithm outperforms many other exact and heuristic methods in terms of running time and solution quality.

Due to the extreme complexity of the capacitated network design problem, heuristics and meta-heuristics have been applied to balance the quality of the final solution generated and the computational effort required. Unlike the exact solutions which guarantee the ascertainment of the optimality, heuristic solutions aim at locating near-optimal solutions in rather short solution time. The solution schemes such as Benders decomposition, branch-and-bound introduced above can also be used as a heuristic, so long as the process stops when the predefined gap between upper and lower bound is reached.

Kim and Pardalos [77] proposed a slope scaling approach for solving the capacitated network design problem by adapting an economic viewpoint to the fixed costs. *Slope scaling* is an iterative scheme that consists in solving a linear approximation of the original formulation at each iteration. The costs of each linear approximation are adjusted in order to reflect the total cost incurred by the solution at the previous iteration. The iterations proceed until two successive solutions are identical. Computational results on a wide range of problems with more than 200 nodes and $10,000$ arcs are reported. The results demonstrated that the proposed procedure generates solutions within 0.65% of optimality on the tested instances. Crainic et al. [44] improved the performance of the slope scaling by importing two features: Lagrangian perturbation and long-term memory perturbation. After the slope scaling process, the dual information from some relaxation of the original problem is used to modify the approximation factors. An intensification/diversification mechanism based on long-term memories is also adopted to escape from local optimum. Computational experiments proved that the intensification/diversification component of the algorithm is essential for the effectiveness, which also demonstrate that the algorithm is competitive with the best known heuristics for the problem, especially on the larger and more difficult instances. Kim et al. [79] also incorporated slope scaling with additional search memories, including short-term memory for intensification and long-term memory for diversification. The searching memories are integrated within a tabu search strategy. Similar with the slope scaling, a capacity scaling idea has been proposed by Katayama et al. [73]. Instead of approximating the fixed cost in each iteration, the authors adjusted the surrogate capacity on each link to scale the contribution from the design cost components. A path-based formulation is studied, and the column and raw generation integrated enables the method to address large problems, however, only results on small instances are reported.

Crainic et al. [42] presented an efficient tabu search heuristic for large-sized network design problems. The algorithm is based on the path-based formulation. Tabu search moves are defined in two neighborhoods: Simplex pivot-type moves on a continuous neighborhood for local search which is guided by column generation, and diversification moves on a discrete neighborhood. The reported results show that the procedure is robust with respect to the parameter selected, and it finds good solutions on large instances which cannot be addressed by the mathematical solvers at the time. Zaleta and Socarrás [114] have later adopted the idea to the undirected version of the network design problem.

Ghamlouche et al. [57] proposed a cycle-based neighborhood for meta-heuristics aiming at the fixed-charge capacitated network design. The neighborhood defines moves that may explicitly consider the impact on the total design cost of potential modifications to the flow distribution of several commodities. The fundamental idea is to explore the space of the design variables by rerouting flows around cycles. Thus, compared to link-based neighborhoods, not only the move evaluations are more comprehensive given all commodities on a cycle are explicitly considered, but also the range of cycle moves is broader because flow deviations are no longer restricted to paths connecting origins and destinations of commodities. A tabu search is developed to evaluate the quality of the proposed neighborhood. Experiments on instances of various sizes (up to 700 arcs and 400 commodities) showed that the powerful performance of the cycle-based neighborhood. The solution algorithm is later strengthened by the *path relinking* method [58]. In the implementation, tabu search provides feasible solutions and contributes to a reference solution set, path relinking then takes two reference solutions from the reference set, and tries to find a "linking" to connect these two solutions in the cycle-based neighborhood. Path relinking phase stops when the reference set has been exploited, then either stopping criteria are verified or the procedure repeats to build a new reference set with tabu search. Several strategies for building reference set and selection of reference solutions are compared. Extensive experiments indicated that the path relinking procedure appears more robust than tabu search in terms of both solution quality and computational effort.

Alvarez et al. [6] studied a capacitated network design problem on an undirected network. Scatter search algorithms are developed and compared, and computational results showed that the relative gap between the optimal solution and scatter search solution in a range varying from 0.4% to 1% on instances with up to 60 edges. The algorithm is strengthened by generating a population of solutions with greedy search heuristic as starting points to the scatter search procedure [7].

Recently, some researchers tried to combine heuristic ideas with exact mathematical solvers (or MIP solvers, usually provided by CPLEX), in order to take advantage from both sides: the robustness of heuristic for finding promising solution domains, and efficiency of MIP solvers for exploring the local domain. Lewis [83] implemented experimental designs to reduce the solution space by identifying and setting critical variables prior to running a MIP solver. The solution gives better results than the MIP solver alone. Hewitt et al. [65] studied both link-based and path-based formulations, where path-based formulation selects variables for the link-based formulation, and elite problems with elaborately selected variables are tackled by MIP solvers. Rodríguez-Martín and Salazar-González [104] applied the local branching idea in order to partition the solution space with local branching cuts, and used MIP solvers to explore each local domain. The same local branching idea has been applied by Fischetti et al. [49] to solve the multi-level network design problem. These methods generally outperform the existing meta-heuristics and produce high-quality solutions on most of the benchmark instances with reasonable computational efforts.

Parallel computation may save the computational time for the realistically dimensioned problem instances. To speed up computation, computational workload is divided and dispatched to several processors. One example comes from Gendron and Crainic [53] who applied parallel computation in the branch-and-bound tree to solve the restricted problem at each node. In the case of heuristic, parallelism may also improve the quality of the solution. Crainic and Gendreau [37] presented a cooperative parallel tabu search method for the path-based formulation. Several communication strategies are analyzed and compared. The experiments demonstrated that parallel implementations find better solutions than sequential ones. Crainic et al. [45] described a first multi-level parallel search algorithm on the cycle-based tabu search for the capacitated network design problem. The method appeared competitive, particularly on difficult problems where a large number of commodities are considered. A general review and analysis of parallel heuristics can be found in Crainic and Toulouse [40], and see Crainic [35] for a taxonomy of parallel approaches for tabu search.

2.4 Summary

Network design is a central problem in combinatorial optimization. The general formulations not only model a variety of network design issues, but also represent a number of related network models. Two commonly applied formulations (with flow variables basing on link and path respectively) are reviewed.

Although easy to describe, network design problems are very difficult to solve. Historically, uncapacitated network design models are well studied. However, for the capacitated version, exact and efficient solution methods generally do not exist, except for special variants of the problem formulation. Branch-and-bound is the most common way to solve this complicated problem. Upper bound presented by a feasible solution is normally given by a heuristic. Relaxation methods, especially Lagrangian relaxations are used to generate the lower bound of the original formulation, however, solving the Lagrangian dual problems can be a hard work.

Because the problem is NP-hard, approximate algorithms are often preferred to exact algorithms for solving large-size instances occurring in practice. Meta-heuristics respectively based on the cycle-based and path-based neighborhoods have been presented, and proved to be efficient for constructing good feasible solutions. Math-heuristics such as slope scaling with guiding memories are also investigated. Parallel and distributed computations offer another interesting perspective to overcome real-life dimension instances within a reasonable solution time.

Recently, some efforts are contributed to the combination of valid inequalities with Lagrangian relaxation, as well as MIP solvers with heuristic ideas to give better solutions. While interesting results are reported, it is noted that successful solutions of this difficult problem may exist in the integration of many different approaches developed in history. Specifically, the sophisticated combination of valid inequalities, heuristics, and MIP solvers implies a promising avenue.

More discussion of modeling and algorithmic issues of the network design problem is surveyed by Gendron et al. [55], Balakrishnan et al. [14], and Minoux [88].

CHAPTER 3

SERVICE NETWORK DESIGN IN FREIGHT TRANSPORTATION

Distributing passengers and all kinds of goods and materials, transportation is essential to the health of the economy. Transportation is implemented by carriers (e.g. airlines, shipping lines, less-than truckload (LTL), railways) who operate planes, ships, trucks, trains, or any combination of above methods, and provide transportation services. Some services are dynamic determined (e.g. taxi), however, to serve the customers' demands, nowadays many carriers set up a service plan to regulate the services to be provided.

The application of network design can be found in many fields, and service network design in transportation represents one of the major categories. Service network design considered in this chapter derives from the freight transportation industry, and is particular relevant to the companies or organizations operating consolidation transportation systems, in which case, services are shared by a broad number of customers.

In the rest of this chapter, we briefly introduce the general freight transportation system and its hierarchical structure, and then address the service network design problem. After that, some selective examples of service network design in transportation are reviewed to illustrate previous researches.

3.1 Freight Transportation System

Freight transportation undertakes the work of carrying commodities from their origins to their destinations in a transportation network. A transportation network consists of all kinds of terminals (e.g. station, port) and physical links (e.g. road, rail tracks) or conceptual links (e.g. air course, sea route).

Customers present transportation demands from their origin to the destination. These demands are generally associated with an O-D pair, and characterized by the weight, type, and dimensions (as in LTL) or unit of cargos (as containers in maritime shipping or cars in railway). Accompany with the demands, customers usually propose requirements for the shipping time, either regarding to the maximal transportation time from the origin or the due date at the destination.

Carriers provide *services* to answer the transportation demands. A service consists of one or a covey of vehicles to carry the shipments from one terminal to another. For example, air service with planes, express delivery service with trucks, and rail service with trains. According to various criteria, we have different types of services.

Main-Line and Feeder Services Regarding to the service scope, we have main-line services and feeder services. Main-line services represent long-haul (e.g. intercity or interstate) transportation between major terminals, and feeder services provide local (or urban) transportation between a major terminal and the secondary terminals nearby, as well as between local secondary terminals. For example, when a container is shipped from America to Europe with intermodal services, it is usually picked up by a feeder service (by trucks or barges) to the nearby port, and shipped by a big containership (main-line service) to cross the Atlantic. In some industries, e.g. airlines and railways, both main-line long-haul services and feeder local services are provided, and hub-and-spoke distribution paradigms usually emerge in such transportation networks.

Customized and Consolidation-based Services Freight services can be divided into *customized* (or dedicated) services and *consolidation-based* services. For the customized transportation, the full truck-load (FTL) door-to-door service offers a typical example, where each demand is carried by one or a fleet of vehicles which are dedicated to this customer, that is, each service is tailored for a demand. Customized services can be dynamic as dial-a-delivery (like dial-a-ride in passenger) transportation, or prescheduled according to long-term contracts with VIP customers who present significant amount of demands regularly. Instead of offering exclusive service for each demand, many carriers (e.g. railway, airline) combine the demands from different customers and move the freights with possibly different origins and destinations by some common services. The progress that service providers group commodities from customers is called *consolidation*. In order to consolidate shipments from different customers, different cargoes and vehicles are grouped, or simply moved from one service to another at the *consolidation terminal* (e.g. rail classification yard, LTL breakbulk) in the transportation network. On the way of its journey, a shipment may pass one or several consolidation terminals before reaching its destination.

Multi-Stop and Non-Stop Services During the journey of a service, there might be some intermediate stops, where some cargos are loaded to or unloaded from the service. Customized services for important customers are generally non-stop, which is from the origin directly to the destination. On the contrary, consolidation-based long-haul transportation presents multi-stop services in many cases, with exceptional non-stop services given there is enough amount of demands between two end-of-line terminals.

A transportation system (or transportation plan) regulates various services and operations in the transportation process. The most important component of a transportation system is the service decision. In general, service decisions are determined in several stages: customized services are usually more profitable and planned first, and the consolidation-based services are studied on top of it. Nevertheless, the consolidation-based transportation contributes to a considerable portion of the business for carriers operating on a large transportation network, and the associated operations are more complicated. In the following, we focus our interests on this case.

In the consolidation-based transportation, carriers must propose regular services, which are generally specified by the origin, destination, intermediate stops, and service route, in order to satisfy the customers' demands. Before the departure of a service, there is usually an *assembly* process where cargos are loaded, e.g. cars are joint to make up trains, containers are loaded on ships. At the destination terminal, convoys are dismantled (or disassembled). Other characteristics are usually associated to a service, such as speed or priority. Speed is generally related to the service schedule, which gives the exact departure time at the origin, the arrival/departure time at the intermediary stops, and the arrival time at the destination. Rather than schedule, sometimes frequency is used to indicate the number of services provided in a certain planning horizon, and priority is usually adopted to address the precedence of each service.

In order to ensure the proper performance of the proposed services, it's necessary for carriers to establish a series of working rules or policies. The consolidation policies present the major decision-makings regarding the issues of which and how cargos are attached to each service. For example, the blocking policy in rails and loading plan in shipping lines. The consolidation policy regulates the traffic grouping operations for carriers, and may brings additional temporal restrictions to customers, such as cut-off time (the latest moment shipments are given to the carrier in order to catch a service). Some other components also appear in a freight transportation system, e.g. empty flow and resource allocation. Due to the imbalance that exists in trade flows, empty cars, trucks, and vehicles must be moved (repositioned) in order to respect the demand in the future. It is worth to notice that even the flow in the network is equilibrated in the long run, it is unnecessary to be balanced in a short term. The problem of how many and where to send each kind of empties (empty cars, empty containers, etc) to be reused appears in many industries (e.g. rail, LTL). Another notion often encountered in transportation system has to do with asset allocation, e.g. how many consolidation equipments should be assigned to each terminal in order to carry the consolidation workload, which power unit and crew should be assigned to carry which service, etc.

Concerning various decisions restricted by a lot of human and material resources, transportation systems are rather complex. See Crainic [36] for a more general presentation of transportation systems.

3.2 Tactical Planning

According to the complex transportation system, transportation planning problems emerges in different corners. Problems facing to transportation planners can be grouped into several classes according to the facet of the organization concerned.

Strategic Level : Strategic decisions determine the general development policies and broad operating strategies of the system, where one is mainly concerned with the construction of infrastructures and acquisition of durable devices that will remain active over a long period of time. Examples of decisions at this planning level are the design and evolution of the transportation network. Other than the firm scale, strategic planning also takes place at the regional or national range, where the transportation networks or services of several carriers are simultaneously considered.

Tactical Level : Tactical problems are related to medium and short term issues that generally involve the specifications of operating policies updated every few months. In this level, decisions are sensitive only to significant variations in parameters. For example, when the new terminals or routes are introduced to the transportation network, or demands change seasonally. Tactical decisions need to be made mainly concerning the design of the service to be provided, general operating rules for each terminal, as well as workload distribution among terminals. The general guideline for empty reposition is also studied in this level.

Operational Level Operational level issues concern the delicate detail of the system. Problems in this level are considered by local management (e.g. steering coordinators or port dispatchers) in a dynamic environment. Yard cargo management, maintenance activities, crews reposition form the main part of the problems in this level.

This hierarchical structure presents how the information flows among the decision-making levels. From the top strategic level to the bottom operational level, each level sets the general policies and guideline for the decisions taken at the next level. On the contrary, lower level decisions supply the essential information for the decision-making process at the higher level. This popular classification is explained in greater detail by Assad [11], who also gave examples of problems that belong to each category.

The tactical planning aims to determine, over a medium-term horizon, an efficient allocation and utilization of resources to achieve the best possible performance of the whole system. Tactical planning handles the major decision-makings for the general operations, and has a great impact to the working efficiency and profitability of transportation service providers. According to Crainic and Laporte [38], the main decisions made through the tactical planning should concern 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 terminal, intermediate stops and physical route. Priority (with frequency) or speed (with schedule) decisions are also indicated 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, etc.

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 congestions.

Some industrial solutions are obtained by solving these planning problems individually, and the final plan is the aggregation of the optimal solutions of each problem. Such efforts include [61] for express shipping, [70] for intermodal transpiration, and [69] for railways. This approach is unnecessarily optimal in a mathematical sense but it allows carriers/planners to easily address the above problems which are specific to the different mode of transportation.

3.3 Service Network Design

Service network design models are typically developed to assist the tactical planning for the transportation industry. The idea is to maximize the profit by setting routes and schedules/frequencies, with respect to various resource constraints. For example, airline companies must determine the air service network and frequency of the flights considering aircrafts/vehicles and crew availability. Similarly, express package delivery companies establish routes, assign aircraft to them and decide about the routing of packages.

Service network design is vital to transportation providers. A proper service network design can yield better service quality and cost reductions. The total amount of these reductions is closely related to each specific problem, however, the economical importance of most of the transportation problems and the key role played by the network design suggest that the savings can be significant. One good example is from Santa Fe Railway, one of the largest railway companies in North America, who applied a whole solution tool for service network design in 1998, and the results reveal a potential for 4% cost savings over the current railroad operating plan, coupled with 6% reduction in late service [59, 60].

Service network design specifies the services to be offered, to satisfy the transportation demands and ensures the profitability of the company. To maximize the profitability, carriers would like to minimize the total operating/transportation cost, which in general matches the customers' interests. However, nowadays more and more customers require not only low tariffs, but also high service quality which is normally expressed in terms of service speed, reliability and flexibility. How to achieve the best trade-off between operating cost and service performance is the main objective of service network design. Therefore, the relation between one hand in service quality and the other hand in transportation cost has to be shown in service network design models.

To select the best solution for the carrier and customers, one has to simultaneously consider the routing of all traffics, level of service on each route and service characteristics, as well as operations in each terminal. That is, most of the issues in the tactical planning. Service network design thus describes a system-wide view of the transportation, including the operations performed in different facilities and the resource requirements which are usually conflicting.

Many efforts have been directed toward various aspects of the service network design in freight transportation, and applications can be found in many contexts, for example, in LTL [e.g. 27, 47, 97, 100, 105], express shipment [e.g. 10, 15, 18, 61, 78], rail [e.g. 17, 41, 72, 93], and multimodal transportation [e.g. 39, 94]. Most service network design models yield multicommodity capacitated network design formulations. Base on the fashion dealing with the temporal aspect, service network design can be categorized into *service network design with frequency* and *service network design with schedule*, based on the static and dynamic formulations respectively. The two variants are illustrated below, together with some examples in the literatures.

3.3.1 Service Network Design with Frequency

Frequency service network design concerns the tactical issues such as: what type of service to offer, how often over the planning horizon to offer it, which traffic itineraries/routes to operate, what are the appropriate terminal workloads and policies, etc. A possible result from such frequency service network design models should like: "Carrier provides 4 services from terminal Montreal to terminal Toronto in 2 days (planning horizon), two of them are express services, and the other two are regular services with an intermediary stop at terminal Kingston".

The frequency service network design model is suitable for situations where demands vary considerably, and/or the actual timing of the demand is unknown. Frequency service network design model may be further classified into *decision* and *output* according to how the formulation generates the frequency for each service. In the first case, service frequencies are explicit integer decision variables, and in the second case, service decision variables are binary, and frequencies are derived from traffic flow.

A good example of service network design with frequency decisions comes from Crainic and Rousseau [39]. The authors presented a general modeling and algorithmic framework for the tactical planning of freight transportation with explicit frequency variable on each possible service route. The model has a nonlinear multicommodity multimodal network design formulation, which considers the cost for providing service, cost of distributing freight, and cost of delay. The solution method of the general model combines a heuristic and decomposition scheme which works alternately on the service level and traffic distribution subproblem. Special applications of the above framework in rail and LTL are developed respectively by [41, 105, 106]. Crainic et al. [41] studied the classification/blocking operations in rails. Experiments of rail applications are from Canadian National Railway, one of the largest transportation corporations in Canada, and the results on historical data suggest a saving of 3% to 4% in total operating cost. Roy and Delorme [106] described a structure of the LTL model. The applications of this tactical planning model on several trucking companies are reported by Roy and Crainic [105].

For the output frequency service network design, a fixed-charge network design formulation is used to address the decision of providing services on the designated route or not. Then the frequency of the service is obtained as the output of the traffic flow variable subject to the restrictions on service level.

Powell and Sheffi [101] proposed a network optimization model to solve the load plan problem for motor carriers. The design problem is formulated as a large-scale mixed-integer model, and some heuristics are developed to determine how to consolidate flows of shipments over the network. Powell [97] later extended the model, and improved the solution method. In the new model, the frequency is derived from the service level constraint, which is represented heuristically through a set of minimum frequencies on links. The method decomposes the problem into one master problem and several subproblems by the hierarchical structure of the system, with an add/delete local improvement heuristic to solve the master problem. The presented algorithm is tested on the data from a large motor carrier with over 300 terminals. Powell and Sheffi [102] further improved the previous work by developing an interactive optimization system that allows users to deal with hard-to-quantify constraints.

Another service network design application exists in express shipment delivery transportation, where shipments are transported from origins to destinations on daily basis, and timing constraints are stringent. Kuby and Gray [81] examined the effectiveness of hub-and-spoke networks with stopovers and feeders, and compared their performance to direct flights into a hub. Two kinds of variables: package flow decision variables and aircraft routing decision variables are used. However, the authors assumed that only one sorting hub exists and the network is small. Kim et al. [78] focused on a multi-hub express shipment problem. A mixed-integer formulation is presented, with the objective function minimizing the costs of deploying transportation assets to form the service network, and costs of transportation in order to respect the demands. A heuristic approach is employed, in which routes are generated first, and then shipment movements are computed. Barnhart et al. [18] extended the work from [78], and the solution is improved by iterating between selecting routes and moving shipments. Armacost et al. [10] transformed the formulation of [78] to a new composite variable formulation for the service network design problem. The authors showed that the LP relaxation of the new formulation gives a better lower bound than conventional approaches. Barnhart and Shen [15] extended the previous work to integrate the service network design for premium and deferred express shipment delivery. New solution approaches based on column generation are developed.

3.3.2 Service Network Design with Schedule

The service network design with schedule, also called *dynamic* service network design, targets at the planning of schedules and support decisions related to issues such as if we offer the service or not and when services depart. A corresponding solution might look like this: " One service is provided from terminal Montreal to terminal Toronto every day, the service departs from Montreal at 10am, and arrives at the intermediary terminal Kingston at 1pm. Loading/uploading process takes 2 hours at Kingston, and service continues from Kingston and arrives at Toronto at 6pm".

To address the schedule of the services, a time dimension is generally introduced to the network structure to construct a time-space network. This is usually achieved by representing the operations in the system over a certain number of *time periods*.

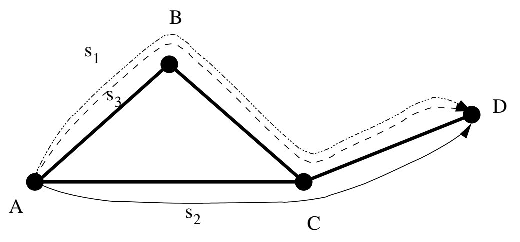

Figure 3.1: Illustration of Physical Network

A typical time-space network structure has been presented by Farvolden and Powell [47]. On the physical network (shown in Figure 3.1), three services are designed, where s_1, s_3 are by the way of $(A \rightarrow B \rightarrow C \rightarrow D)$, and s_2 is via $(A \rightarrow C \rightarrow D)$. The time-space network corresponding to this physical network is illustrated in Figure 3.2.

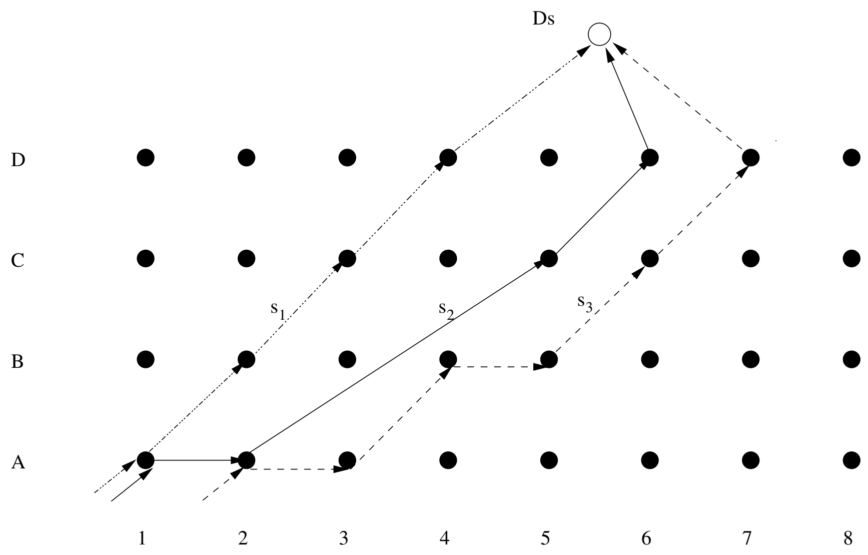

Figure 3.2: Illustration of Time-Space Network

Two sets of nodes and three sets of links are defined in the time-space network. The first set of nodes is *terminal node*, created by replicating each terminal in the physical network at each time in the time horizon. The second set of nodes called *super node*, each corresponds to a terminal in the physical network, serves as the ultimate destination of all flows which end at the terminal (only supper node D_s is shown in Figure 3.2). Each terminal node is connected to the terminal node in the next time of the same terminal by an *inventory link*. Flows on these links represent the waiting process in terminals. *Service links*, connecting two terminal nodes of different terminals, represent shipments being transported between terminals. Finally, the *super links* which connect the one terminal node to the associated super node represent the shipment-specific penalty which depends on the arrival time of each shipment.

Comparing these two figures, one notices that the routes corresponding to the three services in the time-space network not only characterize the routine of the services, but also address the schedule of the services. However, the time-space network has a so much larger and more complicated structure than the physical network.

On the time-space network, Farvolden and Powell [47] described a scheduled service network design problem for LTL transportation. The objective, which is to minimize the total shipment routing and travel costs of tractors over the time-space network, is addressed with two sets of decision variables, the shipment routing decisions and the vehicle dispatching decisions. Penalty cost is applied to penalize deliveries before/after specified due dates. However, the empty balancing and consolidation operations were not described in the model. An efficient primal-partitioning algorithm is proposed, where column generation algorithm is used to solve the freight routing problem, and service configuration is explored by the add/delete heuristic.

Haghani [63] constructed a special time-space network to describe the railway operations. Each terminal is represented by two vertices at each time point, and the terminal operations are explicitly addressed by the links connecting the vertices. The inter-terminal links thus show traffic movements of both loaded and empty cars, and intra-terminal links represent consolidation, connection, and delays for consolidation. A large scale nonlinear mixed-integer formulation is proposed based on the network structure and the model is solved by a heuristic with decomposition technique.

3.4 Summary

Tactical planning arises from airlines, truck companies, railroads, etc., wherever there is a need to determine minimum cost services and operations, given constraints on resource availability and level of services.

Service network design concerns most of the tactical planning issues in the transportation system, and can be a very useful tool to aid the decision-making process for planning operations and services. Most service network design models yield a multicommodity capacitated network design formulation, which currently has no efficient solution. Moreover, when applied into each application field, researchers are facing to additional complexity introduced to account for the particularities of the application, which usually makes the problem even harder.

Crainic and Laporte [38] and Crainic [34] presented recent reviews on service network design in transportation.

CHAPTER 4

RAIL FREIGHT OPERATION PLANNING

Aiming at establishing a plan of operations to achieve the goals of profitability and quality of services, rail operation planning is vital to rail carriers. On one hand, a good operating plan improves the working efficiency and decreases the operating costs; on the other hand, the rigorous competitions between rail companies as well as the pressure arising from other transportation industry (e.g. truck) urge the good application of operating plan for providing reliable and flexible services.

An intuitive way to construct a rail operating plan is to decompose the operating plan according to the working procedure. A blocking model is first made to indicate the routing of freight and the distribution of classification work among the yards in the network, and then train routing and make-up models are studied to determine the routing and frequency of trains and the assignment of blocks to trains. The scheduling model performs on the result of the routing and make-up model to specify the timetable of services. After, the resource allocation model assigns enough locomotive power and crew to support the train movement. Empty balancing model concerning about the reposition of empty cars then enforces the performance of the railroads. This procedure is applied by Ireland et al. [69], who presented a whole solution tool for Canadian Pacific Railway. These tools use operations research approaches, such as an optimal block sequencing algorithm, a heuristic algorithm for block design, simulation, and time-space network algorithms for planning locomotive usage and distributing empty cars.

There are many models exist for rail freight transportation, either addressing an individual issues, or several issues in a combination. We present in this chapter an analysis of major publications on rail tactical planning issues, to understand the state of recent research. However, resource management is only briefly introduced since they are not concerned in the following research.

4.1 Blocking Policy

Blocking policy is developed for a medium-term (tactical) planning horizon, and is updated as traffic conditions and customer needs change significantly. In general, the blocking policy is generated by solving a blocking problem, in order to choose which blocks to build at each yard and to assign sequences of blocks to deliver each shipment, with an objective of minimizing the operating cost which may include total transportation, handling, and delay cost, etc.

Bodin et al. [22] developed a nonlinear mixed-integer programming model for the railroad blocking problem. The model is based on a blocking network, which is a network on the same node set of terminals but with arcs representing potential blocks rather than physical tracks. The authors used a piece-wise objective to model the shipping, handling, and delay cost. They also included constraints on the minimum and maximum block load, maximum number of blocks, and so called *pure strategy* constraints, which assume that all commodities traveling between two terminals should follow the same blocking path. Because the resulting model is too large to be handled by the computational tools available at the time, the authors manually fixed some integer variables, and the best solution found was within 3% of tight lower bound.

Newton et al. [93] extended the aforementioned blocking model, and considered the different priority classes of traffic. A network formulation is proposed on the blocking network. No fixed costs were associated to blocks, but several capacity restrictions were considered to limit the number of blocks and the total volume of freight processed at each yard. A solution approach [16] is developed to find good solutions for the link-based multicommodity network flow model. In this so called *branch-and-price* algorithm, a column generation algorithm is used to solve the LP relaxation throughout the branch-and-bound tree. Numerical experiments indicated that the proposed method is efficient, and for the instance with 150 nodes, $6,800$ possible blocks, and $1,300$ commodities, the algorithm found feasible solution within 0.4% optimal in a couple of hours.

Barnhart et al. [17] used the same formulation as [93], and applied a dual-based Lagrangian relaxation approach to solve the problem. Unlike the previous branch-and-price approach, the linking constraints are relaxed in a Lagrangian fashion, and then the Lagrangian relaxation is decomposed into two disjoint subproblems: a traffic flow subproblem and a block building subproblem. In this manner, the approach greatly reduces the memory requirement and computational effort for large-scale problems. To speed up the convergence of the subgradient optimization, a dual ascent approach is developed to give the initial setting of Lagrangian multipliers. The model is tested on the blocking problem from a major railroad in the United States, and the result showed the blocking plan generated can significantly reduce the current rail operating cost.

Another extension of the above blocking model is studied by Ahuja et al. [4]. A special neighborhood search algorithm is developed to heuristically improve the current blocking policy. The algorithm is tested on the data provided by three major railroads in North America, and the computational results confirmed the efficiency and its potentiality to be applied to rails.

4.2 Train Routing, Scheduling, and Make-up

Given the blocking policy which indicates the distribution of blocks, the train routing problem determines the routing and frequency of trains. Some train routing models are based on the existing blocking policy, and the make-up policy, which indicates the allocation of blocks to trains, usually be combined in the routing model. However, there are some routing models that work with the car flow directly, and either ignore the blocking policy in the system or consider the blocking policy implicitly, in which cases, an explicit train make-up model is necessary to distribute blocks on trains.

Assad [11] provided one of the earliest contributions to the train routing and make-up problem, and a multicommdoity network flow formulation is adopted. Marín and Salmerón [87] studied the train routing problem, and a service network design model is developed in which a train service is defined by its origin, destination, a set of stops, speed, and capacity in terms of cars. Considering the constraints on the number of cars transported on each track segment, number of cars processed at each yard, and train numbers, the model aims to minimize the sum of fixed cost for each train, handling and delay cost of cars, and installation cost for additional trains. To solve this problem, the authors first decomposed the model into two subproblems: the routing for freight cars and the grouping of cars to trains. Three heuristic methods (descent method, simulated annealing, tabu search) are proposed, and a branch-and-bound algorithm is presented for a modified linear formulation.

With the efficient routing of trains and freight produced by routing models, it is necessary to synchronize the use of the available tracks/yards since they are shared by many trains on the physical network. The timetable of trains is addressed by scheduling models based on the train routing model in which the temporal information is usually roughly described by the queuing method. To attach the temporal dimension to a given routing plan, a large number of scheduling models [e.g. 3, 24–26, 113] has been developed to coordinate the moving of train and other corresponding resources (e.g. locomotives). In many cases, the given routing plan provides a good base for the train scheduling; however, sometimes the congestion in the train schedule may suggest a major modification of the routing plan. Some compound models attempting to integrate train routing and scheduling these two intertwined problems have been presented.

Morlok and Peterson [89] are probably the first to integrate train routing and scheduling in a single optimization model. Considering the questions of the route and intermediate stops of the train, departure time of the train, cars per train and speed of the train, a very large mixed-integer formulation has been developed. The objective of this model is to minimize the total cost, including engine and crew cost, intermediate yard cost, car-time cost and cost of additional horsepower per car. Four groups of variables are used in the model. The first group is the binary decision variable for train services, which is defined by a route in the network, a set of stops, a departure time at the initial yard, speed and capacity. The second type represents the cars per train, and another two groups determine the total car time used by the various train scheduling procedures. Instances in somewhat small rails are solved with a branch-and-bound procedure. However, this model did not analyze the yard operations, neither break down freight cars by types.

Huntley et al. [68] developed a computerized routing and scheduling system to help planners at CSX Transportation account for the effects of train decisions. In this model, demands are represented as origin-destination batches. When constructing a plan from a set of batches, the objective is to minimize the operating cost without any car missing connections with trains. In this model, each terminal-to-terminal link in the routes is assumed to follow a preset path. Considering the routing and scheduling function simultaneously, a demand-driven approach is developed.

Newman and Yano [91] proposed a model within the context of scheduling direct and multi-stop trains, and assigning containers for each train for the rail portion of the intermodal transportation. The objective is to achieve on-time delivery with minimum cost, including a fixed charge for each train, variable transportation and handling cost for each container, and yard storage cost. Three kinds of integer variables indicating the number of containers held in inventory, number of containers shipped on tracks, and number of trains are used. The problem is modeled as a piecewise-concave-cost multicommodity network flow problem, and both centralized and decentralized approaches are presented to construct feasible solutions.

With the established blocking policy and service plan, some researchers isolated the train make-up problem, and considered the block-to-train assignment. Nozick and Morlok [94] addressed the rail movement of freight within the context of rail-truck intermodal transportation given a fixed train schedule over a medium-term horizon. The model focused on the operations that are usually within the control of railroads, including the line-haul and yard operations. Taking equipment and locomotive repositioning into account, the objective is to minimize the cost of transportation such that traffic movements are feasible with respect to equipment availability, and on-time delivery of shipments. A heuristic is developed for solving this model. Kwon et al. [82] formulated a multicommodity flow problem to determine the traffic routing, and to improve a given blocking plan and block-to-train assignment. The model is represented on a time-space network structure, and the formulation takes into account the train capacity restrictions, with the objective to find an assignment to minimize the late delivery penalties. Column generation is applied to solve the linear model, and the model is tested on a hypothetical rail network. Jha et al. [71] also looked into the block-to-train assignment problem. The problem is modeled on a time-space network as well with a multicommodity network flow formulation. Both Lagrangian heuristic algorithm and greedy construction heuristic are developed to obtain feasible solutions within a short computing time.

4.3 Empty Car Reposition

The empty car management problem consists of distributing empty cars in the rail network to improve the railroad's ability to promptly answer requests for empty cars while minimizing the costs associated with their movement. The empty car reposition problem is either studied in the operational level or with the service planning in the tactical level. The first approach is to plan the empty flows on top of a given service plan, and dynamically distribute empty cars with the residual capacity on trains. The second way of considering this problem is to predict the demand and supply of empty cars, and then view the empties as commodities in the tactical planning. In this sense, empty flow is treated as a part of demand flow, and empty equipment balancing is incorporated into the traffic routing problem.

As reviewed by Dejax and Crainic [46], some early formulations are to minimize the total empty car-time cost over a certain periods, subject to the given imbalance of cars of a homogeneous fleet. Such efforts include Powell and Carvalho [98, 99], who determined the number of flat cars to be sent from one terminal to another by formulating a logistics queuing network. Recent research extends the area to dynamic, stochastic and with heterogeneous fleet. Sherali and Suharko [108] developed a tactical decision-making system with two empty car repositioning formulations. The first formulation considered uncertainties in transit times, priorities with respect to time and demand locations, and multiple objectives related to minimizing different degrees of lateness in delivery. The second model further integrated the consideration of blocking policy. Heuristics are used to solve the models. Bojović [23] tried to balance the cost of unmet demands and the sum of car ownership and utilization costs, and a model basing on optimal control theory is developed.

Joborn et al. [72] presented an optimization model for empty car distribution in a scheduled railway system. The authors first defined *kernel paths*, representing movements of cars with identical origin and destination. The empty distribution network thus can be transformed into a time-space network with fixed cost on kernel paths. A tabu search heuristic is developed to solve the model. The solutions obtained from the model display the economy-of-scale effect, in the sense that the empty car movements in the solution are coordinated to reduce the number of car clusters which is a group of cars transported together on the network.

Another optimization model for empty car management was applied by Narisetty et al. [90] in Union Pacific Railroad. The problem formulation targets at finding the best possible matches between available empty cars and customer demands, and also compatible with the current operating plan, namely the residual capacity of the scheduled service plan. The desirable match is evaluated by transportation costs, car-substitution costs, early/late delivery penalties, customer priority, and service efficiency. Around 10% improvement is reported in their final results.

Bektas et al. [19] studied the problem of dynamic reassignments of empty cars to outbound trains at a given yard, in order to minimize the total dwell time of the cars processed at the yard. During the reassignment process, both blocking policy and train schedule must be respected. The approach is tested on data from Canadian National Railway, and the experiments showed an around 5% time saving with the proposed procedure.

4.4 Resource Allocation

The term resource (or asset) in this section is specially donated to the horsepower and manpower that required for providing train services. Most of the previous researches were focusing on two major resources, locomotives and crew. Despite the proper assignment, an asset may be repositioned between terminals if it ends a train in one location and is operating its next train from a different location.

Traditionally, in many instances, asset management issues were not directly included into tactical planning process. It is generally implicitly assumed, however, that assets are available when needed, and consequently, their management, assignment, and repositioning is left to be dealt with at the operational level of planning on an already-decided service network [e.g. 3, 5].

Recently, some efforts have been made to combine the asset management to the service network design. Pedersen and Crainic [95] imported vehicle management to intermodal transportation with train canals. Pedersen et al. [96] generalized the model with asset positioning and utilization by addressing constraints on asset availability at terminals, which are denoted as design-balanced constraints. However, only one single type and unit of asset (i.e. engine) is required to perform a service. Several formulations are compared and a tabu search heuristic is proposed to solve the model. Andersen et al. [8, 9] extended the model from [96] to describe multi-asset management and coordination considerations, as well as the interactions between rail services being designed and services in collaborating transportation systems.

4.5 Integrated Models

Apparently, the sequential solution of planning issues obtained through the previous aggregation method might not be necessarily optimal, due to the lack of the consideration of trade-offs among service decisions and block decisions, as well as their schedule. Some researchers therefore focus on the study of developing approaches of integrating several aspects of rail operations and analysis of a number of activities simultaneously.

In the work of Crainic et al. [41], traffic class is first defined as an O-D pair, together with a commodity type. A set of *itineraries* for each traffic class is also defined, each to specify a feasible journey of this traffic class. Itinerary includes the train service path followed and the operations (classification or transfer) performed at intermediate stops. By selecting the best traffic distribution for each traffic class, one not only solves the traffic routing problem but also determines the blocking and train make-up strategies as well as the distribution of classification work between yards. Two decision variables are used, which are continuous volume variable of each traffic class traveling on its itineraries, and integer frequency variable associated with possible train services. The objective function is the sum of operation and delay cost of itineraries and train services. The model presents a nonlinear mixed-integer multicommodity formulation. By introducing the train capacity constraints into the objective function, the authors solved the problem by decomposing the problem into a master problem and subproblems one for each traffic class. The master problem modifies the service level based on the given traffic distribution. For each subproblem, column generation is applied to determine the best traffic distribution for each traffic class. The model is solved by placing high frequencies for all services at the beginning, then iterates between the master problem and subproblems until the improvement in the objective function is less than a predefined value. Crainic and Rousseau [39] further presented a general model for multicommodity multimode freight transportation and explained the solution methodology in greater detail.

Haghani [63] attempted to combine train routing, scheduling, make-up and empty car distribution problems. Based on a given physical network, a time-space network is constructed with fixed operating times. In the time horizon considered, links are built representing normal routing, express routing, classification, delays and deliveries. With the decision variables concerning the flow of loaded cars, flow of empty cars, and number of engines provided on links, the model chases the objective to minimize the total cost defined by routing cost, classification cost, delay cost for classification and connection, and penalty cost for late delivery. The model has not only flow conservation constraints on the loaded cars, empty cars and engines, also restrictions for the linkage between loaded and empty cars. The formulation is a large-scale mathematical program with a nonlinear objective function and linear constraints. A heuristic decomposition approach is proposed to solve somewhat simple problems and appears efficient for small rail systems.

Keaton [74] presented a model examining the problem of simultaneously deciding which pairs of terminals are provided with direct train service, and whether to offer more than one train per day, as well as the routing of freight and the blocking of rail cars. The service network was made up of one network for each pair of yards in the system with positive demand. Links represented trains and connections in yards, as well as *a priori* determined blocking alternatives. Two kinds of variables are used in the model, integer variables for train connections and continuous variables for the distribution of traffic. However, blocking was not an explicit decision. The objective is to minimize the total cost which includes train cost, car time costs, and classification costs in yards. The constraints considered in the model include the classification capacity in terms of blocks, train load capacity in terms of cars, as well as the flow conservation constraints. A heuristic method based on Lagrangian relaxation is developed for the model, but it is unable to give good lower bound with the train load limits. When ignoring the train load constraints, the proposed algorithm has been proved to be efficient. If the resulting solution contains some overloaded trains, heuristic adjustments are necessary to restore the feasible operating plan. Keaton [75] worked on the same problem, and more constraints are considered, such as the maximum transit time constraint for each O-D pair. With the assumption that all cars between each O-D pair must follow the same routine (pure strategy), the problem is modeled with two kinds of integer variables. Lagrangian relaxation technique is used to solve the problem. However, it is difficult to evaluate the feasible operating plan produced by the model since no tight lower bound is reported.

Gorman [59] proposed a model aiming at the design of a scheduled operating plan that follows the particular operation rules of a given railroad. Model simplifications have been introduced to achieve a comprehensive mathematical network design formulation, and costs considered are the fixed crew costs of train, marginal cost per unit on a train including fuel and locomotive cost per unit, equipment and locomotive time costs while a train is in transit, and classification cost for each unique demand on a train. However, the delay cost for freight in terminals is not addressed. A hybrid meta-heuristic (tabu-enhanced genetic search) was developed to generate candidate train schedules, which were evaluated on their economic, service, and operational performances. On relatively small but realistic instances, the meta-heuristic performed well and was used for strategic scenario analysis for a major railroad in North American [60].

4.6 Summary

Tactical planning for rail freight transportation is an important and complicated field. One source of complication in rail freight transportation is the complex double consolidation organization, car-to-block and block-to-train grouping procedure. The relations between consolidation processes and their interactions with the rest of the system are the most critical trade-offs in rail freight transportation.

Various models focusing on each part of the rail system have been reviewed. Due to the intuitive transportation network topology, service network design has been widely applied in rail tactical planning. However, most of the previous models focus on one level consolidation, and either basing on the result from, or simply ignoring, or implicitly considering the other consolidation level.

Our review of many previous researches reveals areas where opportunities exist for new optimization development. A good opportunity lies in the improvement of formal methods for linking decisions from different consolidation levels. Another opportunity exists in the combination of operating policies with schedule. Since the traditional approach does not take the scheduling into consideration when making the train routing plan and blocking policy, considerable savings might be generated if we plan and at the same time synchronize the trains services and yard operations.

Integrating blocking policy, train routing, scheduling, and make-up policy and traffic distribution (loaded and empty) into one model may be a promising direction, but also a challenging avenue. Compound models lead to difficulties in both modeling approach and solution method. A more complicated network structure is necessary to fully describe the trade-offs among planning issues. The targeting model always results in a complex mathematical formulation in the form of multicommodity capacitated network design, and needs the development of special tailored solution algorithms.

CHAPTER 5

3-LAYER TIME-SPACE NETWORK STRUCTURE AND MODELS

Rail freight transportation is based on a rail network (or physical network). A physical network can be denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex $v \in \mathcal{V}$ represents a terminal and each link $e \in \mathcal{E}$ stands for a rail track segment. Planning at the tactical level, secondary stations are aggregated and only major yards handling a large number of traffic, and main-line tracks connecting yards are considered. One simple physical network with 4 yards and 4 directed tracks is illustrated in Figure 5.1.

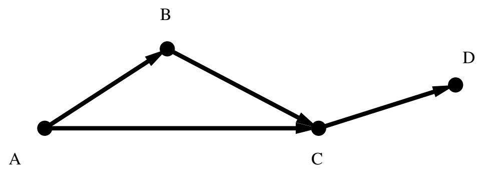

Figure 5.1: A Simple Rail Network

The physical network comes with a number of restrictions on rail operations. The volume of activities a classification yard may perform during any time period is restricted by the layout and physical characteristics of the yard infrastructure, as well as available facilities (e.g. shunter engines). Let u_v^C be the car handling capacity in terms of cars, and u_v^B the number of blocks that may be built at the yard. To simplify the presentation, we assume that a classification track is required for each block being built, and u_v^B is determined by the number of classification tracks of the yard. Furthermore, we assume a rail track $e \in \mathcal{E}$ cannot accommodate more than u_e number of trains at the same time. In general, train running capacity depends on the physical track condition, track mile, the speed and direction of each train, as well as time interval between adjacent departures, etc. Working at the tactical level, however, there is no need to specify the detailed on-track delay and we assume the train running capacity only depends on track condition.

Apparently, the physical network, which often appears in earlier works, is insufficient to picture the temporal information and the flows in different formats. In this chapter, a 3-

layer time-space structure is first presented, to describe the time-dependent operations of rail freight transportation. A general model for integrated rail planning is then proposed with the associated cost and capacity analysis, followed by a discussion of a simplified variant.

5.1 3-Layer Time-Space Network Structure

To picture the temporal information, we associate to the rail network a time dimension with $\mathbf{T}$ time points, denoted by $t \in \{0, \cdots, \mathbf{T}-1\}$. As rails work periodically, the time dimension is defined as *cyclic*, that is, time point $\mathbf{T}-1$ is followed by 0. The time horizon is thus divided into $\mathbf{T}$ time periods.

With the double consolidation organization, in rails, flows present in three formats: services, blocks, and cars. To properly present traffic movements, we further delaminate the time-space structure into three layers, which are denoted *service layer*, *block layer*, and *car layer*, as shown in Figure 5.6. Each layer concerns operations and delays on shipments either in the form of service, block or car. The time dimension is the same in the three layers.

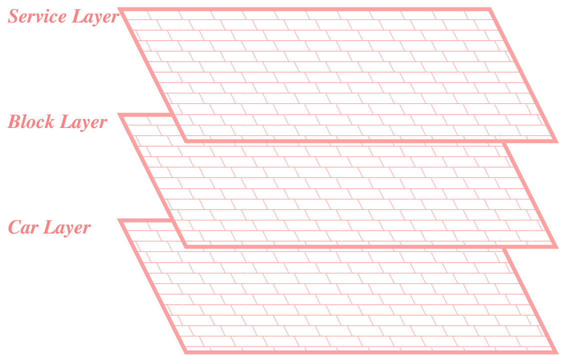

Figure 5.2: 3-Layer Structure

Most consolidation work is carried out in yards. To represent yard operations, we decompose each yard with two nodes at each time point, an IN node and an OUT node. The IN node depicts the receiving of services, blocks or cars in the yard at the time point, and the OUT node represents the departure of corresponding objects. Denote $\mathcal{N}^{\mathrm{S}}$, $\mathcal{N}^{\mathrm{B}}$

and $\mathcal{N}^{\mathrm{C}}$ the sets of nodes in the service, block and car layer, respectively.

Links are defined to represent various operations and delays (by horizontal links in each layer) and formation transforms between cars and blocks, as well as blocks and services (by vertical links connecting layers). Define *temporal length* (or *length*) of a link the total time periods this link covers. Vertical links have length 0.

5.1.1 Service Layer and Service

Links in the service layer represent possible operations on trains, including *moving links* and *stop links*. Some moving links and stop links are shown in Figure 5.3 which derives from the rail network in Figure 5.1. From an OUT node representing the origin yard at departure time to an IN node representing the destination yard at arrival time, a moving link stands for a non-stop train movement on a physical route with a constant travel time. A set of rail tracks, defined as $\mathcal{E}(a)$, is associated to each moving link a to indicate the physical route passed. On a physical route, shorter moving links represent train running on a greater speed, and using lower speed trains causes longer transit time. For example, (i_5, i_6) and (i_5, i_7) on route ($B \to C$). Parallel moving links, with the same origin and destination node as well as the same length, may also exist. Parallel moving links represent train movements following different routes between two yards with the same departure time and transit time. In Figure 5.3, a_1 and a_2 are both from node i_1 to i_2, however, they follow routes ($A \to B \to C$) and ($A \to C$), respectively. Describing train tarry at an intermediate yard, a stop link connects an IN node to the next OUT node of the same yard, as shown (i_4, i_5). Let $\mathcal{A}^{\mathrm{SM}}$ be the set of moving links, $\mathcal{A}^{\mathrm{SR}}$ the set of stop links.

In the service layer, a *service* (denoted by s) is defined as a path consisting of a series of moving links and stop links. At its origin OUT node $o(s)$, the service assembles and departs, and at the destination IN node $d(s)$, the service disassembles. The journey of a service is characterized by the moving links (collected in set $\mathcal{A}^{\mathrm{SM}}(s)$) it passes. For each service, we are therefore aware of its origin, destination, service route, intermediate stops, and schedule. Let $\mathcal{S}$ be the set of all services we may provide.

A *service section* is a sub-journey of a service, and it is represented by a sub-path

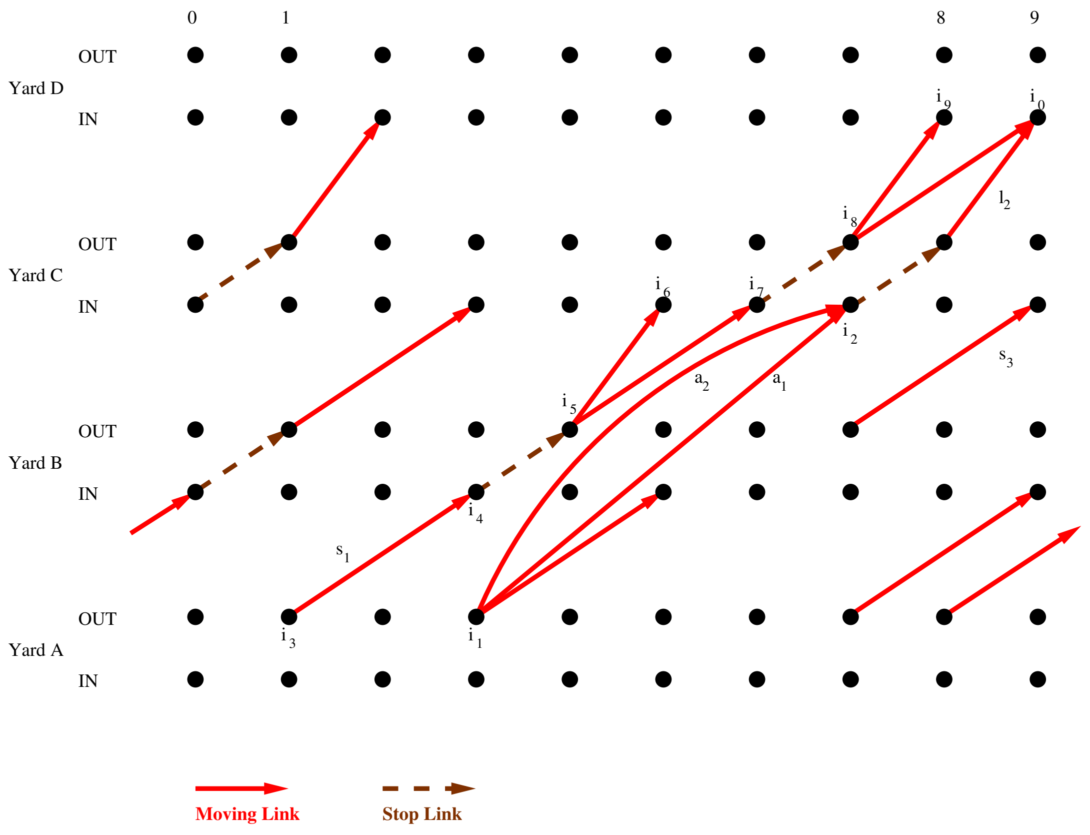

Figure 5.3: Service Layer

between two unnecessarily consecutive stops. For example, service s_1 in Figure 5.3 departs from yard A and arrives at yard D. The service consists of 3 moving links, and includes 6 service sections: $(i_3 \rightarrow i_4)$, $(i_5 \rightarrow i_7)$, $(i_8 \rightarrow i_9)$, $(i_3 \rightarrow i_4 \rightarrow i_5 \rightarrow i_7)$, $(i_5 \rightarrow i_7 \rightarrow i_8 \rightarrow i_9)$, and the service itself. Given a service $s \in \mathcal{S}$, $\mathcal{L}(s)$ is the set of its service sections, and for a service section l, $s(l)$ denotes the service that l belongs to. The journey of service section l is marked by a sub-set of moving links in service $s(l)$, $\mathcal{A}^{\text{SM}}(l) \subseteq \mathcal{A}^{\text{SM}}(s(l))$. The stop links in the service section l is collected in set $\mathcal{A}^{\text{SR}}(l)$.

5.1.2 Block Layer and Block

Links in the block layer represent yard operations on blocks, which are either transferred or delayed for connection. Two types of links, *transfer links* and *transfer delay links* are defined, collected in set $\mathcal{A}^{\text{BT}}$ and $\mathcal{A}^{\text{BH}}$ respectively. As shown in Figure 5.4, a transfer link, e.g. (i_7, i_8), connects an IN node to the next OUT node, representing the service-switching procedure, that is, blocks are disconnected from one service and later connected to another. A transfer delay link attaches two consecutive IN nodes of a yard, e.g. (i_6, i_7), standing for one-period delay for catching the next service.

In order to attach the block layer to the service layer, vertical links are defined to represent loading/unloading blocks to services. Each *load block link* departs from an OUT node in the block layer to its upstairs OUT node in the service layer, representing loading blocks onto services. On the contrary, an *unload block link* is a downward link between two vertical IN nodes, standing for unloading blocks from services.

Regardless of the building process, the journey of a block is defined by a *block path* (or *block*) from an OUT node to an IN node (OUT and IN are from different yards). A block is formed by a series of service sections from different services, which are connected by transfer delay links, transfer links and necessary vertical links. The service sections, transfer links, and transfer delay links in block b are collected in set $\mathcal{L}(b)$, $\mathcal{A}^{\text{BT}}(b)$, and $\mathcal{A}^{\text{BH}}(b)$ respectively. In addition, building a block takes a classification track in the block origin, and the classification track occupancy time for forming block b is denoted by $h(b)$. Let $\mathcal{B}$ be the set of all potential blocks.

If project the service sections in the top layer to the block layer, we have a *block-layer projection* consisting of service sections, transfer links and transfer delay links, as shown in Figure 5.4. $(i_3 \rightarrow i_6 \rightarrow i_7 \rightarrow i_8 \rightarrow i_9)$ corresponds to a block, which is taken by a service section departing from yard A at time point 1, delayed and transferred at yard C, then shipped by another service section (i_8, i_9) and terminates at yard D. $(i_3 \rightarrow i_7 \rightarrow i_8 \rightarrow i_9)$ describes a similar block, which takes service section (i_3, i_7) from yard A to C.

5.1.3 Car Layer

The bottom car layer as shown in Figure 5.5 has *classification links*, *car waiting links*, and *car holding links*, expressing classification and delays on cars. A classification link, e.g. (i_2, i_4), characterizes the classification operation, that is, cars are moved from receiving tracks into classification tracks. Car waiting links, e.g. (i_1, i_2), represent

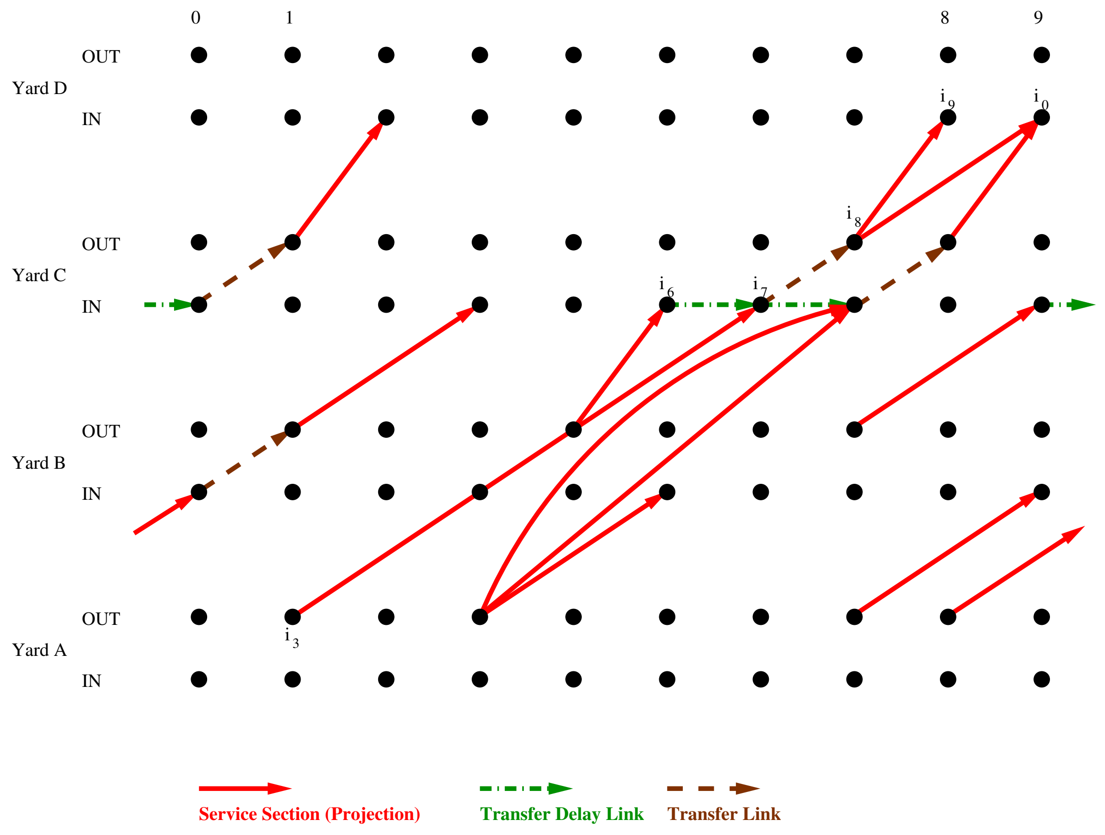

Figure 5.4: Block Layer

delays for classification: cars wait in receiving tracks to be classified. Next, between two neighboring OUT nodes, a car holding link is built to describe one-period of holding: cars wait in classification tracks for more cars coming in, as (i_4, i_5). Let $\mathcal{A}^{\text{CC}}$, $\mathcal{A}^{\text{CW}}$ and $\mathcal{A}^{\text{CH}}$ be the set of classification links, car waiting links and car holding links, respectively.

More vertical links are required to connect the car layer with the other two. One upward link is made from each OUT node in the car layer to the corresponding OUT node in the block layer. The link is called *form up link* and depicts the operation that blocks are formed up. On the contrary, one downward *break down link* connects two vertical IN nodes between the block layer and the car layer, and shows the process that blocks are broken down into cars.

A yard section of the 3-layer time-space network is illustrated in Figure 5.6.

We assign the car reception from and delivery to customers at the IN nodes of the car

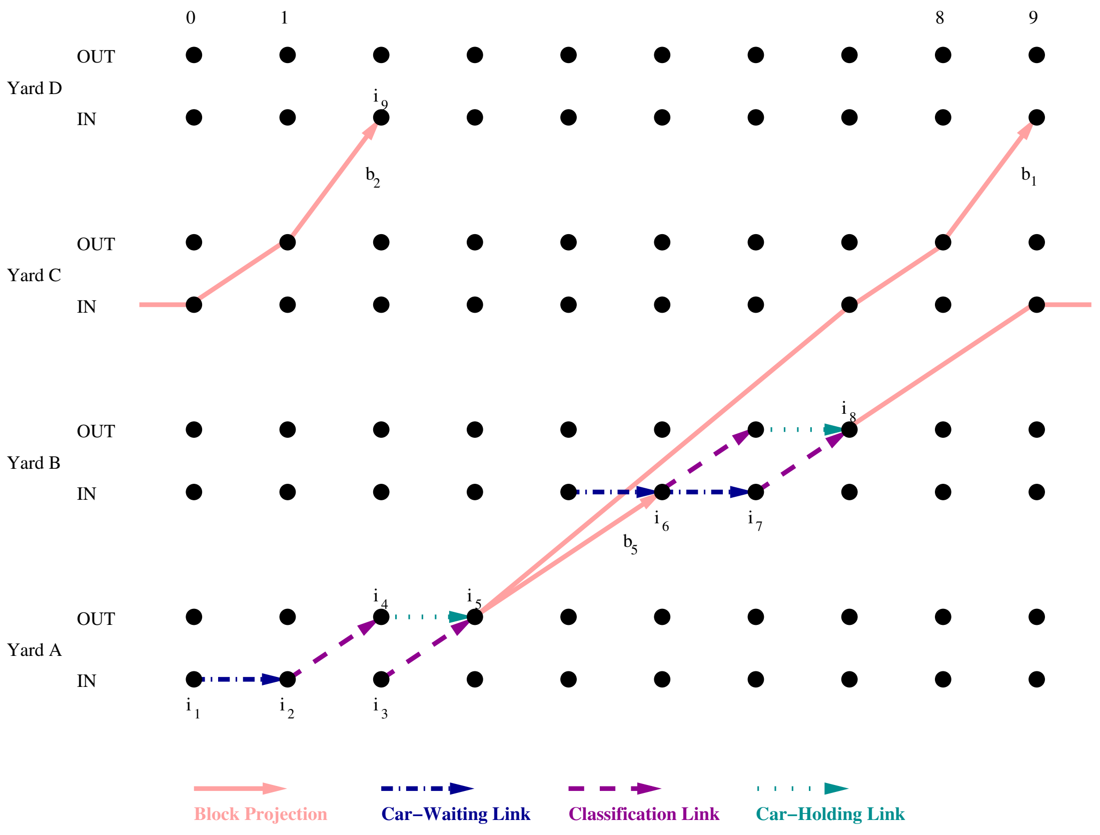

Figure 5.5: Car Layer

layer. The journey of a particular demand, denoted *itinerary*, is then a path in the 3-layer time-space network between two IN nodes of the car layer representing their origin and destination yards. From the origin IN node which represents the receiving at the origin yard, the cars first wait in car waiting links. After being classified through a classification link, the cars reach an OUT node, and are delayed on car holding links until enough cars are gathered. When the block is formed, the traffic goes up to the block layer through a block form-up link. Then, the block is loaded onto a service and shipped by service sections in the service layer. At an intermediate yard, the block is unloaded to the block layer, delayed and later transferred to another service. At the block destination, the block is broken down, and the traffic returns back to the car layer. The cars are delivered if it is the final yard of the shipment; otherwise, the cars will be re-classified and pass the block layer and service layer again.

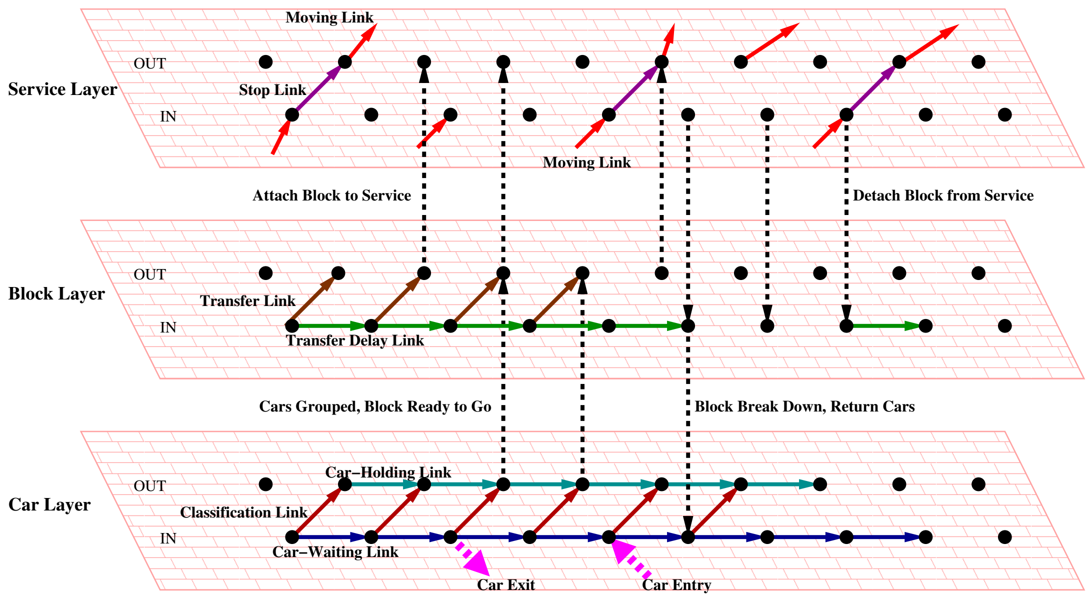

Figure 5.6: Yard Section of 3-Layer Time-Space Network

5.2 Integrated Service Network Design Model

Base on the 3-layer time-space structure proposed above, we proceed to define costs and capacities, and formulate the integrated service network design model.

To specify the characteristics of different demands, *traffic* (or *shipment*) is first defined. Each traffic p is specified by the origin yard $o(p) \in \mathcal{V}$, destination yard $d(p) \in \mathcal{V}$, type of commodity $m(p)$, quantity (number of cars) $w(p)$, receiving time $r(p)$ representing when cars are received from customer at their origin, and maximum delivery time $h(p)$ indicating the maximal transit time allowed. Let $\mathcal{P}$ be the set of all traffic. Each traffic therefore is associated with an O-D pair in the 3-layer structure, from an IN node representing the receiving of freight at the yard $o(p)$ at time $r(p)$, to another IN node representing the delivery at yard $d(p)$ at time $r(p) + h(p)$. We notice, $r(p) + h(p)$ might represent a time in the next planning cycle as we are working on a cyclic time dimension.

Denote $\mathcal{A}$ the union of links in all layers. We define three types of variables which are associated with car flow, block decision, and service decision respectively.

x_{ap}	≥ 0	number of cars of traffic p traveling on link $a \in \mathcal{A}$;
y_b	$\in \{0,1\}$	if we build block $b \in \mathcal{B}$, $y_b = 1$; otherwise $y_b = 0$;
z_s	$\in \{0,1\}$	if service $s \in \mathcal{S}$ is provided, $z_s = 1$; otherwise $z_s = 0$.

Denote x_{bp} the flow of traffic p on block b. Also, we define the indicator function $\delta_{al}^{sb} = 1$ if the moving link $a \in \mathcal{A}^{\text{SM}}$ of service s belongs to the service section $l \in \mathcal{L}(b)$ of block b, and 0 otherwise. Notation $x_{asp} = \sum_{b\in\mathcal{B}} \sum_{l\in\mathcal{L}(b)} \delta_{al}^{sb} x_{bp}$ indicates the flow of p on moving link $a \in \mathcal{A}^{\text{SM}}$ of service s, and $x_{sp} = \sum_{a\in\mathcal{A}^{\text{SM}}(s)} x_{asp}$ indicates the total workload of service s with respect to the cars hauled on all its moving links.

The Integrated Service Network Design (ISND) model is thus formulated as follows. All notations are summarized later in Table 5.I.

$$\Phi = \min \sum_{p\in\mathcal{P}} \sum_{a\in\mathcal{A}} c_{ap} x_{ap} + \sum_{b\in\mathcal{B}} c_b^{\text{F}} y_b + \sum_{s\in\mathcal{S}} c_s^{\text{F}} z_s \tag{5.1}$$

$$\text{s.t.} \sum_{a\in\mathcal{A}^+(n)} x_{ap} - \sum_{a\in\mathcal{A}^-(n)} x_{ap} = w_n^p \qquad \forall n \in \mathcal{N}^{\text{C}}, \forall p \in \mathcal{P}; \tag{5.2}$$

$$\sum_{p\in\mathcal{P}} x_{bp} \leq y_b u_b \qquad \forall b \in \mathcal{B}; \tag{5.3}$$

$$\sum_{p\in\mathcal{P}} x_{asp} \leq z_s u_s \qquad \forall a \in \mathcal{A}^{\text{SM}}, s \in \mathcal{S}; \tag{5.4}$$

$$y_b \leq z_{s(l)} \qquad \forall l \in \mathcal{L}(b), b \in \mathcal{B}; \tag{5.5}$$

$$\sum_{p\in\mathcal{P}} x_{ap} \leq u_a \qquad \forall a \in \mathcal{A}^{\text{CC}}; \tag{5.6}$$

$$\sum_{b\in\mathcal{B}(v,t)} y_b \leq u_v^B \qquad \forall v \in \mathcal{V}, \forall t \in \{0, \cdots, \mathbf{T}-1\}; \tag{5.7}$$

$$\sum_{s\in\mathcal{S}(e,t)} z_s \leq u_e \qquad \forall e \in \mathcal{E}, \forall t \in \{0, \cdots, \mathbf{T}-1\}; \tag{5.8}$$

$$x_{ap} \geq 0 \qquad \forall a \in \mathcal{A}, \forall p \in \mathcal{P}; \tag{5.9}$$

$$y_b \in \{0,1\} \qquad \forall b \in \mathcal{B}; \tag{5.10}$$

$$z_s \in \{0,1\} \qquad \forall s \in \mathcal{S}. \tag{5.11}$$

The objective function (5.1) is the sum of operating costs on all links, and costs of building blocks and providing services. A fixed cost c_s^{F} is defined on each service s, standing for the cost of supplying the service (power, crew, and so on), as well as assembling and disassembling the train. A fixed cost c_b^{F} is attached to block b representing the classification track occupancy during the building process and transfers indicated by the transfer links in the block path. Define $c_b^{\mathrm{F}} = c_{bo}^{\mathrm{F}} + \sum_{a \in \mathcal{A}^{\mathrm{BT}}(b)} c_{ba}^{\mathrm{F}}$, where c_{bo}^{F} is the fixed cost from classification track occupancy, and c_{ba}^{F} is the contribution from the swap operation on transfer link a.

c_{ap} is the unit flow cost for traffic p on link $a \in \mathcal{A}$. c_{ap} represents both operating costs and time cost on links. That is, for inter-yard moving links $a \in \mathcal{A}^{\mathrm{SM}}$, c_{ap} represents the hauling expense and time cost depending on the physical route passed, train running speed, etc. For car classification link $a \in \mathcal{A}^{\mathrm{CC}}$, c_{ap} is the time cost and handling expense for inspection and the power for moving cars from receiving tracks to classification tracks. c_{ap} represent mainly a time cost on the stop links $a \in \mathcal{A}^{\mathrm{SR}}$, block transfer links $a \in \mathcal{A}^{\mathrm{BT}}$, block transfer delay links $a \in \mathcal{A}^{\mathrm{BH}}$, car waiting links $a \in \mathcal{A}^{\mathrm{CW}}$, and car holding links $a \in \mathcal{A}^{\mathrm{CH}}$.

For the sake of later presentation, we define some notations. Let c_{lp} be the flow cost for traffic p on service section l, that is $c_{lp} = \sum_{a \in \mathcal{A}^{\mathrm{SM}}(l)} c_{ap} + \sum_{a \in \mathcal{A}^{\mathrm{SR}}(l)} c_{ap}$. c_{bp} is the flow cost for traffic p on block b, which is the sum of flow costs on its service sections, transfer links, and transfer delay links, $c_{bp} = \sum_{l \in \mathcal{L}(b)} c_{lp} + \sum_{a \in \mathcal{A}^{\mathrm{BT}}(b) \cup \mathcal{A}^{\mathrm{BH}}(b)} c_{ap}$. Apparently, if we look on the *car-layer projection* where potential blocks are mapped into the bottom car layer (as in Figure 5.5), the objective (5.1) can also be written as,

$$\Phi = \min \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}^{\mathrm{CC}} \cup \mathcal{A}^{\mathrm{CW}} \cup \mathcal{A}^{\mathrm{CH}}} c_{ap} x_{ap} + \sum_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} c_{bp} x_{bp} + \sum_{b \in \mathcal{B}} c_b^{\mathrm{F}} y_b + \sum_{s \in \mathcal{S}} c_s^{\mathrm{F}} z_s \qquad (5.12)$$

Traffic flow conservation constraint (5.2) guarantees the proper delivery for each traffic, wherein $\mathcal{A}^+(n)$ and $\mathcal{A}^-(n)$ represent the sets of outward and inward links of node $n \in \mathcal{N}^{\mathrm{C}}$, and w_n^p is the absolute demand for traffic p at node n. $w_n^p = w(p)$ if n is the origin node of p; $w_n^p = -w(p)$ if n is the destination node; otherwise $w_n^p = 0$.

Linking constraints (5.3) - (5.5) explicitly describe relations among cars, blocks, and services. Each block contains a number of cars, and a block must be built before any cars are assigned to it, thus we have (5.3). u_b is the capacity of block b in terms of the number of cars, which yields from the length of the classification track assigned to build the block. Constraint (5.4) states that for each service, the total number of cars on all blocks must be less than the capacity of the service u_s. More locomotives on a train may increase the hauling ability, however due to the infrastructure of the tracks and terminals passed, there is still a restriction on the load of each train. In this research, the maximal load u_s on service s is defined in terms of cars, which can also be modeled in terms of weight. Similar with (5.3), blocks are transported by service sections, and a service must be provided if one of its sections is asked by open blocks. The relation is described by (5.5).

In the operating process, we also meet restrictions from diverse facilities/resources of the rail system, such as the physical characteristics of the tracks, the yard construction and configuration, equipment allocation and crew assignment policy, etc. All those restrictions could be very close to each instance, and generally, at the tactical level, we have the rail constrains (5.6) - (5.8). In car handling constraint (5.6), u_a is the handling capacity on classification link $a \in \mathcal{A}^{\mathrm{CC}}$ in terms of cars, which derives from the yard classification capacity u_v^C. With the constraint, yard congestion is avoided and excess classification work must be delayed and performed in later time periods. Let $\mathcal{B}(v,t)$ be the set of blocks which are built at yard v in time period t, meaning $\forall b \in \mathcal{B}(v,t)$, t is in $h(b)$ time periods ahead of $o(b)$. Block building constraint described in (5.7) restricts the number of blocks being built in a yard, where the block building capacity u_v^B is applied. Train running constraint (5.8) is considered so as to restrict the number of trains running on a track at the same time. $\mathcal{S}(e,t)$ is the set of services running on physical track $e \in \mathcal{E}$ in time period t. With the proper train running constraints in consideration, we prevent the on-track congestion and ensure the on-time transportation of trains.

Finally, constraints (5.9) - (5.11) specify the domain of each variable.

The ISND model generates a mixed-integer programming formulation. By choosing z_s for all $s \in \mathcal{S}$, we characterize the service decision, including route, schedule, capacity and speed (by the length of moving links). The variable y_b decides which block should be built, and the make-up policy is addressed by the according $\mathcal{L}(b)$. With the variable x_{ap} for all the $a \in \mathcal{A}$ and $p \in \mathcal{P}$, we determine the routing of loaded and empty cars

Notation	Description
$\mathcal{G}$	rail network, $(\mathcal{V}, \mathcal{E})$ are the set of yards and tracks;
t	$\in \{0, \cdots, \mathbf{T}-1\}$, time point;
$\mathcal{A}^{\text{SM}}$	set of moving links in service layer;
$\mathcal{A}^{\text{SR}}$	set of stop links in service layer;
$\mathcal{S}$	set of services in service layer;
$\mathcal{A}^{\text{SM}}(s)$	set of moving links in service s;
$\mathcal{L}(s)$	set of service sections in service s;
$s(l)$	the service that section l belongs to;
$\mathcal{A}^{\text{BT}}$	set of transfer links in block layer;
$\mathcal{A}^{\text{BH}}$	set of transfer delay links in block layer;
$\mathcal{B}$	set of blocks in block layer;
$h(b)$	classification track occupancy time: before origin node $o(b)$, time required for a classification track assignment to build b;
$\mathcal{L}(b)$	set of service sections in block b;
$\mathcal{A}^{\text{CW}}$	set of waiting links in car layer;
$\mathcal{A}^{\text{CC}}$	set of classification links in car layer;
$\mathcal{A}^{\text{CH}}$	set of holding links in car layer;
$\mathcal{A}$	$\mathcal{A}^{\text{SM}} \cup \mathcal{A}^{\text{SR}} \cup \mathcal{A}^{\text{BT}} \cup \mathcal{A}^{\text{BH}} \cup \mathcal{A}^{\text{CW}} \cup \mathcal{A}^{\text{CC}} \cup \mathcal{A}^{\text{CH}}$, the union of links;
$\mathcal{P}$	set of traffic;
c_{ap}	flow cost of traffic $p \in \mathcal{P}$ on $a \in \mathcal{A}$;
c_{lp}	flow cost of traffic $p \in \mathcal{P}$ on service section $l \in \mathcal{L}$;
c_{bp}	flow cost of traffic $p \in \mathcal{P}$ on block $b \in \mathcal{B}$;
c_s^{F}	fixed cost for providing service s;
c_b^{F}	fixed cost for building block b, $c_b^{\text{F}} = c_{bo}^{\text{F}} + \sum_{a \in \mathcal{A}^{\text{BT}}(b)} c_{ba}^{\text{F}}$;
$\mathcal{A}^{+}(n)$	outward car layer links/blocks of node $n \in \mathcal{N}^{\text{C}}$;
$\mathcal{A}^{-}(n)$	inward car layer links/blocks of node $n \in \mathcal{N}^{\text{C}}$;
w_n^p	absolute demand for traffic p at node $n \in \mathcal{N}^{\text{C}}$;
u_a	car handling capacity on classification link $a \in \mathcal{A}^{\text{CC}}$;
$\mathcal{S}(e,t)$	set of services running on physical track e in time period t;
u_e	train running capacity on track $e \in \mathcal{E}$;
$\mathcal{B}(v,t)$	set of blocks building in the yard v and in time period t;
u_v^B	block building capacity of yard $v \in \mathcal{V}$;
u_s	train load capacity of service $s \in \mathcal{S}$;
u_b	block load capacity of block $b \in \mathcal{B}$.

Table 5.I: Notation Summary

and the processes cars should go through at each yard. Moreover, our model gives an approximated schedule for yard operations (classification and transfer) by x_{ap}.

This model considers the most important elements in the tactical planning level in rail freight transportation. By minimizing the total cost while satisfying various constraints, shipments are distributed through blocks and services in a cost-efficient manner, which also gives explicit decisions for block building and service providing. The model balances the trade-offs among blocking policy, train routing, scheduling, make-up policy and traffic distribution, and produces a skeleton for the operating plan.

5.3 Direct Service Network Design Model

When we look into the rail industry, sometimes a special case emerges where all services are direct (without intermediary stops). Such case often presents in industrial transportation (e.g. coal, agriculture products) or regional transportation.

Define a Direct Service Network Design (DSND) model a special case of ISND where only direct services present. We are particularly interested in the DSND because it shows a good property. With only one service section on each direct service, we have $\mathcal{S} = \mathcal{L}$, and blocks are actually transported by direct services other than service sections. In the scope of DSND, we use $\mathcal{S}$ to denote the direct service (or service section) set. This property allows degenerating the time-space network to a 2-layer structure with only the block layer and car layer, where direct services are mapped to the new block layer, which is shown in Figure 5.7. The car layer in the 2-layer structure remains unchanged.

Links in the block layer now include direct services, transfer links, and transfer delay links. A block is defined as a path in the block layer describing its journey and the associated handling operations. The path of a block in DSND is made up of a series of direct services, which are connected by transfer and transfer delay links. For example, the path ($i_3 \rightarrow i_4 \rightarrow i_5 \rightarrow i_6$) in Figure 5.7 represents a block put on a direct service departing from yard A at time 1, transferred at yard B at time 3, then moved by service (i_5, i_6) to its destination yard C at time 5.

The DSND model has the same objective function (5.1) to minimize the total operating cost, and respects constraint (5.2) - (5.11). However, given the fact that each direct service consists of only one moving link, $|\mathcal{A}^{\text{SM}}| = |\mathcal{S}|$, the train load constraint (5.4) can

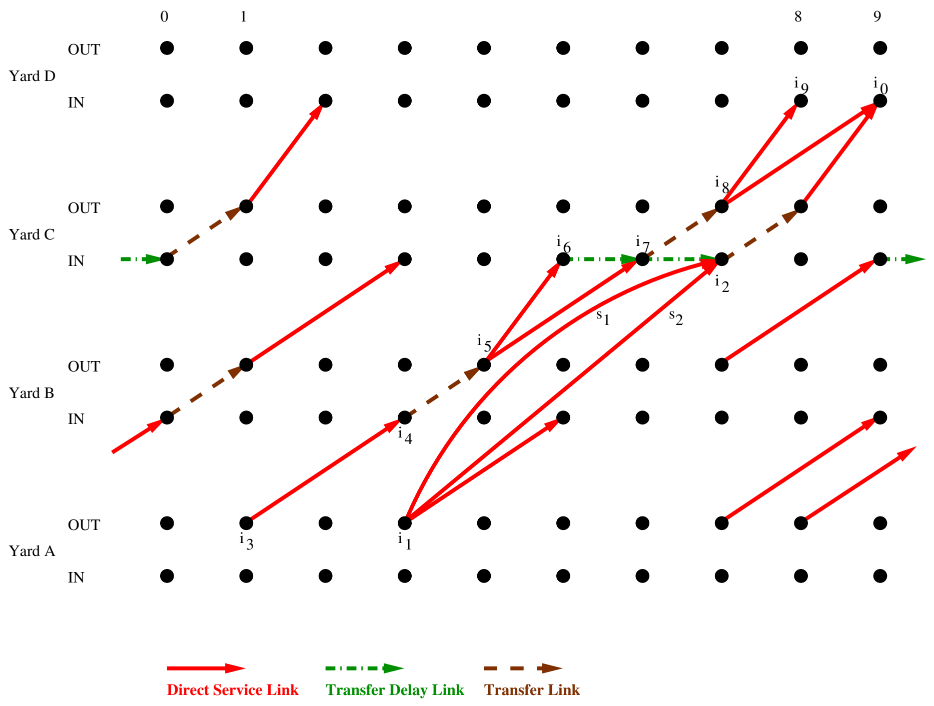

Figure 5.7: Block Layer of 2-Layer Time-Space Network

be rewritten as,

$$\sum_{p \in \mathcal{P}} x_{sp} \leq z_s u_s \qquad \forall s \in \mathcal{S}. \tag{5.13}$$

5.4 Summary

Compared with the physical network which presents the rail infrastructure, the 3-layer time-space network describes operations on services, blocks, and cars in respective layers, and addresses the temporal aspect of operations and delays with a time dimension.

The ISND model shows a complicated network design formulation. With service design variables in the service layer and block decisions in the block layer, the model also presents a multi-level network design. Other than the extreme complexity, very large instance size in applications is another obstacle that keeps ones from solving this problem efficiently.

A special simplification with only direct services is also discussed. With an interesting characteristic, DSND not only reduces the variable set of ISND (as only direct services and blocks consisting of direct services are considered), but also degenerates the network to a 2-layer structure.

CHAPTER 6

HEURISTIC ALGORITHMS

The ISND and DSND model proposed above can be viewed as a special form of the FMCND formulation. Moreover, when applied in rails, we are facing additional complexity introduced to account for the particularities of rail, which makes the problem even harder. In addition, for the instances of interesting size, given the plentiful amount of potential blocks to be built and services for selection, we obtain a large number of variables. As a result, we do not expect good performance from exact methods on this complex and large-sized problem, and only interested in heuristic algorithms which are capable to provide near-optimal solutions within reasonable computing efforts.

In this chapter, we study several heuristic solutions. Starting from DSND, we first develop a tabu search algorithm. An algorithm based on slope scaling method is later constructed for the ISND model. Then a hybrid heuristic combining the tabu search, slope scaling, and column generation ideas is developed for the very-large-size ISND.

6.1 Tabu Search

In the context of DSND model, we have only direct services, that is $|\mathcal{S}| = |\mathcal{L}| = |\mathcal{A}^{\text{SM}}|$. As a minimization problem, we notice in the DSND model, services which are not used by any open block should not appear in the final design, and whenever a block is open all its component direct services must also be open according to the linking constraints. Each block pattern thus has its optimal direct service pattern. The advantage gives us the privilege to focus on the block design, and only implicitly consider the service decisions during the solution procedure. Based on the observation, we define a neighborhood aiming at changing the status of blocks in the means of deviating traffic flows in the time-space network.

The cycle-based neighborhood idea is applied due to its promising performance on FMCND. Focusing on the modification of car flow, we work on the car-layer projection consisting of car-layer links and blocks. First we define a *cycle* as a closed chain consisting of car-layer links and blocks, and the car-layer links/blocks in the chain may follow different directions. A *block design* is a vector $\tilde{y}$ as well as the inherent optimal service design vector $\tilde{z}$. If $\tilde{y}$ honors the block building constraints (5.7) everywhere and the associated $\tilde{z}$ satisfies the train running constraints (5.8) on all tracks in each time period, we say the block design $\tilde{y}(\tilde{z})$ is *eligible*. For any solution with an eligible block design, the neighbors of current block design are defined as all the eligible block designs we can obtain through the steps in Figure 6.1.

1. Choose a block. Starting from the block, identify a cycle.
2. Deviate the total car flow through the cycle, from one path to the other, so that at least one block status changes.
3. On the cycle, open all the blocks with flow, and close all other blocks.

Figure 6.1: Cycle-based Neighborhood Specification

The neighborhood specified by the above procedure can be very large if we take all the blocks in consideration, and an enumeration of all potential cycles is impractical. To achieve good feasible solutions in a short computing time, we restrict the neighborhood scope in each move.

Given a block design $\tilde{y}(\tilde{z})$, the distribution of cars is given by a capacitated multicommodity network flow problem in the car-layer projection. The associated car flow problem is defined as,

$$\phi(\tilde{y}(\tilde{z})) = \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}^{\mathrm{CC}} \cup \mathcal{A}^{\mathrm{CW}} \cup \mathcal{A}^{\mathrm{CH}}} c_{ap} x_{ap} + \sum_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} c_{bp} x_{bp} \tag{6.1}$$

subject to traffic flow conservation constraint (5.2), car handling constraint (5.6), as well as the flow capacity on blocks (6.2) and trains (6.3), which are derived from (5.3) and

(5.13) respectively.

$$\sum_{p\in\mathcal{P}} x_{bp} \le \tilde{y}_b u_b \qquad \forall b \in \mathcal{B}; \tag{6.2}$$

$$\sum_{p\in\mathcal{P}} x_{sp} \le \tilde{z}_s u_s \qquad \forall s \in \mathcal{S}. \tag{6.3}$$

The problem is denoted as Traffic Routing Sub-Problem (TRSP). The objective of TRSP, denoted $\phi\big(\tilde{y}(\tilde{z})\big)$, includes the travel costs on all car-layer links and blocks, but does not have the fixed design costs of the block design. The DSND objective value on block design $\tilde{y}(\tilde{z})$ is then obtained as $\phi\big(\tilde{y}(\tilde{z})\big)$ plus the design costs, which is $\sum_{b\in\mathcal{B}} c^{\mathrm{F}}_b \tilde{y}_b + \sum_{s\in\mathcal{S}} c^{\mathrm{F}}_s \tilde{z}_s$.

Our tabu search algorithm starts with an initialization phase, which provides an eligible block design. Then, the algorithm iteratively improves the current block design by implementing a sequence of *cycle-based local searches* until the *stop criteria* are met. In each local search, the existing solution is improved by replacing the current block design with one of its neighbors, and then the traffic distribution is determined by the associated TRSP. When we reach a promising area, an *intensification* phase is called for further improvement. In case the improvement of a local search is insignificant or no good neighbor shows up, *diversification* is launched to escape from the local optimum. Schematically, an outline of the tabu search algorithm is shown in Figure 6.2,

1. *Initialization*.
2. Repeat until *stop criteria* are met,
    - *cycle-based local search*;
    - if the current solution is better than or close to the best overall, *intensification*;
    - else if the improvement is negligible, *diversification*.
3. Stop.

Figure 6.2: General Procedure of Tabu Search

In the following, we describe the main algorithmic choices and detail each procedure, except the stop criteria which will be set within computational experiments.

6.1.1 Initialization

The initialization phase provides an initial eligible block design. In order to satisfy the customers' demands to the maximum extent, it is preferable to open all potential blocks in the very beginning. However, the train running capacity u_e and the block building capacity u_v^B prevent us from opening all the blocks.

An integer programming formulation is solved to generate an initial eligible block design with maximal number of blocks. The problem has two sets of binary decision variables on blocks and services correspondingly. The objective (6.4) is to maximize the number of open blocks, subject to block building constraints (5.7) and train running constraints (5.8), as well as the linking constraints (5.5) between blocks and services. The integer programming problem is then solved by the branch-and-bound method. As there is little need to find the optimal solution for the initial problem, the solving procedure stops when a near-optimal solution is obtained in a short time.

$$\max \sum_{b \in \mathcal{B}} y_b \tag{6.4}$$

By opening as many blocks as possible, the initial block design leads to preposterously large design costs on both blocks and services. Nevertheless, the extra high fixed cost cannot guarantee the feasible flow of cars. This is due to the additional constraints considered in the model, coming from the car handling capacity restricting the number of cars that can be classified at each yard, as well as the car flow capacity on each block. The infeasibility of the according TRSP is avoided by introducing the *artificial links* (or *super itineraries*) in the car layer. For each traffic, an artificial link is appended, from the IN node that represents the receiving of the demand to the IN node corresponding to the delivery. The length of the artificial link equals to the due transit hour of the traffic. Each artificial link receives the same flow capacity as the volume of the associated traffic demand, and an arbitrarily high flow cost ensures that the artificial link will bear flow only if the distribution is infeasible. With the additional artificial links, the associated

TRSP always has a solution irrespective of the current block design. Our algorithm will start with the initial eligible block design, regardless if the artificial links are used or not, and try to reach a feasible solution before meeting the stop criteria.

6.1.2 Residual Network and Local Search

Starting from an initial block design, we progress to improve the current solution by running a series of *local searches*. The intention of local search is to divert car flows following a cycle, so that the status of one or several blocks is modified, and leads to significant modification of traffic flows. Keeping the idea in mind, we are interested in the cycles bearing enough car flow and after deviation flows on some blocks are empty so that those blocks are free to close.

First, we define *residual value* the flow volume one can deviate around the cycle. Consequently, we are interested in such cycles with their residual value equal to the flow on one open block. On a block design $\tilde{y}(\tilde{z})$, we denote Γ as the set of flow volumes on all open blocks,

$$\Gamma(\tilde{y}) = \{\sum_{p \in \mathcal{P}} x_{bp} > 0, b \in \mathcal{B} \text{ and } \tilde{y}_b = 1\}.$$

For each value $\gamma \in \Gamma$, we identify the cycles that support the variation of γ units of car flow, and an accessorial γ-*residual network* is constructed.

The γ-*residual network* has the same nodes as the car layer, and the conceptual arcs in the residual network represent the changing (either increasing or decreasing) of γ flow on blocks, classification links, car waiting links and car holding links. The γ-residual network uses a positive-cost arc to represent an increase of γ flow on the car-layer link/block when there is enough flow capacity remaining, and flow decrease on the car-layer link/block is described by an arc with negative cost if extra γ flow exists.

On different links/blocks, we have different regulations for introducing residual arcs. First, for each block b from nodes i to j, we introduce a forward arc $(i, j)^+$ if an additional γ cars of flow can pass on block $b = (i, j)$. That is, the residual capacity on b is greater than or equal to γ. A positive cost c^+_{ij} is associated with the arc $(i, j)^+$, approximates the cost for routing extra γ cars of unified commodity on the block, plus the possible design cost if the block, as well as the direct services underneath, are currently closed. Symmetrically, we include a backward arc $(j, i)^-$ if the total traffic flow on block $b = (i, j)$ is no less than γ. The cost on arc $(j, i)^-$ is the saving value for reduction of γ flow on block b. The possible design cost of block b, including the fixed cost on the block, and the potential fixed costs on some of the direct services if applicable, is also subtracted once the reduction leaves the block and services empty. The construction of the γ-residual network is illustrated in Figure 6.3. For an open block (i_3, i_5), we may have arc $(i_3, i_5)^+$ if there is enough flow capacity left and, arc $(i_5, i_3)^-$ if the current flow on the block is no-less-than γ. Corresponding to a close block (i_3, i_6), we include arc $(i_3, i_6)^+$ if the residual capacity on the block is over γ.

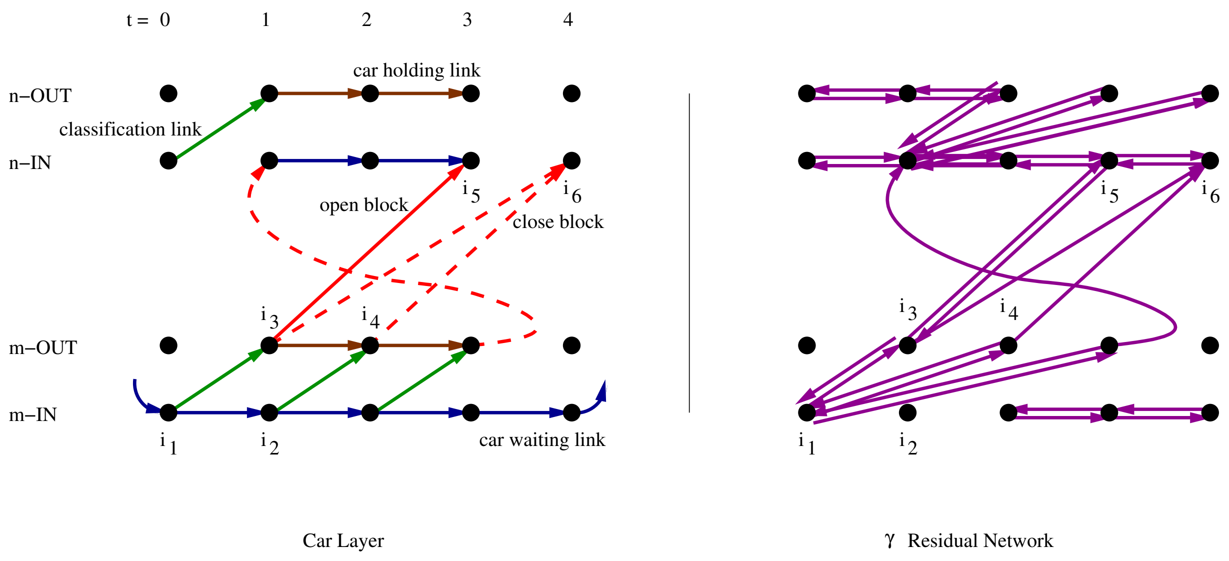

Figure 6.3: Construction of γ-Residual Network

On each car waiting link, e.g. (i_1, i_2), or car holding link, e.g. (i_3, i_4), both the $(i_1, i_2)^+$ and $(i_2, i_1)^-$, as well as the $(i_3, i_4)^+$ and $(i_4, i_3)^-$ can be defined following the same principle above. Particularly, we always have forward arc $(i_1, i_2)^+$ and arc $(i_3, i_4)^+$ since there is no flow capacity applicable on both car waiting links and car holding links. Focusing our attention on design decisions, instead of constructing residual arcs for each classification link, we aggregate the classification flows and build residual arcs to represent adding/subtracting flows on intra-yard paths made of classification links, car waiting links and car holding links between each (IN, OUT) node pair (IN and OUT are from a same yard). For example, an arc $(i_1, i_4)^+$ is used to represent that extra γ flow is added between i_1 and i_4, precisely on paths $(i_1 \rightarrow i_2 \rightarrow i_4)$ and $(i_1 \rightarrow i_3 \rightarrow i_4)$. If the total flow on these paths exceeds γ, an arc $(i_4, i_1)^-$ is included on which we associate a negative cost to represent the saving for the decline of γ flow on the intra-yard paths between i_1 and i_4.

Given an open block b in the current solution, we therefore generate a residual network with a residual value equal to the car flow γ on block b. Starting from the residual arc representing clearing the flow on the block, enumeration of all cycles passing through the arc in the γ-residual network is very expensive. We here are particularly interested in the lowest-cost cycle only, which implies the maximal savings following the deviation. The idea of forming such lowest-cost cycles is to find the lowest-cost path from j to i in the residual network to complete the cycle for an arc $(i, j)^-$. To find the path with the lowest cost from j to i, an *extended Bellman-Ford algorithm* is developed. The algorithm keeps a vector at each node, labeling the temporal length of the lowest-cost path. The length vectors eliminate over-length paths since by the assumption each demand must be delivered in its due transit hour. Furthermore, examinations are performed to avoid looping in lowest-cost paths because negative arcs present in the residual network. The pseudo code of the extended Bellman-Ford algorithm is described in Figure 6.4. From j to i, the algorithm is proven efficient to find a series of lowest-cost paths with different temporal lengths. We pick the one with the overall lowest cost, together with the origin arc, a *lowest-cost cycle* is then obtained.

In a local search, lowest-cost cycles deriving from all open blocks in the current solution are compared and the one with the minimum cost is selected. Furthermore, the minimum-cost cycle is accepted only if it is non-positive (means saving). A non-positive cost cycle accelerates the convergence, and has the highest probability to yield a better solution. We then move the flow following the minimum-cost cycle. After the deviation, the origin block of the cycle is cleared and closed. Some other blocks may be opened or closed accordingly to achieve the new design.

```
1. Define temporal length upper bound $\tau$.
2. For each node $n \in N$ and each $t \in \{1, \cdots, \tau\}$ do,
    • $distance(n, t) = \infty$ and $prev\_node(n, t) = \oslash$.
3. For each $t \in \{1, \cdots, \tau\}$ do,
    • $distance(j, t) = 0$.
4. $LIST = \{j\}$.
5. While $LIST \neq \oslash$ do,
    • remove an element $u$ from $LIST$;
    • for each node $v \in N^+(u)$ and each $t \in \{1, \cdots, \tau\}$ do,
        – $t' = t + len(u, v)$;
        – if $t' \leq \tau$ and $distance(v, t') > distance(u, t) + cost_{u,v}$ and $v$ is not in
          the path from $j$ to $u$ do,
            ∗ $distance(v, t') = distance(u, t) + cost_{u,v}$;
            ∗ $prev\_node(v, t') = u$;
            ∗ if $v \notin LIST$, then add $v$ to $LIST$.
6. Stop.
```

Figure 6.4: Extended Bellman-Ford Algorithm

An example for local search is given below. Figure 6.5 illustrates such journeys through a vertical projection of the 2-layer network on the car layer (only a few car-layer links and the projections of blocks are shown). Four demands are specified by their O-D node pairs. Notice that demands d_1 (i.e., from origin O_1 to destination D_1) and d_2 have the same origin and destination node, but they follow different itineraries in the current solution. d_1 goes through block b_4 and then block b_6, which means the traffic passes two classifications in yards A and C. d_2 is first delayed, but only block b_1 is used.

The instance shown in Figure 6.5 is further detailed in Figure 6.6, where the information on each link is illustrated as (fixed cost, unit flow cost, current flow, flow capacity).

Given the open block b_4 and its existing flow $\gamma = 20$, we generate a γ-residual network. As shown in Figure 6.7, emanating from the destination node of arc $(i_5, i_3)^-$,

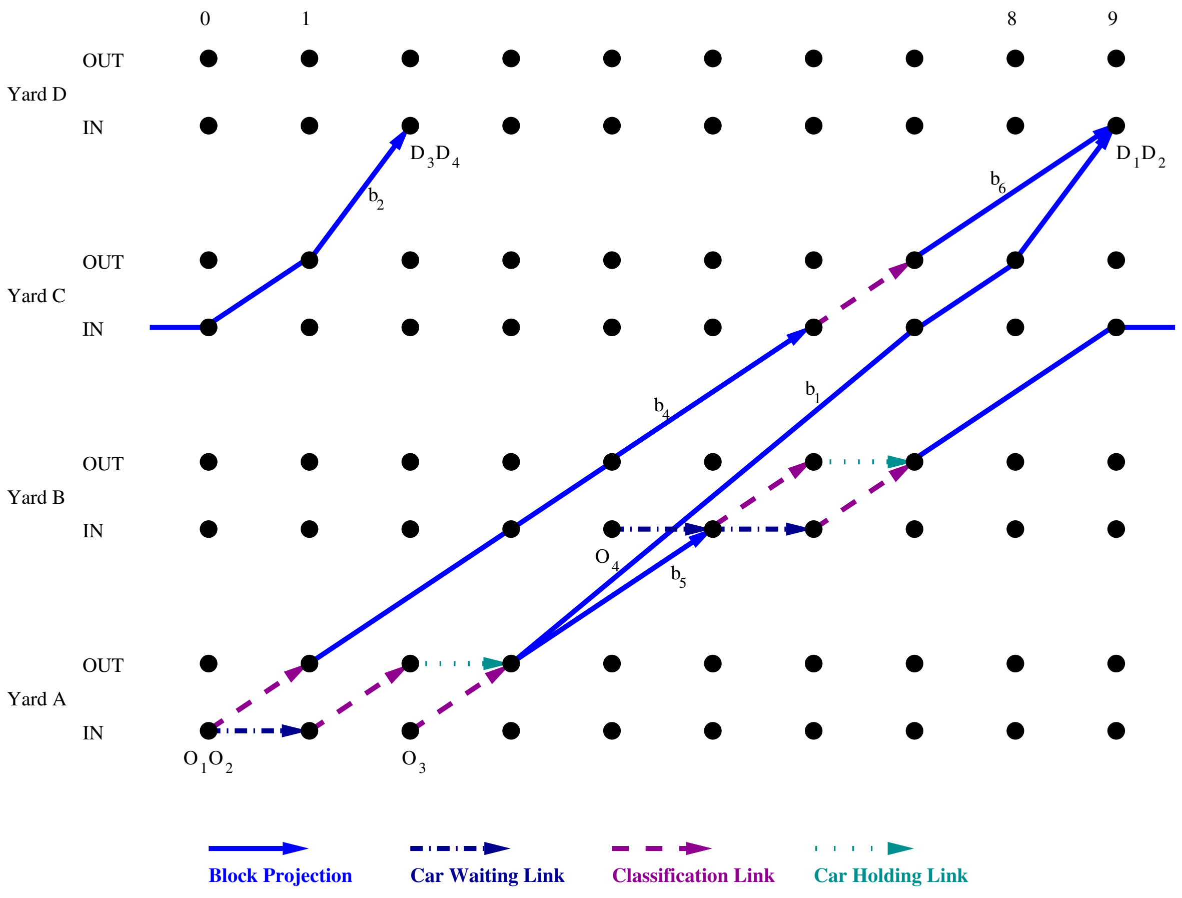

Figure 6.5: Flows in 2-Layer Time-Space Network (Projection)

representing eliminating γ flow on block b_4, we apply the extended Bellman-Ford algorithm. From i_3 to i_5, the path with the overall low cost is selected. Together with arc $(i_5, i_3)^-$, a lowest-cost cycle is formed.

Suppose the cycle shown in Figure 6.7 is also the minimum-cost cycle, among the lowest-cost cycles from all open blocks in the current design. Following the cycle, flows are moved and a neighbor block design is obtained (shown in Figure 6.8). On the new block design, demand d_1 now will be delayed and later go through block b_1 together with demand d_2, and blocks b_4 and b_6 are cleared and closed.

Two short-term memories, one for blocks and one for services, are used to prevent local search from looping or entering infeasible domains. The memories, called *block*

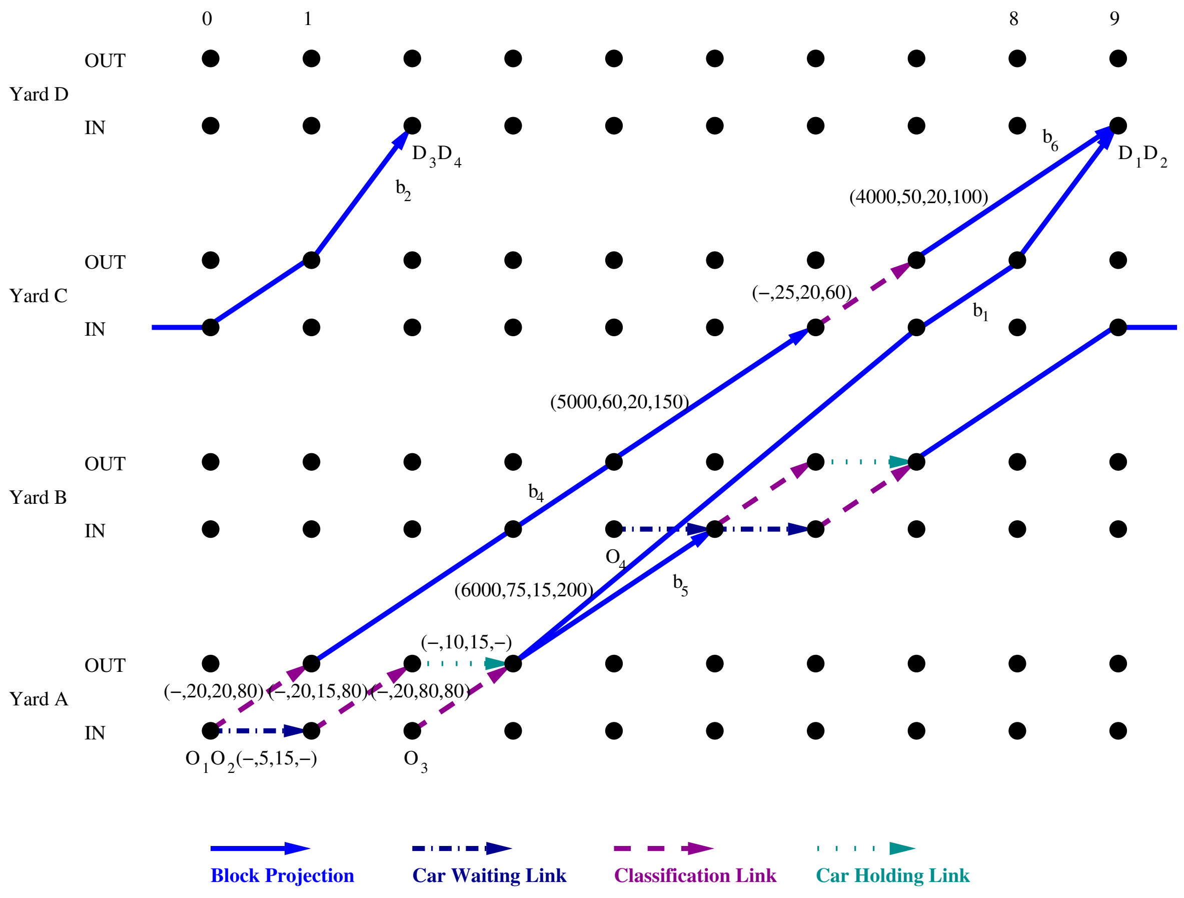

Figure 6.6: Solution in 2-Layer Time-Space Network (Projection)

tabu list and *service tabu list*, keep a tabu tenure for each block/service. A positive tenure means the block/service is prohibited to be opened or closed. When a new design is achieved, random tabu tenures are attached to all the blocks/services have just been modified. All positive tenures in both tabu lists descend by 1 each time we return to a feasible flow distribution on an eligible block design.

After a candidate neighbor block design is obtained through a non-positive cost cycle, a verification of the block building constraint and train running constraint is performed. The verification ensures the procedure remains on the eligible block design domain. If one of these constraints is violated, the last eligible block design is restored, and we tabu the blocks that disobey the block building constraint in the block tabu list,

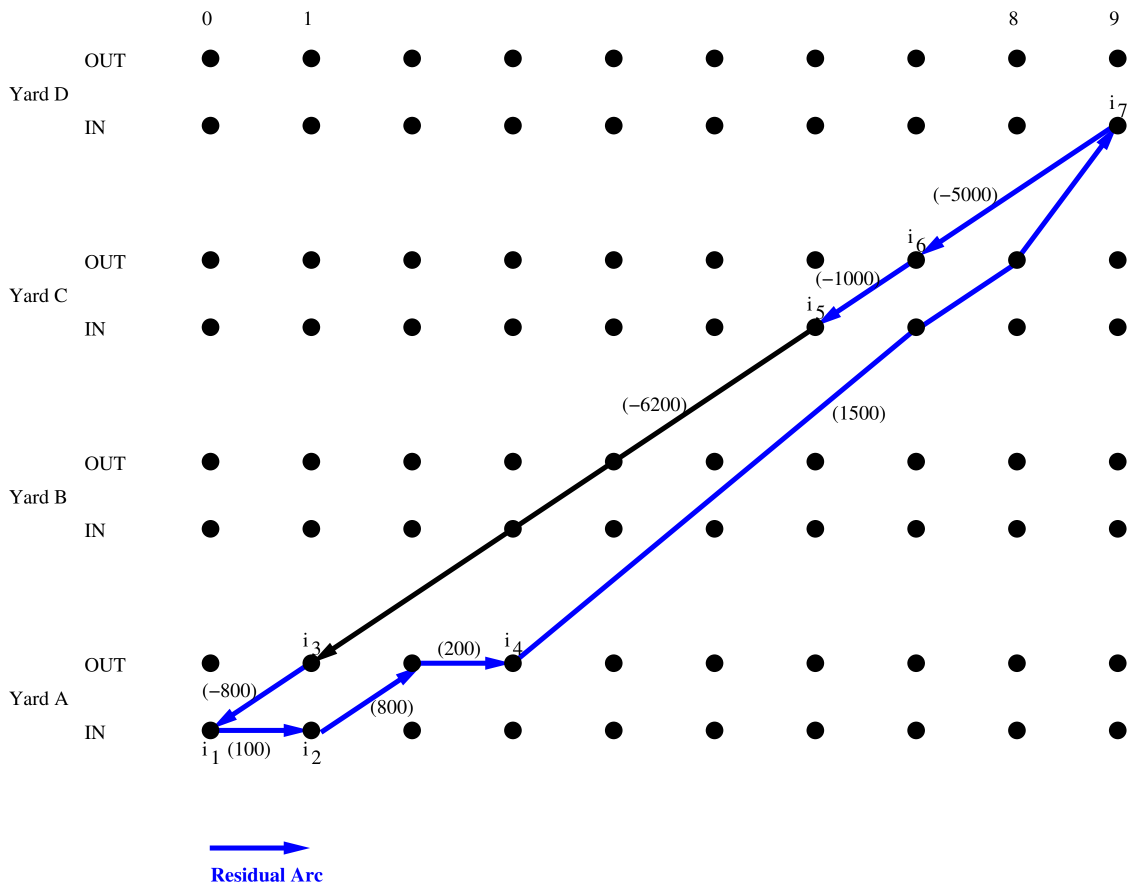

Figure 6.7: A Lowest-Cost Cycle in γ-Residual Network

as well as the illegal services for the train running constraint in the service tabu list. The local search then restarts to find another minimum-cost cycle consisting of non-tabu blocks and non-tabu services, and then the outcome block design is validated again. As long as the new block design satisfies the block building constraint and train running constraint, we move to the new eligible block design.

The traffic is re-distributed by solving the associated TRSP on the new block design. As a linear network flow problem, TRSP on a given block design can be well solved by the Simplex method. It is possible that artificial flows exist in the resulting TRSP solution. The reasons for the artificial flow are twofold. First, in the local search procedure, we failed to explicitly take the train load constraint into account, and the defacto

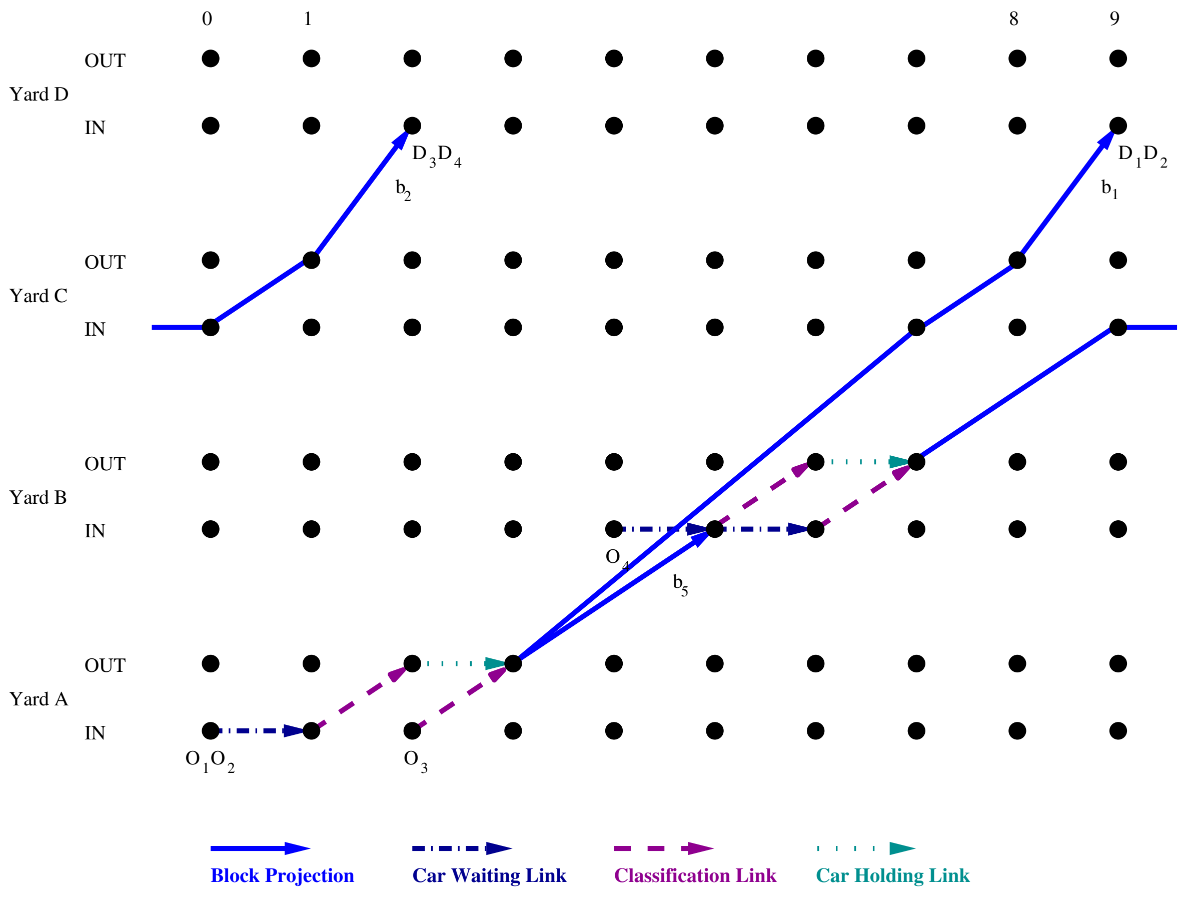

Figure 6.8: New Solution on Neighbor Block Design (Projection)

relaxation may lead to overflows on some services. Second, we move a group of demand flows at each local search move. As a common part of traffic itineraries is replaced, the modification may not be practicable for all the relative demands. A *restoration* phase is then implemented to regain a feasible flow distribution. The restoration phase is similar to the local search procedure. However, we generate cycles deriving from artificial links instead of open blocks, and the residual value set Γ is restricted to the flow volumes on artificial links.

The local search procedure is synthesized in Figure 6.9.

1. Given the current solution, for each open block b do,
   - determine the current flow γ on block b;
   - construct the γ-residual network;
   - based on block b, apply the extended Bellman-Ford algorithm to determine a set of lowest-cost paths with different lengths by non-tabu blocks and services;
   - pick the path with the lowest cost, and form a lowest-cost cycle;
   - update the minimum-cost cycle.
2. If the minimum-cost cycle is non-positive do,
   - following the minimum-cost cycle, obtain a candidate block design;
   - update *block tabu list* and *service tabu list*;
   - if the candidate block design is eligible,
     - move to the candidate block design by opening and closing the appropriate blocks and the associated direct services;
     - solve the associated TRSP;
     - if artificial links are used, *restoration*;
3. Stop.

Figure 6.9: Local Search Procedure in Cycle-based Neighborhood

6.1.3 Intensification

One of the privileges of the cycle-based neighborhood is the wide range of moves in the solution space. However, when working in a petite promising space, it becomes a drawback and the search may “step over” the optimum with a faulty cycle, which suggests a considerable cost reduction but results in an infeasible solution. We propose the *intensification*, a supplementary procedure aims to elaborately polish the current solution. Two separate intensification phases are employed in a sequence.

In the first intensification phase, a *sliding problem* is solved. The sliding problem has the same formulation as the DSND model, and only includes the blocks currently open as well as their “parallel” blocks: the blocks with the same routing departing at the previous time point as well as the next time point. The sliding problem integrates many cycles that represent postponing/advancing the existing open blocks for one period, and it helps to converge steeply by avoiding the analysis of a sequence of "small" cycles separately. With only a few variables considered, the sliding problem is tractable with the MIP solver.

No matter the first intensification progresses or not, we implement a second phase. The second phase can be viewed as a special local search procedure, and we consider the flow of a demand at each time, assuming the flows of other demands remain unchanged. The same cycle-based neighborhood definition is applied. As in the local search, a set of residual values is first determined, which is now the flow of one traffic on every open block. For a traffic $p \in P$, Γ_p is then defined as $\Gamma_p(\tilde{y}) = \{x_{bp} > 0, b \in \mathcal{B}$ and $\tilde{y}_b = 1\}$. A γ-residual network is built based on each $\gamma \in \Gamma_p$, and the lowest-cost cycles are formed and evaluated. Moves are adapted only if the cycle with the minimum cost has a negative value. The process repeats until no negative-value cycle is detected in any γ-residual network, and then the process continues to the next traffic p. Since the pivots in the intensification phase concern the flow of one traffic only, there is no need to re-solve the associated TRSP every time. Car flows will be shifted by hand, as long as the train load constraints are satisfied.

The intensification phases are implemented when reaching a hopeful solution area, that is, when a new best or near-best solution is found.

6.1.4 Diversification

In case the local search fails to improve the current best solution, or the improvement is insignificant, it implies that we are restricted in a local optimum. As we are working in a huge solution space, being trapped in a local area is inevitable. To jailbreak from the local optima, we propose a *diversification* process.

The diversification is actualized by two long-term memories, which are used to keep the opening frequency for blocks and services. These memories are denoted as *block frequency list* and *service frequency list*, respectively. Each time a feasible flow distribution on an eligible block design is obtained, we accumulate the opening frequencies of all the open blocks and open services by 1 in the frequency lists.

To diversify the solution space, a small fraction of open blocks with high opening frequencies are forced to close. Also, tabu tenures of these blocks are updated to avoid returning back in the following several local search moves. Same strategy is applied on services, and by closing the most-opened services, all blocks passing on the services are also closed. Furthermore, to prevent closing the same backbone block repeatedly, an additional local memory is adopted to keep the diversification history.

With the closure of several essential blocks, artificial links might be used to carry extra flows. Restoration process is applied to find an alternative block design and recover the feasible traffic distribution. The local search will restart there when a feasible solution is achieved.

The tabu search we proposed performs local search moves in the cycle-based neighborhood. Compared with the cycle-based tabu search by Ghamlouche et al. [57], in which the algorithm evaluates all the cycles coming from all the links, our local search is restricted to the cycles deriving from blocks currently open, and we only adopt the non-positive cycles. Furthermore, in our algorithm, we focus our attention on the design decisions only, and leave the intra-yard flow determined by the network flow problem. All these arrangements impel the procedure to converge faster, and enable us to find superior solutions within logical solving time.

6.2 Slope Scaling with Ellipsoidal Search

Compared with DSND, the ISND with multi-stop services is based on the 3-layer time-space network structure, which makes ISND much more complex. One major advantage of the 3-layer structure is to detail the relations between service sections and blocks, however, it also puzzles the previous tabu search algorithm proposed on the 2-layer structure. The effectiveness of the tabu search comes from the cycle-based neighborhood applied, which provides good reduced cost evaluations for opening or closing blocks. The cycle-based neighborhood fails on ISND because blocks are transported by service sections but direct services in the 3-layer structure. For example, in the previous structure, if closing a block also leaves one direct service empty, we count the fixed service cost into the block closing contribution; while in the 3-layer structure, closing a block does not lead to the change of services because only service sections are left empty, i.e. services are only partly empty. All these require the development of a new solution algorithm.

To develop a solution for the ISND, we applied the slope scaling method. Given the defect of slope scaling on the general FMCND (as reviewed in Section 2.3), we adopt a factor perturbation process similar with [44]. Moreover, a new population-based approach, named as *ellipsoidal search* is proposed to further improve the solution quality. The overall procedure is guided by long-term memories, which conduct both factor perturbation procedure for slope scaling and ellipsoidal search procedure. In the rest of this section, we detail slope scaling, long-term memory perturbation, ellipsoidal search and the overall procedure, respectively.

6.2.1 Slope Scaling Procedure

In the ISND formulation, block building constraint (5.7) and train running constraint (5.8) concern only binary design variables. Define R1-ISND a relaxed ISND by removing constraints (5.7) and (5.8). We observe, given a flow distribution $\tilde{x}$, it is easy to produce the corresponding block design $\tilde{y}$ as well as service design $\tilde{z}$ underneath by opening only blocks and services bearing car flow. The traffic distribution, together with the open blocks and services thus form a feasible solution for the R1-ISND. Due to the absence of constraints (5.7) and (5.8), the design $\tilde{y}(\tilde{z})$ might be infeasible for the original ISND. Nevertheless, solutions of the R1-ISND can be remedied by postponing or advancing some open blocks and services, to release the overflows on some tracks and in some yards. In this algorithm, we apply the slope scaling method to solve the R1-ISND and generate feasible ISND solutions whenever good solutions of R1-ISND are achieved.

To approximate the total cost in R1-ISND, two vectors of linearization factors are defined, which are α_b on each block b and β_s on each service s. In each iteration, linearization factors are modified to reflect the exact fixed costs. An Approximation

Problem with factor α and β is defined as,

$$\text{AP1}(\alpha, \beta) = \min \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} c_{ap} x_{ap} + \sum_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} \alpha_b x_{bp} + \sum_{s \in \mathcal{S}} \beta_s x_{sp} \tag{6.5}$$

subject to traffic flow conservation constraint (5.2), car handling constraint (5.6), as well as the flow capacity on trains (6.6) and blocks (6.7).

$$\sum_{p \in \mathcal{P}} x_{asp} \leq u_s \qquad \forall a \in \mathcal{A}^{\text{SM}}, s \in \mathcal{S}; \tag{6.6}$$

$$\sum_{p \in \mathcal{P}} x_{bp} \leq u_b \qquad \forall b \in \mathcal{B}. \tag{6.7}$$

AP1(α, β) is a linear multicommodity capacitated network flow problem which can be solved by the Simplex method, as we solve TRSP in tabu search. However, given the larger size of ISND instance, it takes an overlong time and extra memory for the Simplex method to reach an optimum. A heuristic solver based on the shortest augmenting path algorithm is proposed to solve AP1, where flows are assigned to the lowest cost itinerary with positive residual capacity. The residual capacity of an itinerary is the minimal residual capacity on classification links and blocks passed. Moreover, on each block, we also consider the train load capacity on all service sections used. Experiments in Section 7.2 show that the heuristic algorithm generates very good solutions with much shorter solving time comparing with the Simplex method.

Given a setting of linearization factors, an AP1 is determined and solved, the flow pattern $\tilde{x}$ then leads to a design where only blocks and services with positive car flows are opened. If the total traffic flow on a block is positive, say $\sum_{p \in \mathcal{P}} \tilde{x}_{bp} > 0$ on block b, $\tilde{y}_b = 1$ and $\tilde{y}_b = 0$ otherwise. Symmetrically, in the case the total workload $\sum_{p \in \mathcal{P}} x_{sp}$ on service s is positive, $\tilde{z}_s = 1$, and $\tilde{z}_s = 0$ on the contrary. Consequently, the total design cost is the sum of fixed costs from open blocks and open services, $\sum_{b \in \mathcal{B} | \tilde{y}_b = 1} c_b^{\text{F}} + \sum_{s \in \mathcal{S} | \tilde{z}_s = 1} c_s^{\text{F}}$. To reflect the total design cost, condition (6.8) should be satisfied.

$$\sum_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} \alpha_b \tilde{x}_{bp} + \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}} \beta_s \tilde{x}_{sp} = \sum_{b \in \mathcal{B} | \tilde{y}_b = 1} c_b^{\text{F}} + \sum_{s \in \mathcal{S} | \tilde{z}_s = 1} c_s^{\text{F}} \tag{6.8}$$

The left side of (6.8) is the second component in the AP1 objective (6.5), and it can be rewritten as,

$$\sum_{b\in\mathcal{B}|\tilde{y}_b=1} \alpha_b \sum_{p\in\mathcal{P}} \tilde{x}_{bp} + \sum_{s\in\mathcal{S}|\tilde{z}_s=1} \beta_s \sum_{p\in\mathcal{P}} \tilde{x}_{sp}$$

As a result, we modify linearization factors with the following rules.

$$\alpha_b = c_b^{\mathrm{F}} / \sum_{p\in\mathcal{P}} \tilde{x}_{bp} \qquad \text{if } \tilde{y}_b = 1; \tag{6.9}$$

$$\beta_s = c_s^{\mathrm{F}} / \sum_{p\in\mathcal{P}} \tilde{x}_{sp} \qquad \text{if } \tilde{z}_s = 1. \tag{6.10}$$

And the linearization factors of close blocks and close services remain unchanged. It is seen that with this setting, condition (6.8) is respected.

Linearization factors at iteration i are denoted as $\alpha(i)$ and $\beta(i)$. The initial factor values are set as,

$$\begin{aligned} \alpha_b(0) &= c_b^{\mathrm{F}}/u_b && \forall b \in \mathcal{B}; \\ \beta_s(0) &= c_s^{\mathrm{F}}/(u_s|\mathcal{A}^{\mathrm{SM}}(s)|) && \forall s \in \mathcal{S}. \end{aligned}$$

We use the aggregated flow capacity (aggregation of capacities on component moving links) in the β definition, because β is updated by the total workload x_{sp} of the service. By Kim and Pardalos [77], this setting provides an effective initial solution.

In iteration i, AP1 with factors $\alpha(i)$ and $\beta(i)$ is then solved, and the outcome car flow pattern is used to complete a R1-ISND solution by opening only blocks and services with positive flow. Linearization factors in iteration $i+1$ are then updated by (6.9) and (6.10), and the procedure proceeds to solve a new AP1 problem.

The overall history of slope scaling is stored in long-term memories, which track three service and block utilization measures over all iterations (the measures are not re-initialized following perturbation and ellipsoidal search phases). The *average flow*, *peak flow* and *opening frequency* on each block, as well as on each service, of the previous iterations are kept. For each block b, the average flow $\bar{x}_b$ is the mean of previous car flows on the block, defined as $\sum_{1\leq j\leq i} \sum_{p\in\mathcal{P}} \tilde{x}_{bp}(j)/i$, where $\tilde{x}_{bp}(j)$ is the flow of traffic p on block b in iteration j. The peak flow $\hat{x}_b$ is the maximal car flow on the block so far, $\hat{x}_b = max_{1 \le j \le i}\{\sum_{p \in \mathcal{P}} \tilde{x}_{bp}(j)\}$. The opening frequency f_b is the number of iterations with $\sum_{p \in \mathcal{P}} \tilde{x}_{bp} > 0$. Similarly, on each service s, the average workload $\bar{x}_s = \sum_{1 \le j \le i} \sum_{p \in \mathcal{P}} \tilde{x}_{sp}(j)/i$, peak workload $\hat{x}_s = max_{1 \le j \le i}\{\sum_{p \in \mathcal{P}} \tilde{x}_{sp}(j)\}$, and f_s denotes the opening frequency of service s. The long-term memories are used within the perturbation and ellipsoidal search phases.

In the slope scaling procedure, each time we find identical flow distributions in consecutive iterations, there is no update for linearization factors and slope scaling is stalled. Moreover, we observe, though a different solution may have been obtained for the AP1 problem, the corresponding feasible R1-ISND solution might not improve with respect to the previous one, and sometimes it takes many iterations to converge to a local optimal R1-ISND solution. Given the large magnitude of the 3-layer time-space network, solving a series of AP1 could be rather time-consuming even if they are solved heuristically. Another parameter $I_{max}^{nonimprove}$ is used to regulate the maximum number of continuous iterations without any improvement on the current objective of R1-ISND. Therefore, we stop slope scaling iterations either it fails to determine a different traffic distribution in continuous slope scaling iterations or we reach the maximum number of non-improved iterations according to the R1-ISND solution.

The local optimal solution of R1-ISND obtained may have overflows on tracks and yards, and violates block building constraints as well as train running constraints in the ISND formulation. To regain feasibility, we slide some open blocks/services forward or backward along the time horizon, in order to lighten the block building load at some yards and reduce services on some tracks. Based on a R1-ISND design $\tilde{y}(\tilde{z})$, a *sliding problem* is solved to search feasible solutions for the ISND. Similar as defined in Section 6.1.3, the sliding problem is a small-size ISND, with only open blocks/services in $\tilde{y}(\tilde{z})$, and their parallel neighbors. Parallel neighbors of a service include services with the same train route and transit time, departing at the previous and the later time point. Parallel neighbors of a block are defined similarly. Such a sliding problem has only a small portion of design variables, and can be efficiently solved by a MIP solver. It is possible that a sliding problem does not return any feasible solutions according to the given set of open blocks/services and their parallel neighbors. The infeasibility implies that there exist too many blocks building at some yard at a time or too many services running on some track simultaneously. In this case, penalties are associated to the relevant linearization factors following the diversification strategy (to be introduced in the next section), and we continue with the next iteration of slope scaling.

6.2.2 Factor Perturbation

Factor perturbation phase disturbs linearization factors in order to guide the search into a promising area of the solution space where slope scaling can again proceed. Perturbations are implemented by incenting or reducing linearization factors. Two perturbation strategies are proposed, *diversification* and *intensification*, which follow the same guideline as the diversification and intensification in tabu search. In diversification, we would like to move the search into a not-yet explored area of the solution space, by making some blocks/services that attract flow frequently less interesting and some blocks/services more interesting if they are rarely used. On the contrary, intensification solidifies the blocks/services with high flow frequency, and leaves the rest less interesting.

In either diversification or intensification, one key question is which linearization factors should be adjusted. To measure the blocks/services frequently or rarely used, four frontier values δ^+, δ^- and θ^+, θ^- are defined by opening frequency f_b and f_s. The average opening frequency $\bar{f}_{\mathcal{B}}$ and $\bar{f}_{\mathcal{S}}$, and the standard deviation $\sigma_{\mathcal{B}}$ and $\sigma_{\mathcal{S}}$ are first calculated. For blocks, δ^+ and δ^- are computed as, $\delta^+ = \bar{f}_{\mathcal{B}} + \omega^+\sigma_{\mathcal{B}}$, $\delta^- = \bar{f}_{\mathcal{B}} - \omega^-\sigma_{\mathcal{B}}$. For services, we have $\theta^+ = \bar{f}_{\mathcal{S}} + \omega^+\sigma_{\mathcal{S}}$ and $\theta^- = \bar{f}_{\mathcal{S}} - \omega^-\sigma_{\mathcal{S}}$, where ω^+ and ω^- are parameters. δ^+ and δ^- mark the frontiers of block frequency. If frequency f_b is bigger than δ^+, block b is viewed as frequently used. On the contrary, when $f_b < \delta^-$, we say b is rarely used in previous iterations. Same rule applies on services with frontiers θ^+ and θ^-.

While δ^+ and δ^- regulate the blocks frequently or rarely used, rewards or penalties are associated to the linearization factors so that the blocks are more/less interesting in the following iterations. The extent of the reward or penalty is conducted by the ratio

$\mu_b = \bar{x}_b/\hat{x}_b$ ($\mu_b = 0$ if $\hat{x}_b = 0$). The ratio is used to estimate the flow variability. When μ_b is close to 1, it means the historical car flows on block b are always close to the maximal flow. If μ_b approaches 0, the amounts of flow attracted by block b varies significantly along iterations. In diversification, higher penalties are added to the blocks with high variability, and less penalties to those with small variations. We define in diversification, for each block $b \in \mathcal{B}$, $\alpha_b = \alpha_b(1+\mu_b)$ if $f_b \geq \delta^+$, and $\alpha_b = \alpha_b\mu_b$ if $f_b < \delta^-$. Symmetrically, in intensification, higher rewards for less variations, and lower rewards for higher variations. For each block $b \in \mathcal{B}$, $\alpha_b = \alpha_b(1-\mu_b)$ if $f_b \geq \delta^+$, and $\alpha_b = \alpha_b(2-\mu_b)$ if $f_b < \delta^-$. Same rule is applied on services with definition $\mu_s = \bar{x}_s/\hat{x}_s$ ($\mu_s = 0$ if $\hat{x}_s = 0$). When diversification, for each service $s \in \mathcal{S}$, $\beta_s = \beta_s(1+\mu_s)$ if $f_s \geq \theta^+$, and $\beta_s = \beta_s\mu_s$ if $f_s < \theta^-$. In intensification, $\beta_s = \beta_s(1-\mu_s)$ if $f_s \geq \theta^+$, and $\beta_s = \beta_s(2-\mu_s)$ if $f_s < \theta^-$.

As we have two perturbation strategies, the next question is which one, either diversification or intensification, should we choose at each perturbation. Apparently, a sequence of continuous intensifications emphasize and explore the local area well with a price of ignoring other promising areas, and over-diversified search covers more blocks/services with a chance of missing the optimum. To avoid the situation where we intensify or diversify too much, two other parameters are defined. Let I^{inten}_{max} denote the maximal number of continuous implementations of intensification, and I^{diver}_{max} for diversification. I^{inten}_{max} prohibits repeated intensifications and I^{diver}_{max} forces the process exit diversification and look deeper for better solutions.

After perturbation, the slope scaling then restarts from the new linearization factor setting. We denote SS+LMP the procedure which iterates between slope scaling (SS) and long-term memory perturbation (LMP).

6.2.3 Ellipsoidal Search

As the slope scaling procedure is only applied indirectly, sometimes ISND solutions suggested by the R1-ISND leave a considerable optimality gap. Given a series of feasible solutions produced by sliding problems, we intent to incorporate the promising attributes shown from the solutions, and generate better solutions. A new population-based algorithm is proposed.

The algorithm derives from the *path relinking* scheme: during the slope scaling procedure, each time a sliding problem is solved, the obtained feasible solution is added into a *reference solution set* if applicable; then two reference solutions in the reference set are selected, one as the *initial solution* and the other as the *guiding solution*. Instead of finding a path between two solutions in a neighborhood as the classical path relinking, we shape an ellipsoid space, and apply a MIP solver to explore the ellipsoid space. The initial solution is then removed from the reference set. If the MIP solver finds a different solution than the guiding solution, the reference set is updated by the new solution, and the algorithm restarts, otherwise stop. We name this new method *ellipsoidal search*, and a pseudo code is given in Figure 6.10. In the following, we discuss the major components in the ellipsoidal search, including reference solution selection and ellipsoid shaping.

1. Given a *reference solution set*.
2. Select *guiSol* and *iniSol*.
3. Shape an ellipsoid space.
4. Generate an intermediate feasible solution in the ellipsoid with MIP solver.
5. Remove *iniSol* from the reference set.
6. If the intermediate feasible solution updates the reference set, go to step 2.
7. Stop.

Figure 6.10: Ellipsoidal Search Procedure

In our algorithm, feasible reference ISND solutions are provided during the slope scaling by solving sliding problems, and the quality and diversity of reference solutions are supported by factor perturbations, which is conducted by long-term memories.

Identification of the initial and the guiding solution is critical as "seeds" are important to an evolutionary algorithm. Several policies for choosing initial and guiding solutions are investigated by Ghamlouche et al. [58] regarding the FMCND problem. Their conclusion indicates that two policies are of the most interesting and with comparable performance. One is to define guiding solution as the best solution in the reference set, while initial solution as the solution with the maximum hamming distance from the guiding solution; the other is to define guiding solution and initial solution as the most distant solutions in the reference set. The success of these policies emphasized the importance of diversity of the reference solutions. The first policy is more theoretically sound, as on one hand, it preserves the overall best solution, and on the other hand it provides the maximal variation. We choose to work with this referent selection policy.

Shaping an ellipsoid also has great impacts to the quality and efficiency of the procedure. An ellipsoid solution space is built with the *local branching* idea. Equipped with the initial solution and the guiding solution, we first calculate the hamming distance between the two solutions, denoted it as k. An *elite problem* is defined as an ISND with k-distance cut, and fed with the *initial solution* as the starting solution. The elite problem therefore explores the k-opt neighborhood of the initial solution, which is defined as all solutions with maximal k design variable status changed from the initial solution. Apparently, the k-opt neighborhood of the initial solution covers the guiding solution. The k-distance cut in the elite problem dramatically reduces the solution space, however with a large number of design variables, the k-opt neighborhood could still be very big. The k-opt neighborhood is further trimmed, and an ellipsoid (illustrated in Figure 6.11) is shaped.

The process of shaping an ellipsoid space in the elite problem includes variable selection and variable fixing. Variable selection chooses only a set of decision variables to be considered in the elite problem in order to further reduce the searching space, and variable fixing preserves some variables with values according to the guiding solution, to ensure the proper searching direction. According to the nature of ISND, we have choices to select or fix binary variables basing on either block or service decisions as blocks and services are interrelated. Compared with block decisions, service decisions generally have more impacts as each service is shared by many blocks. Missing one key service variable in the elite problem will severely deteriorate the solution quality. As a result, in the following we shape ellipsoids based on block decisions, i.e. select and fix block decision variables, and include all service variables accordingly.

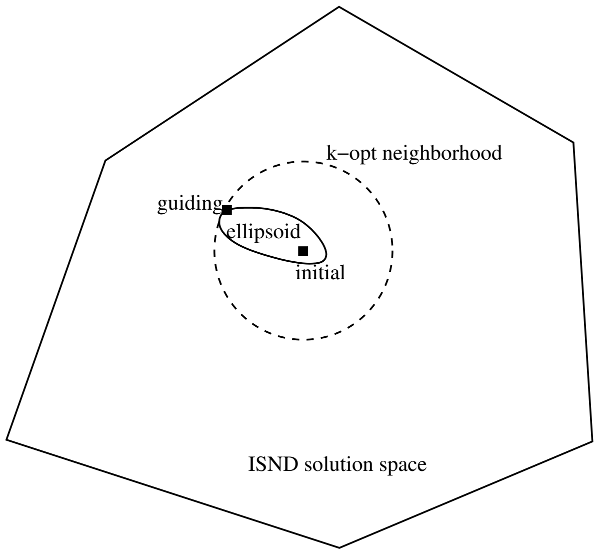

Figure 6.11: Illustration of Ellipsoid within k-opt Neighborhood

Variable selection elaborately builds a block variable set. Denote $\mathcal{B}_{ini}$ the set of open blocks in the initial solution and $\mathcal{B}_{gui}$ the set of open blocks in the guiding solution. An intuitive proposal is to define the variable set as the union of open blocks in both solutions, $\mathcal{B}_{ini} \cup \mathcal{B}_{gui}$. However, one may want to further improve the variable set. There are trade-offs: on one hand, one would limit the variable set to improve the efficiency because removing one block from the variable set leads to the deduction of a number of flow variables; on the other hand, without missing the optimum, one would like to include all blocks with promising characteristics.

To enrich the variable set, we explore the long-term memories. That is, a block not in $\mathcal{B}_{ini} \cup \mathcal{B}_{gui}$ also has its chance to be considered in the elite problem, if it shows steadily in the previous slope scaling iterations, which is regulated by long-term memories, $\bar{x}_b$ (average flow on block b) and $\hat{x}_b$ (maximal flow on b),

$$\bar{x}_b / \hat{x}_b > \lambda \tag{6.11}$$

where λ is a parameter. Only blocks satisfying inequality (6.11) are included in the elite problem.

An alternative for variable selection is to regulate the size of the block variable set. Other than the block variables in $\mathcal{B}_{int} \cup \mathcal{B}_{gui}$, vacancies in the variable set are filled with blocks following the sequence indicated by $\bar{x}_b/\hat{x}_b$. Blocks bearing steady flows are filled first, until the variable set reaches the predefined cardinality. The variable set size is defined with parameter φ,

$$|\mathcal{B}_{gui}| \times \varphi \tag{6.12}$$

The two variable selection variants will be compared in Section 7.2. As a result, in the elite problem, only a rather small portion of block variables picked by one of the regulations above is considered.

Variable fixing ensures the searching direction from the initial towards the guiding solution, and accelerates the solving procedure of elite problems. Among the selected block variables, blocks open in both initial and guiding solutions (in set $\mathcal{B}_{int} \cap \mathcal{B}_{gui}$) are fixed to open ($= 1$) in the elite problem. Furthermore, the common flows from both reference solutions are also fixed.

Because the slope scaling process actually addresses the R1-ISND problem, and the ellipsoid search intensifies the ISND problem directly, the trajectory included by ellipsoid search is not a superposition with slope scaling.

6.2.4 Overall Procedure

The overall procedure incorporates the ellipsoidal search (ES) with SS+LMP, and the SS+ES+LMP is indicated in Figure 6.12.

The overall procedure is organized into many phases, which are separated by factor perturbations. In each phase, a sequence of slope scaling iterations find a local best of the R1-ISND, and sliding problems are solved to produce feasible solutions of the ISND. If the new solutions contribute to the reference set, ellipsoidal search is implemented before factor perturbation is triggered. The procedure repeats until either we meet the maximal solving time or the maximal number of iterations of slope scaling.

The SS+ES+LMP approach presented mixes up several heuristic ideas, and takes advantage from both slope scaling converging speed on large instances and efficient per-

1. Set initial values for linearization factors α, β.
2. Until factor perturbation condition is satisfied, repeat,
    - generate an approximation problem AP1(α, β);
    - solve the AP1(α, β) with the augmenting path heuristic;
    - update factor α, β according to the traffic distribution.
3. Solve sliding problems to generate a series of feasible solutions.
4. If *reference solution set* is updated, implement *ellipsoidal search*.
5. If stopping criteria are met, stop.
6. Disturb linearization factors α, β based on long-term memories, go to 2.

Figure 6.12: Slope Scaling + Ellipsoidal Search + Long-term Memory Perturbation

formance of MIP solvers on small ones. On one hand, compared with Crainic et al. [44] and Kim et al. [79], the defect of slope scaling method is further amended by the proposed ellipsoidal search algorithm. On the other hand, compared with Hewitt et al. [65] and Rodríguez-Martín and Salazar-González [104] who constantly "push" the search with a MIP solver from the current best, our method only shapes an ellipsoid space whenever slope scaling returns good feasible solutions, then MIP solvers are applied to find "linking" between reference solutions. In addition, while applying MIP solvers to explore a local area, ellipsoidal search is more directionally conducted (from the initial to the guiding solution), and accelerates the procedure as common open variables are fixed.

6.3 A Hybrid Heuristic

We observe, due to the fact that each block consists of a series of service sections, in each ISND instance there exist much more block variables than service variables, and the very large block variable set expands the instance size dramatically together with the traffic flows on each block. Any application of the ISND model to a very-large-scale rail company leads to a huge number of variables. Finding the optimal solution, even a near-optimal solution for such problems is very difficult.

Given the path property of blocks, block variables can be considered implicitly rather than explicitly. In this section, we assume that only transportation demands and potential services are available in the problem setting, and we solve the ISND by dynamically generating blocks during the solving procedure.

The methodology is based on an innovative slope scaling scheme, where only the decision variables in the service layer are linearized in the approximation problem. By preserving the blocking decision variables, the approximation problem shows a network design formulation. Tabu search heuristic is applied to solve the approximation problem. The new slope scaling scheme also allows integrating column generation idea to address large instances. We propose a heuristic column generation procedure, denoted *block generation*, which is implemented in the tabu search process at each time the cycle-based neighborhood is evaluated.

The overall procedure thus performs in a double looping organization: in the outer loop, slope scaling improves the ISND solution together with ellipsoidal search, and inner loop tackles approximation problems by tabu search, while providing interesting block variables with block generation. The outer loop is guided by long-term memories, and the inner loop is conducted by tabu memories. The major algorithmic components in both outer loop and inner loop are detailed as below.

6.3.1 Outer Loop

In Section 6.2, we applied slope scaling to solve the ISND by linearizing all decision variables in the approximation problem, and the approximation problem presents a network flow formulation. However, by doing so, one loses the track of block decisions in the approximation problem. In order to preserve the block variables and eventually generate blocks dynamically, we propose a new slope scaling scheme where only service design costs are approximated.

Given a feasible flow distribution $\tilde{x}$, a service s should be open if the workload on the service is positive. Say $\sum_{p \in \mathcal{P}} \tilde{x}_{sp} > 0$, $\tilde{z}_s = 1$. When a service is open, the fixed service cost c_s^{F} must be paid. We approximate the fixed service costs with a vector of linearization factor β, each defined on a service s. To describe the fixed cost on a service, β_s is adjusted according to equation (6.13) if service s is open in $\tilde{z}$.

$$\sum_{p \in \mathcal{P}} \beta_s \tilde{x}_{sp} = c_s^{\mathrm{F}} \tilde{z}_s \tag{6.13}$$

An Approximation Problem with linearization factor β is thus formulated as follows.

$$\mathrm{AP2}(\beta) = \min \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}} c_{ap} x_{ap} + \sum_{b \in \mathcal{B}} c_b^{\mathrm{F}} y_b + \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{P}} \beta_s x_{sp} \tag{6.14}$$

subject to traffic flow conservation constraint (5.2), block load constraint (5.3), car handling constraint (5.6), block building constraint (5.7), as well as train load capacity (6.6). To distinguish with the AP1 problem in Section 6.2, we denote this approximation problem AP2.

The ISND objective (5.1) and AP2 objective (6.14) differ only at the last component, where the service design costs are approximated by β. Three main observations emerge according to the AP2 definition. First, AP2 also presents a FMCND formulation. Define a surrogate flow cost for each service section $c'_{lp} = \sum_{a \in \mathcal{A}^{\mathrm{SM}}(l)} (c_{ap} + \beta_s) + \sum_{a \in \mathcal{A}^{\mathrm{SR}}(l)} c_{ap}$. Each block is associated with a fixed cost c_b^{F} and a surrogate flow cost c'_{bp} for each traffic p. c'_{bp} is the sum of flow costs on the block path, where each component service section has flow cost c'_{lp}. Therefore, AP2 can be translated into a block network design formulation, and the objective (6.14) can be rewritten as,

$$\mathrm{AP2}(\beta) = \min \sum_{p \in \mathcal{P}} \sum_{a \in \mathcal{A}^{\mathrm{CW}} \cup \mathcal{A}^{\mathrm{CC}} \cup \mathcal{A}^{\mathrm{CH}}} c_{ap} x_{ap} + \sum_{p \in \mathcal{P}} \sum_{b \in \mathcal{B}} c'_{bp} x_{bp} + \sum_{b \in \mathcal{B}} c_b^{\mathrm{F}} y_b$$

Second, without service decisions, train running constraint (5.8) is omitted, and AP2 is actually the approximation of a relaxation of ISND. We define R2-ISND as an ISND without constraint (5.8). Third, any AP2 solution contains a flow pattern $\tilde{x}$ and a block design $\tilde{y}$, which can be intuitively converted to a R2-ISND solution by opening only services used by the open blocks in $\tilde{y}$. That is, in the case one of the service sections of

s is used by b and $\tilde{y}_b = 1$, $\tilde{z}_s = 1$; $\tilde{z}_s = 0$ on the contrary.

Denote $\beta(i)$ the linearization factor setting at slope scaling iteration i. When AP2 with linearization factor $\beta(i)$ is solved with the inner loop (to be explained in Section 6.3.2), an AP2 solution $\tilde{x}$ and $\tilde{y}$ is obtained. A solution of R2-ISND with service design $\tilde{z}$ is thus achieved by opening only the services employed. Linearization factors β are then modified to approximate the fixed costs for providing services in $\tilde{z}$. According to condition (6.13), linearization factors β are updated following the regulation (6.15) if service $\tilde{z}_s = 1$, and remain the same otherwise.

$$\beta_s = c_s^{\mathrm{F}} / \sum_{p \in \mathcal{P}} \tilde{x}_{sp} \tag{6.15}$$

Slope scaling repeats the above progress to improve the R2-ISND objective, until a local optimum of R2-ISND is reached. Without constraint (5.8), a R2-ISND solution might violate the train running capacity in ISND. To retrieve the feasibility of ISND, some open services and blocks are slid along the time horizon. A *sliding problem* (defined as same as in Section 6.2.1) is solved by MIP solver to produce feasible solutions for the ISND.

To overleap from a local domain, and restart in a different area, slope scaling is perturbed by disturbing linearization factors. A similar perturbation method as Section 6.2.2 is applied. Long-term memories record the average workload, maximal workload, and opening frequency of each service. Two same strategies for perturbations, intensification and diversification are applied. Intensification incents the car flow on services with high opening frequency by decreasing the corresponding linearization factors, and diversification attracts more cars on services that seldom been used. In each perturbation phase, either intensification or a diversification is performed by disturbing linearization factors according to long-term memories. The decision on perturbation strategy is conducted by parameters addressing the upper bound of continuous intensification/diversification.

Slope scaling process is therefore divided into phases by perturbations. Each phase solves a sequence of AP2 until we reach a local optimum of R2-ISND, then sliding problems are solved to generate feasible solutions of ISND, which are saved in a reference solution set. Ellipsoidal search is applied to combine the attracting characteristics of good solutions in the reference set, and further improve the solution quality.

The outer loop procedure is summarized in Figure 6.13.

1. Set initial values for linearization factors β.
2. While outer loop stopping criteria are not met,
    - generate an approximation problem AP2(β);
    - *inner loop* solves AP2(β);
    - if factor perturbation criteria are met
        - generate ISND solutions by solving *sliding problems*,
        - *ellipsoidal search*,
        - *factor perturbation* by *long-term memories*,
    - else
        - update linearization factor β.
3. Stop.

Figure 6.13: Outer Loop Procedure

From the initial setting of linearization factors $\beta(i = 0)$, slope scaling iteratively improves the R2-ISND solution. Factor perturbation criteria are set as either we find identical block designs in consecutive iterations or reaching the maximum number of iterations without improvement on the R2-ISND solution. If one of the above criteria is met, a series of sliding problems are solved until there is no progress, and ellipsoidal search is implemented to improve the ISND solution. Linearization factors are then perturbed and slope scaling restarts if the stopping criteria are not met, which is generally indicated by the maximal solving time.

6.3.2 Inner Loop

Each time linearization factors β are updated, inner loop is engaged to solve the AP2(β). AP2 presents a complicated block network design formulation, and we intent to solve AP2 with dynamic generation of block variables in order to address very-large-scale instances. The general column generation with the use of dual variables provides an intuitive option, however, the method is insufficient for AP2. This is because the additional attributes associated with blocks. Recall the block definition, the flow cost of a block is the sum of the flow costs on its component service sections, transfer links and transfer delay links. The fixed cost of a block is determined by its origin yard (classification track occupancy cost c^{F}_{bo}) and intermediate transfers (fixed swap cost c^{F}_{ba}), and by assumption, capacity u_b and classification track occupancy time $h(b)$ are decided by the block origin. Apparently, the attributes of a block is determined when the block path in block-layer projection is fixed. However, as transfers links contribute to both block flow costs and fixed blocking cost, it is impossible to prove the lowest-cost block path before we decide the flow on block.

The reason we choose cycle-based tabu search heuristic to solve AP2 is twofold. First, according to one of the observations from the previous research: the efficiency of slope scaling relies on the performance of the approximation problem solver. Even though AP2 are solved heuristically, the short AP2 solving time enables the overall procedure to accommodate more slope scaling iterations, and eventually improve the solution quality. Second, unlike the link-based neighborhood where some links in the network are maneuvered (opened or closed) and flows are redistributed accordingly, cycle-based neighborhood deviates flows and links are opened or closed corresponding to the new flow pattern. Aiming at flows, a residual value (car flow volume to be deviated in the neighborhood) is selected before each move evaluation in the cycle-based neighborhood. With a selected residual value, we are able to address the fixed costs associated with blocks, and it facilities the block generation.

In Section 6.1, we applied the cycle-based neighborhood and developed a tabu search heuristic to solve the DSND problem with non-stop services. Without the service design variables, each AP2 here can be viewed as a special DSND case where fixed service costs are all set to 0. A similar tabu search algorithm is applied to solve each AP2. Additional modifications are made to adapt in the slope scaling scheme. First, the eligible block design definition is a little different as train running constraints do not present in AP2.

Second, as in the consecutive slope scaling iterations, only a portion of linearization factors are updated, so that only the surrogate flow cost c'_{bp} on some blocks are changed and block fixed costs remain the same. That implies, AP2($\beta(i-1)$) and AP2($\beta(i)$) are kind of similar. As we are solving a series of similar AP2, there is no need to solve each AP2 from scratch, and we use the best solution of AP2($\beta(i-1)$) as the initial solution of AP2($\beta(i)$). Third, we incorporate the block generation procedure to generate more interesting blocks during the tabu search.

In the 3-layer time-space network, blocks show a link-path duality: from the car layer perspective, each block is a link connecting an OUT node to an IN node; from the block layer perspective, each block is a path. The link nature causes it impractical to price blocks implicitly in the car-layer projection. Therefore, when generating blocks, we work on the block-layer projection, which consists of service sections, transfer links, and transfer delay links.

The journey of a block is described by a path from an OUT node to an IN node in the block-layer projection. Define *eligible* OUT→IN node couple in the block-layer projection, where OUT and IN are from different yards and at different time point. Given an eligible node couple, we might have multiple paths from the OUT to the IN node consisting of a series of service sections connected by transfer links and transfer delay links. We define a *kernel block set* for each eligible node couple, where all blocks connecting the origin OUT node to the destination IN node are collected.

During the tabu search procedure, a γ-residual network is constructed with two types of arcs: forward arcs and backward arcs. A forward arc represents allocating γ more flow on the car-layer link/block, and backward arc depicts that γ unit flow is withdrew from the link/block. Residual arcs are derived from either blocks or car-layer links. When constructing a γ-residual network, we focus on the residual arcs from blocks as car-layer links are directly assessable. Given an eligible node couple, say (i,j), where i is the OUT node, and j is the IN node. Each block in the kernel block set may supply a forward arc $(i,j)^+$ and a backward arc $(j,i)^-$ for the γ-residual network. That is, in the γ-residual network, we might have multiple forward and backward arcs connecting the node couple.

After the γ-residual network is constructed, the backward arc where γ derives is first selected. We then determine a cycle with the lowest cost, which is made of the selected backward arc, together with a shortest path in the residual network connecting the target to the source of the backward arc. The shortest path consists of a series of residual arcs, either forward or backward. Now suppose the shortest path uses a residual arc between an eligible node couple (i, j) in the γ residual network, If a residual arcs from j to i is used (if there exists any), it must be the backward arc with the lowest cost among all $(j, i)^-$. If a forward residual arc from i to j is used, only the forward arc $(i, j)^+$ with the lowest cost needs to be considered. As backward arcs only derive from open blocks which are already in consideration, our interest exists in the forward residual arc with the lowest residual cost connecting the OUT→IN node couple. That is, we are interested in finding in each kernel block set the block with the capacity to carry extra γ flow, suggesting the lowest-cost forward arc.

Finding such blocks equals to finding lowest-cost paths for each eligible node couple in the block-layer projection. Equipped with a residual value γ provided by the cycle-based neighborhood, we are able to linearize the fixed cost contribution from transfer links. A conceptual (β, γ)-*block layer* is constructed, where we have service sections (only the ones with at least γ residual capacity), transfer links and transfer delay links, each with a customized flow cost. The customized flow costs are regulated as below. Average cost definitions are applied given the γ residual value from tabu search are considered as unified flow. (6.16) gives the surrogate flow cost on each service section, which is the average flow cost on service sections with linearization factor effects. Customized transfer link cost is presented in (6.17), where the block fixed cost component is linearized by γ. Average transfer delay cost is shown in (6.18).

$$c'_l = \sum_{a \in \mathcal{A}^{\text{SM}}(l)} \big(\frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} + \beta_{s(l)}\big) + \sum_{a \in \mathcal{A}^{\text{SR}}(l)} \frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} \qquad \forall l \in \mathcal{L}; \tag{6.16}$$

$$c'_a = \frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} + \frac{c^F_{ba}}{\gamma} \qquad \forall a \in \mathcal{A}^{\text{BT}}; \tag{6.17}$$

$$c'_a = \frac{\sum_{p \in \mathcal{P}} c_{ap}}{|\mathcal{P}|} \qquad \forall a \in \mathcal{A}^{\text{BH}}. \tag{6.18}$$

On the (β, γ)-block layer, we find the shortest (most cost efficient) path for every eligible OUT→IN node couple. The extended Bellman-Ford algorithm proposed in Section 6.1 is applied, which is further modified to ensure the shortest path generated respects the block definition, that is, each block starts and ends with service sections. To achieve this, we first find shortest paths from the origin OUT node (say node i) to an OUT nodes (say node q). According to the network structure, such paths either stop at the origin OUT node $q = i$ or end with a transfer link. These paths are then attached with services sections (q, j) departing from the OUT node q, and the distance (cost) at the final IN node j is updated and labeled.

When a new block path is generated, a block is obtained with a flow cost c_{bp} (sum of the flow cost on component links), a fixed cost c_b^{F} (sum of c_{bo}^{F} from the origin yard and c_{ba}^{F} from transfer links passed), a capacity u_b and a building time $h(b)$ (from origin yard). Such a block leads to a forward arc which is able to carry γ extra flow from the origin OUT node to the destination IN node with the lowest residual cost, and will be considered when we build cycles in the γ-residual network.

Our block generation process has similarities and differences with the general column generation process. The AP2(β) with current block set serves the restricted master problem, and the shortest path algorithm acts as the pricing function to select the interesting blocks to be enclosed. However, compared with the column generation applied on network flow problems where paths are selected according to dual variable information and then flows are re-distributed, we preselect a flow amount γ, linearize the flow cost on each link, generate more paths (blocks), and feed them into tabu search to further modify the flows.

The tabu search is guarded from vain looping by tabu memories where tabu tenures are saved, however, when solving AP2, tabu tenures are only associated with blocks. When the status of a block has been changed (opened or closed), a tabu tenure is randomly picked in a predefined integer range and associated to the block. After each move, the tabu tenure is updated by -1 if positive. Another functionality of tabu memory is to prevent the search from infeasible domains. Each time a minimum-cost cycle is calculated, the block building constraint is verified first. If the cycle is leading to a neighbor block design where block building constraints are violated, relevant blocks are sent to the tabu memory, and we restart to calculate next non-tabu cycle.

Same as the previous tabu search, processes have been applied to direct the search into/out of specific searching areas. In the case a new best AP2(β) solution is found, one would like to look deeper in the area and try to locate other good solutions. Cycles of each demand are evaluated with an assumption that all flows from other traffics are fixed. In the case there is no or just minor improvements in the last move, it is very possible that the tabu search is restricted in a local area. To diversify the tabu search, long-term frequency memories from outer loop are explored, and open blocks hauled by the frequently-opened services are forced to close.

The inner loop procedure is recapitulated in Figure 6.14. The best solution of the last AP2($\beta(i-1)$) provides the starting solution for AP2($\beta(i)$). From the solution, a sequence of moves in the cycle-based neighborhood thus improves the AP2(β) solution, and each time we evaluate the neighborhood with a residual value γ, on the (β, γ)-block layer, blocks are generated to enhance the block set. Inner loop stopping criteria are indicated by the maximal number of tabu search moves or maximal number of continuous non-improved moves according to the AP2(β) solution.

6.3.3 Overall Procedure

We denote the new hybrid algorithm the BEST method as Block generation, Ellipsoidal search, Slope scaling and Tabu search are integrated in the compound heuristic. The overall BEST procedure starts with an initialization phase, where an initial set of blocks $\mathcal{B}'$ is generated by building a block on each service section. That is, only "direct" blocks are generated. For those blocks, the block flow cost equals to the flow cost on the service section used, the block fixed cost and block load capacity come from the block origin. The SS+ES+LMP heuristic is applied to provide an initial ISND solution on the given set of blocks. Preliminary linearization factor $\beta(0)$ in BEST are then set to approximate the service fixed costs from the initial ISND solution, and the rest of the initial solution (flow distribution and block design) is adopted as the starting solution of AP2($\beta(0)$). The algorithm then starts the outer loop to improve the ISND solution,

1. Given AP2(β), and known block set $\mathcal{B}'$.
2. While inner loop stopping criteria are not met,
   - reset *minCycle*;
   - for each open block b in *curSol*,
     - calculate the flow value γ on b,
     - generate blocks for each eligible OUT$\rightarrow$IN node couple,
     - include new blocks into set $\mathcal{B}'$,
     - construct a γ-residual network,
     - find the lowest-cost cycle starting from the backward arc representing block b,
     - update the *minCycle*;
   - deviate flow following the *minCycle*, and change the block design;
   - distribute traffic on the new block design.
3. Stop.

Figure 6.14: Inner Loop Procedure

where each AP2 is solved with inner loops.

Compared with the SS+ES+LMP heuristic, BEST heuristic presents a quite algorithmic difference. We propose a new slope scaling scheme within BEST, which basically provides a decomposition of the 3-layer structure: one block network design problem in the lower two layers (car-layer projection) and one block generation problem in the upper two layers (block-layer projection). Keeping the exact block design costs in the objective, AP2 in the lower layers presents a better approximation for the original ISND than AP1 in SS+ES+LMP. However, by preserving the block decisions, AP2 shows a FMCND formulation. Comparing with the network flow formulation of AP1, the complex network design formulation of AP2 generally leads to a longer computing time even we solve them heuristically, but the trade-off is we are now working on smaller AP2 with much less variables than AP1.

CHAPTER 7

COMPUTATIONAL RESULTS AND DISCUSSION

The performance of the proposed algorithms is analyzed in this chapter. According to different models, we first generate sets of random instances, and calibrate the main parameters and procedures conducting the solving process. Algorithm efficiency is then tested on simulant instances with the overall best parameter setting.

To evaluate the performance, algorithms are compared with the standard branch-and-bound algorithm offered by a well-known mathematical package, CPLEX v10, which is also used as the MIP solver embedded. All programs are coded in C++, and tests are implemented on computers with 2.4Hz CPU and 16 GB of RAM, operating under Linux.

The required input of the DSND and ISND model includes the physical characteristics, demand pattern, as well as potential services and blocks. Deriving from the rail system, physical network features and demand pattern are straightforward, and can be easily accessed. To enumerate candidate services, service routes are picked and combined with possible speeds and stops. The complete service list is obtained by departing all promising route-stop-speed combinations at each time point in the planning horizon.

Service sections from the service list are connected by transfer links and transfer delay links in the block layer to generate potential blocks. During the generation procedure, additional restrictions can be integrated. Such restrictions come from equipment availability, environment, operating time, transportation laws, human resources, etc. For example, the blocks on over-zigzag routes are shunned; temporal (e.g. holiday) regulations of yards. These *application constraints* help us capture the essential attributes of the real data, and effectively prune the irrational blocks.

The instances tested are randomly generated. Even with the same physical magnitude and the same time horizon, random instances differ greatly in the number of blocks and services. Moreover, various constraints on different operations with diverse resources make it impossible to unify the compact/loose concept between all capacities and flows in the multi-layer time-space network. The relative ratio between fixed cost and flow cost is also hard to specify because we have fixed costs from both blocks and services. As a result, we leave the variance of the instances to randomicity, and for each combination of physical magnitude and time horizon, instances are generated with increasing demand numbers to roughly partition the car flow density.

7.1 Results for Tabu Search

3 sets of direct service instances are generated to test the tabu search algorithm. Set D-S includes the small instances with only 4 to 5 yards and a time horizon with 10 time periods. Set D-M is the medium set, and the instances in set D-M have 7 yards with either 10 or 14 time periods. In set D-L, the instances vary from 7 to 10 yards with 14 time periods, which are very large and difficult to solve.

Instance parameters are shown in Table 7.I, 7.II, and 7.III, where the columns hold the instance name, number of potential blocks, number of possible direct services, number of yards, number of tracks, time dimension and demand number. Three tables show that the appended time dimension and the 2-layer structure multiple the network scale. Even for the smallest instance with only 4 yards, we generate hundreds of blocks and end up with more than $10,000$ variables. Moreover, with the increase of the number of yards and tracks, as well as the length of the time horizon, the size of the instances increases in an exponential fashion. For large instances with 10 yards and 14 time periods, we easily produce over $120,000$ blocks, and over 100 million flow variables.

Inst	Block	D-Serv	Yard	Track	Time	Demand
ds01	1340	400	4	10	10	40
ds02	720	280	4	10	10	80
ds03	660	280	4	10	10	120
ds04	1780	560	5	14	10	100
ds05	1420	440	5	14	10	150
ds06	1140	350	5	14	10	200
ds07	1790	680	5	18	10	50
ds08	1050	500	5	18	10	100
ds09	1420	620	5	18	10	150

Table 7.I: Instance Parameters: Set D-S

Inst	Block	D-Serv	Yard	Track	Time	Demand
dm01	9310	2140	7	20	10	150
dm02	5760	1440	7	20	10	200
dm03	6740	1240	7	20	10	250
dm04	6240	2300	7	32	10	200
dm05	2970	1230	7	32	10	250
dm06	5730	2040	7	32	10	300
dm07	29092	3080	7	20	14	250
dm08	80374	3136	7	20	14	300
dm09	19824	2212	7	20	14	350

Table 7.II: Instance Parameters: Set D-M

Inst	Block	D-Serv	Yard	Track	Time	Demand
dl01	35014	6356	7	32	14	450
dl02	9212	3556	7	32	14	500
dl03	17570	3528	7	32	14	400
dl04	99946	15274	10	60	14	400
dl05	126490	18872	10	60	14	500
dl06	116494	13356	10	60	14	600
dl07	39172	16464	10	60	14	700
dl08	81298	23072	10	60	14	800
dl09	124488	22358	10	60	14	900

Table 7.III: Instance Parameters: Set D-L

7.1.1 Parameter Calibration

The major parameters guiding the tabu search algorithm are calibrated here.

- Tabu tenure is the most important parameter in a tabu search. A proper setting of tabu tenure prevents the search from cycling without constraining it too much. Each time a tabu status is updated, either in block tabu list or service tabu list, a random tenure is selected in the interval and is associated to the according block or direct service. Two intervals, $[7, 15]$ and $[10, 20]$ are proposed.

- To conduct the intensification phase, a threshold is used to determine the good solutions. When the gap between the current solution and the best solution so far is less than the threshold, the intensification phase is triggered. Two values, 5% and 7%, are compared.

- The diversification percentage figures out the variance range in each diversification phase, and manages the searching scope of the tabu search. Either the 8%, 10% or 15% top-frequency blocks/services are closed in the tabu search diversification phase.

A total of 12 parameter setting combinations are thus examined. Calibration experiments are implemented on 9 instances (3 instances randomly picked from each instance set), which are each solved with all parameter settings. The calibration experiments end when we reach the 20 continuous non-improved local searches. We rank the parameter combinations for each instance according to both solution quality and solution time. A score of 10 to 1 is assigned to each of the first 10 places on the basis of solution quality and solution time respectively. The better the solution, the higher the solution score; the longer the solution time, the lower the time score. The scores on each problem are summed up and displayed in Table 7.IV, where the first three columns present the setting, the two columns in the middle give the solution score and time score, and the last column reports the total score.

TabuTenure	IntensThreshold	DiversRatio	Sol. Score	Time Score	Ttl Score
[7,15]	5%	8%	47	43	90
[7,15]	5%	10%	41	51	92
[7,15]	5%	15%	64	30	94
[7,15]	7%	8%	46	39	85
[7,15]	7%	10%	35	48	83
[7,15]	7%	15%	59	27	86
[10,20]	5%	8%	48	40	88
[10,20]	5%	10%	35	32	67
[10,20]	5%	15%	49	47	96
[10,20]	7%	8%	35	41	76
[10,20]	7%	10%	42	45	87
[10,20]	7%	15%	57	35	92

Table 7.IV: Tabu Search Parameter Calibration

Statistics in Table 7.IV show that setting ($[7, 15]$, 5%, 15%) gives the best solutions, and setting ($[7, 15]$, 5%, 10%) leads to the shortest solution times. Parameter setting with $TabuTenure = [10, 20]$, $IntensThreshold = 5\%$ and $DiversRatio = 15\%$

achieves the highest total score, and balances the trade-offs between solution quality and efficiency. The tabu search experiments in the following are conducted with this setting.

7.1.2 Result Analysis

The proposed tabu search algorithm is tested on the instances in set D-S, D-M and D-L respectively.

For comparison purposes, the instances are first solved by CPLEX. The CPLEX solver stops either when the optimum is found or when we reach the maximum CPU time (t = 10 hours = $36,000$ sec).

Numerical results of the 3 instance sets are reported in Table 7.V, 7.VI, 7.VII respectively. Comparison tables are divided into 3 groups. The first group is the CPLEX solution, including the best solution, solution time (in second), and the optimal gap. The second group is the tabu search solution, which terminates at 20 non-improved local searches. Tabu search solution, solution time, solution gap with the CPLEX solution, and number of local search iterations are reported. Longer runs are preformed to further illustrate and analyze the heuristic's behavior. When the tabu search program stops at a maximum of 300 local searches, the results are shown in the third group. Same as the CPLEX solution, tabu search always ends after reaching the maximum CPU time. All results are rounded to integer for the sake of display.

The conclusion that emerges from the first instance set is that for small instances, the proposed tabu search algorithm is proven to be efficient to find optimal or near-optimal solutions within a short time. Compared with CPLEX, which solves all the instances, our tabu search solution (stop at 20 non-improved iterations) achieves the average optimal gap 0.81% within an average 40% of solution time.

The results displayed in Table 7.VI prove the robust performance on medium instances. Where CPLEX fails to prove the optimality, tabu search finds good solutions which are close to the best solution yielded by CPLEX, yet within a much shorter solution time. Moreover, with the increase of the instance, tabu search results approach and start to catch up with the best CPLEX solution. When the optimal gap is considerable, we have a better opportunity to outperform CPLEX. On set D-M, the mean CPLEX gap

Inst	CplSol	time(s)	OptGap	TS.Sol	time(s)	CplGap	Iter	TS.(300i)	time(s)	CplGap	Iter
ds01	33670	100	0.00%	33670	25	0.00%	26	33670	497	0.00%	300
ds02	57412	108	0.00%	57631	176	0.38%	39	57630	2401	0.38%	300
ds03	88099	282	0.00%	88099	124	0.00%	21	88099	6225	0.00%	300
ds04	87435	2037	0.00%	87934	492	0.57%	23	87619	7669	0.21%	300
ds05	117000	7819	0.00%	118975	776	1.69%	28	117810	10583	0.69%	300
ds06	147273	11057	0.00%	149190	621	1.30%	22	149190	t	1.30%	268
ds07	46944	279	0.00%	47537	117	1.26%	28	46944	1046	0.00%	300
ds08	92601	2602	0.00%	92907	325	0.33%	21	92906	4884	0.33%	300
ds09	136493	1143	0.00%	138916	276	1.77%	26	138023	13056	1.12%	300
avg						0.81%				0.45%	

Table 7.V: Tabu Search Results on Instance Set D-S

is 0.27%, and on average our heuristic takes only 31% of the solution time as CPLEX does.

Inst	CplSol	time(s)	OptGap	TS.Sol	time(s)	CplGap	Iter	TS.(300i)	time(s)	CplGap	Iter
dm01	138357	t	3.17%	142825	5137	3.23%	39	142825	24616	3.23%	300
dm02	206363	t	1.57%	211229	7315	2.36%	58	209084	t	1.32%	143
dm03	205191	t	2.77%	206432	10590	0.60%	53	206432	t	0.60%	139
dm04	181202	t	3.57%	183983	9760	1.53%	56	183983	t	1.53%	143
dm05	197500	t	1.51%	202724	5677	2.64%	34	200089	t	1.31%	117
dm06	213495	t	1.48%	215003	6433	0.71%	27	214120	t	0.29%	98
dm07	208530	t	5.34%	215738	10505	3.46%	34	209485	t	0.46%	94
dm08	261751	t	18.18%	232060	26190	-11.34%	41	232060	t	-11.34%	58
dm09	286731	t	4.32%	284656	19429	-0.72%	51	284656	t	-0.72%	75
avg						0.27%				-0.36%	

Table 7.VI: Tabu Search Results on Instance Set D-M

As expected, the solutions are further improved when we extend the solution process up to 300 local searches. In a CPU time of 10 hours, we have identified higher-quality solution for 8 out of 16 instances in set D-S and D-M if they are not optimized already. The average optimal gap drops from 0.81% to 0.45% in set D-S, and in set D-M, the gap with CPLEX decreases from 0.27% to -0.36% (improvement).

Table 7.VII summarizes the behavior of the cycle-based tabu search on large instances, where our tabu search performs superiorly. The performance of CPLEX deteriorates fast with the increase of instances. With the same running time, we always find better solutions than CPLEX with prominent advancement. The average improvement is over 11%, and the maximal improvement reported reaches up to 14.18% (dl02). When CPLEX fails to provide any solution, indicated by $\times$ as in the last part of the table, feasible solutions are always presented by the tabu search algorithm. One notices, in most cases, we do not reach the 20 non-improved criteria and the procedure stops due to the maximal solving time, which is also an indication of the complexity of the DSND problem.

Inst	CplSol	time(s)	OptGap	TS.Sol	time(s)	CplGap	Iter	TS.(300i)	time(s)	CplGap	Iter
dl01	304782	t	12.67%	284529	t	-6.65%	38	284529	t	-6.65%	38
dl02	337940	t	24.40%	291499	11541	-13.74%	25	290016	t	-14.18%	61
dl03	282844	t	26.22%	244149	t	-13.68%	30	244149	t	-13.68%	30
dl04	393135	t	19.44%	360179	t	-8.38%	32	360179	t	-8.38%	32
dl05	456321	t	25.70%	397350	t	-12.92%	21	397350	t	-12.92%	21
dl06	$\times$	t	-	459216	t	-	13	459216	t	-	13
dl07	$\times$	t	-	555333	t	-	16	555333	t	-	16
dl08	$\times$	t	-	510350	t	-	13	510350	t	-	13
dl09	$\times$	t	-	639816	t	-	10	639816	t	-	10
avg						-11.07%				-11.16%	

Table 7.VII: Tabu Search Results on Instance Set D-L

After the tests on three instance sets, we conclude that the tabu search algorithm on the cycle-based neighborhood is able to yield very good solutions for the DSND model studied. Compared with CPLEX, the heuristic helps to locate the optimal solution for small-sized problems and near-optimal solutions for medium problems within a reasonable amount of computation time. More importantly, the algorithm provides good quality solutions for the large instances which are intractable for other methods.

7.2 Results for SS+ES+LMP

To test the SS+ES+LMP heuristic, 30 ISND instances are generated and divided into 2 sets. Both sets include instances with 5 to 10 yards and 14 to 60 tracks, while set C-S has a shorter time dimension with 7 time periods and set C-L has 10. Instance parameters are listed in Table 7.VIII and 7.IX. The two tables show that the multi-stop service instances in the 3-layer structure dramatically expand the physical network, and they generally present more potential services and blocks than the direct service instances. Even the smallest instance with only 5 yards, we obtain thousands of blocks and end up with hundreds of thousands of variables. On the instance with 10 yards and 10 time periods, we easily produce over 300 million flow variables.

Inst	Block	Service	Yard	Track	Time	Demand
cs01	1855	301	5	14	7	150
cs02	2765	266	5	14	7	200
cs03	2121	322	5	14	7	250
cs04	3241	259	5	18	7	250
cs05	1533	273	5	18	7	300
cs06	7497	413	5	18	7	350
cs07	39473	756	7	20	7	350
cs08	29449	693	7	20	7	400
cs09	19453	623	7	20	7	450
cs10	18424	840	7	32	7	450
cs11	9093	749	7	32	7	500
cs12	11102	700	7	32	7	550
cs13	140105	1834	10	60	7	600
cs14	236628	2016	10	60	7	700
cs15	279230	2674	10	60	7	800

Table 7.VIII: Instance Parameter, Set C-S

Inst	Block	Service	Yard	Track	Time	Demand
cl01	8900	550	5	14	10	100
cl02	4700	490	5	14	10	150
cl03	3600	500	5	14	10	200
cl04	1580	340	5	18	10	150
cl05	2570	500	5	18	10	200
cl06	8060	510	5	18	10	250
cl07	46150	1230	7	20	10	200
cl08	49920	1270	7	20	10	250
cl09	50000	1080	7	20	10	300
cl10	58930	1350	7	32	10	250
cl11	55580	1140	7	32	10	300
cl12	22900	1220	7	32	10	350
cl13	491150	3080	10	60	10	500
cl14	421370	3030	10	60	10	700
cl15	326550	3050	10	60	10	900

Table 7.IX: Instance Parameter, Set C-L

7.2.1 Parameter Calibration

In SS+ES+LMP, slope scaling, ellipsoidal search, and factor perturbation are all complex procedures managed by many parameters. For the sake of clarity, we first discuss the factor perturbation calibration and procedure options for slope scaling based on SS+LMP, and then calibrate ellipsoidal search on the chosen setting of SS and LMP.

7.2.1.1 Factor Perturbation Calibration

As indicated in the literatures, the perturbation phase has a great impact to the slope scaling results. The performance of SS+LMP depends on the appropriate calibration of parameters conducting the factor perturbation.

- Parameter $I_{max}^{nonimprove}$ regulates one of the entries of the perturbation phase, and concerns the local optimum of R1-ISND. A larger threshold means more efforts in polishing the local best and a lower value leads to a wider search domain but with a higher possibility of missing the local optimum. Three values, 8, 10 or 15 are compared.

- Parameters I_{max}^{diver} and I_{max}^{inten} indicate the maximum number of consecutive implementations of diversification and intensification strategy, respectively. Here we intent to balance the procedure and minimize vain repeats. The setting $I_{max}^{diver} = 1$, $I_{max}^{inten} = 1$ forces to perform one strategy to the other irrespective whether the current strategy improves the R1-ISND solution or not. Small values on both parameters provide rational tolerance for repeats and leave equal chances for both strategies. Two value pairs $(2, 2)$ and $(3, 3)$ are proposed.

- The range of perturbation is managed by parameters ω^+ and ω^-. In most cases, in a good solution of R1-ISND, only a small portion of blocks/services are opened, which causes the average opening frequency $\bar{n}_{\mathcal{B}}$ and $\bar{n}_{\mathcal{S}}$ very small. In order to keep the frontier δ^- and θ^- positive so that some blocks and services are considered as rarely used, we choose $\omega^- = 0$. Similarly, ω^+ should be ≤ 1 in order to keep δ^+ and θ^+ rational. In calibration, we compare two options $\omega^+ = 0$ and $\omega^+ = 1$.

All 12 parameter combinations are evaluated on another set of random instances, denoted instance set C-C (shown in Table I). A maximum 300 slope scaling iterations is applied as the stopping criteria. As the iteration number is generally proportional to the solution time, parameter combinations are weighted according to solution quality only. A point of 10 to 1 is assigned to each of the top 10 places. The better the solution quality, the higher the score. Scores of each setting are summed up and shown in Table 7.X, where the first three columns present the setting of $I_{max}^{nonimprove}$, $(I_{max}^{diver}, I_{max}^{inten})$ and ω^{+}, and the last column holds the score. In SS+LMP, both shortest augmenting path heuristic and Simplex method are used as AP1 solver (see next section for the detailed discussion), however, only scores from shortest augmenting path are reported here.

$I_{max}^{nonimprove}$	$(I_{max}^{diver}, I_{max}^{inten})$	ω^{+}	Score
8	(2, 2)	0	33
8	(2, 2)	1	49
8	(3, 3)	0	34
8	(3, 3)	1	46
10	(2, 2)	0	36
10	(2, 2)	1	67
10	(3, 3)	0	45
10	(3, 3)	1	49
15	(2, 2)	0	51
15	(2, 2)	1	61
15	(3, 3)	0	39
15	(3, 3)	1	59

Table 7.X: Perturbation Parameter Calibration

Table 7.X reveals that, setting $I_{max}^{nonimprove} = 8$ generally performs worse than the other two values, and $I_{max}^{nonimprove} = 15$ is on average better than $I_{max}^{nonimprove} = 10$ setting, but the difference is minor. For parameter pair $(I_{max}^{diver}, I_{max}^{inten})$, $(2, 2)$ and $(3, 3)$ are rather comparable. Setting $\omega^{+} = 1$ usually outperforms $\omega^{+} = 0$. One combination $I_{max}^{nonimprove} = 10$, $(I_{max}^{diver}, I_{max}^{inten}) = (2, 2)$ and $\omega^{+} = 1$ stands out with the highest rank, and conducts the overall best performance. In the following experiments, we disturb linearization factors with this setting.

7.2.1.2 Slope Scaling Calibration

In the slope scaling calibration, we fix the solver for approximation problems. As discussed before, we have two solver options for AP1, the Simplex method and the proposed shortest augmenting path heuristic. Both procedures are evaluated on instance set C-C. Experiments show that the augmenting path heuristic outperforms the Simplex method on all parameter settings. We here only include two figures displaying the evolutions of two variants (on instance cc06 with parameter setting $I_{max}^{nonimprove} = 8$, $(I_{max}^{diver}, I_{max}^{inten}) = (2, 2)$ and $\omega^+ = 1$). Figure 7.1 compares the R1-ISND evolutions with augmenting path heuristic and Simplex method, and Figure 7.2 differentiates the performance of two variants according to the ISND solutions generated. In each figure, the horizontal axis gives the solving time, and the vertical axis gives the solution value.

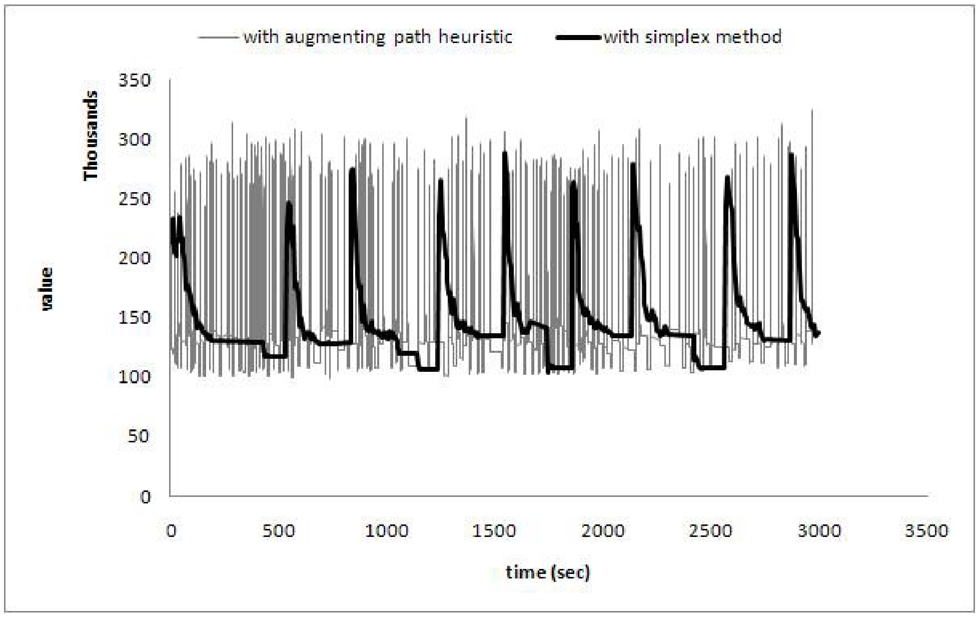

Figure 7.1: SS+LMP, Comparison of R1-ISND Evolutions

The first conclusion from Figure 7.1 is that of both variants, slope scaling converges really fast and the perturbation works well: intensifications force the search to find better solutions, and diversifications help to escape from local optima even though it initially deteriorates the solution significantly. The augmenting path heuristic is more efficient

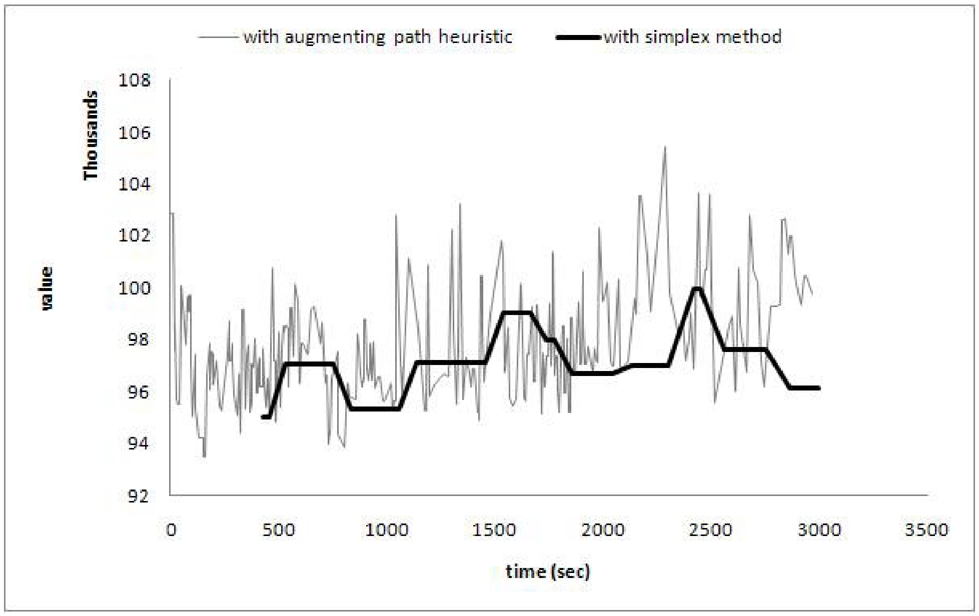

Figure 7.2: SS+LMP, Comparison of ISND Evolutions

to produce feasible AP1 solutions and enables more slope scaling iterations in a limited computing time. Traffic distribution obtained by the augmenting path heuristic is not necessarily optimal, but it offers more promising bases, and eventually generates better ISND solutions as shown in Figure 7.2. Hence, we choose the augmenting path heuristic as the AP1 solver.

7.2.1.3 Ellipsoidal Search Calibration

The success of the ellipsoidal search relies on the proper shape of the ellipsoid space. Two ellipsoid shaping strategies are compared. The first one, regulated by (6.11) is denoted as $s1$, and the other one, managed by (6.12) is $s2$.

- In strategy $s1$, λ marks the steady flow threshold. If λ is over small, the elite problem to be solved is large, and it takes a rather long time to converge. If λ is too big, we have a higher chance to miss the promising blocks in the elite problem. Three values, 15%, 25% and 50% are compared.

- Strategy $s2$ loads a fixed number of block variables into the ellipsoid, where φ

regulates the cardinality of the block set. Generally, we have $\varphi \geq 2$ in order to contain all blocks in $\mathcal{B}_{ini} \cup \mathcal{B}_{gui}$. Therefore, we choose 2 as the first value for φ. Two more values, 5 and 10 are also proposed.

Total 6 settings ($s1$ with $\lambda = 15\%, 25\%, 50\%$; $s2$ with $\varphi = 2, 5, 10$) are compared based on instances in set C-C. A same 10 to 1 rating system as above is adopted, and Table 7.XI displays the score of each setting. We observe, $s2$ usually outperforms $s1$. This is possibly because when with large instances, percentage threshold loses the control of ellipsoid scale, and causes over-sized elite problems. The setting $s2, \varphi = 5$ achieves the highest score, and is picked for the later experiments.

$s1, \lambda = 15\%$	$s1, \lambda = 25\%$	$s1, \lambda = 50\%$	$s2, \varphi = 2$	$s2, \varphi = 5$	$s2, \varphi = 10$
71	76	75	74	81	80

Table 7.XI: Ellipsoid Search Parameter Calibration

With this setting, only a rather small portion of blocks are selected. For example, in instance cc08, less than 200 block variables out of $75,000$ are picked in each elite problem.

7.2.2 Results Analysis

To analyze the behavior of both factor perturbation and ellipsoidal search, as well as the overall performance of the heuristic proposed, we organize experiments into two stages. First, the SS+LMP process is evaluated, and then the complete heuristic, SS+ES+LMP.

CPLEX results on ISND instances are first obtained. Table 7.XII displays the computational results of the instance set C-S. CPLEX solution, solution time, and optimal gap are shown in the first three columns. CPLEX solver terminates when either the optimum is found or reaching the maximal solving time (t= 10 hour). The first observation derives that except some small instances, CPLEX is unable to find an optimal solution within the time limit, which further demonstrates the complexity of the ISND problem. Moreover, with the increase of the instance, CPLEX soon loses its efficiency, and the limited performance leaves a considerable optimal gap. On large instances (e.g. cs13 with 10 yards and 60 tracks), an $\times$ indicates that CPLEX fails to identify any feasible solution due to the extreme complexity or exhausted memory.

Inst	CplSol	time(s)	OptGap	SS+LMP(1000i)	CplGap	time(s)	TimeGap
cs01	75157	329	0.00%	75667	0.68%	266	-19.22%
cs02	72471	t	2.69%	74228	2.43%	410	-98.86%
cs03	78629	816	0.00%	79616	1.25%	345	-57.74%
cs04	82784	t	0.41%	83462	0.82%	688	-98.09%
cs05	98705	828	0.00%	100173	1.49%	963	16.28%
cs06	110333	t	7.13%	109542	-0.72%	1042	-97.11%
cs07	267198	t	39.17%	187930	-29.67%	4154	-88.46%
cs08	230244	t	38.23%	171912	-25.34%	4005	-88.88%
cs09	174471	t	32.79%	135343	-22.43%	7830	-78.25%
cs10	176248	t	19.37%	161141	-8.57%	2891	-91.97%
cs11	155526	t	22.97%	140845	-9.44%	2782	-92.27%
cs12	183386	t	18.07%	169669	-7.48%	t	0.00%
cs13	$\times$	-	-	216927	-	t	-
cs14	$\times$	-	-	201368	-	t	-
cs15	$\times$	-	-	229553	-	t	-

Table 7.XII: SS+LMP Results on Instance Set C-S

The next four columns in Table 7.XII show the SS+LMP solution, solution gap with CPLEX solution, SS+LMP solving time as well as the time gap. A limit of $1,000$ slope scaling iterations is imposed, and for the purpose of comparison, the maximal solving time (10 hour) is also imposed. On small instances with 5 yards (cs01-cs06), SS+LMP approaches the CPLEX solution. An average 0.99% CPLEX gap is reached with an average 59.12% saving on computing time. Especially in the cases CPLEX proves the optimality (cs01, cs03, cs05), SS+LMP solution is rather close to the optimal solution. On instances with 7 yards (cs07-cs12), SS+LMP converges impressively, and outperforms CPLEX in both solution time and solution quality. The mean improvement on those instances is 17.16%, and it takes only an average 26.70% computing time as CPLEX does. The best performance is reported on instance cs07, compared with the CPLEX solution obtained in 10 hours, nearly 30% improvement is achieved by SS+LMP within less than 1.5 hour. Furthermore, where CPLEX fails to give any solution (cs13-cs15), the superiority of SS+LMP is emphasized by providing feasible solutions, even though it terminates at the maximal solving time without reaching $1,000$ iterations.

Results on instance set C-L are shown in Table 7.XIII. Instances in set C-L have a longer time dimension and are generally even larger. On these instances, SS+LMP displays a similar performance, and appears significantly more robust than the branch-and-bound method applied by CPLEX.

Inst	CplSol	time(s)	OptGap	SS+LMP(1000i)	CplGap	time(s)	TimeGap
cl01	94900	t	4.02%	96345	1.52%	126	-99.65%
cl02	90967	t	6.78%	92495	1.68%	203	-99.43%
cl03	128099	t	2.68%	130037	1.51%	304	-99.16%
cl04	79418	27534	0.00%	79566	0.19%	278	-98.99%
cl05	129305	1822	0.00%	131273	1.52%	343	-81.18%
cl06	105275	t	9.30%	105144	-0.12%	1006	-97.21%
cl07	225581	t	50.09%	146270	-35.16%	1399	-96.11%
cl08	169187	t	19.04%	165325	-2.28%	1734	-95.18%
cl09	293613	t	53.99%	178798	-39.10%	3350	-90.70%
cl10	208201	t	32.98%	163591	-21.43%	3162	-91.22%
cl11	187314	t	35.07%	149691	-20.09%	3317	-90.79%
cl12	226847	t	29.12%	187950	-17.15%	1900	-94.72%
cl13	$\times$	-	-	273842	-	t	-
cl14	$\times$	-	-	264326	-	t	-
cl15	$\times$	-	-	326381	-	t	-

Table 7.XIII: SS+LMP Results on Instance Set C-L

To further study the performance, we extend the SS+LMP solution time to 10 hours. The SS+LMP solutions and solution gap with CPLEX are reported in the first two columns in Table 7.XIV and Table 7.XV. In 16 out of 24 cases we achieved a better solution than SS+LMP(1000i) results with a longer computing time. Compared with CPLEX solutions with the same solution time, SS+LMP reaches an average 0.52% solution gap on instances with 5 yards (cs01-cs06, cl01-cl06), and an average 20.30% improvement on instances with 7 yards (cs07-cs12, cl07-cl12).

Ellipsoidal search is then incorporated in order to further improve the solution quality. Results of SS+ES+LMP are shown in the column indicated by SS+ES+LMP(10h). Column SS+LMP(10h)Gap and CplGap present the relative gap with SS+LMP solutions

Inst	SS+LMP(10h)	CplGap	SS+ES+LMP(10h)	SS+LMP(10h)Gap	CplGap
cs01	75347	0.25%	75157	-0.25%	0.00%
cs02	73446	1.35%	72483	-1.31%	0.02%
cs03	79202	0.73%	78731	-0.59%	0.13%
cs04	83462	0.82%	82784	-0.81%	0.00%
cs05	99651	0.96%	98705	-0.95%	0.00%
cs06	109502	0.75%	107576	-1.76%	-2.50%
cs07	187930	-29.67%	183527	-2.34%	-31.31%
cs08	170525	-25.94%	166283	-2.49%	-27.78%
cs09	134790	-22.74%	133110	-1.25%	-23.71%
cs10	161141	-8,57%	158486	-1.65%	-10.08%
cs11	136806	-12.04%	132723	-2.98%	-14.66%
cs12	169669	-7.48%	167424	-1.32%	-8.70%
cs13	216927	-	212204	-2.18%	-
cs14	201368	-	195208	-3.06%	-
cs15	229553	-	228770	-0.34%	-
Avg				-1.55%	

Table 7.XIV: SS+ES+LMP Results on Instance Set C-S

Inst	SS+LMP(10h)	CplGap	SS+ES+LMP(10h)	SS+LMP(10h)Gap	CplGap
cl01	94944	0.05%	94676	-0.28%	-0.24%
cl02	91432	0.51%	89554	-2.05%	-1.55%
cl03	129256	0.90%	128097	-0.90%	0.00%
cl04	79563	0.18%	79418	-0.19%	0.00%
cl05	130413	0.86%	129305	-0.85%	0.00%
cl06	104138	-1.08%	101656	-2.38%	-3.44%
cl07	146270	-35.16%	142283	-2.73%	-36.93%
cl08	165325	-2.28%	158157	-4.34%	-6.52%
cl09	178798	-39.10%	167934	-6.08%	-42.80%
cl10	160778	-22.78%	155919	-3.02%	-25.11%
cl11	149691	-20.09%	145052	-3.10%	-22.56%
cl12	186722	-17.69%	180356	-3.41%	-20.49%
cl13	273842	-	271156	-0.98%	-
cl14	264326	-	257230	-2.68%	-
cl15	326381	-	326575	0.06%	-
Avg				-2.20%	

Table 7.XV: SS+ES+LMP Results on Instance Set C-L

and CPLEX solutions with 10 hours computing time, respectively.

Results displayed in Table 7.XIV and Table 7.XV underscore the important role of the ellipsoidal search phase in the solution procedure. In almost all cases (29 instances out 30), SS+ES+LMP obtains better solutions than SS+LMP. The average improvement is reported as 1.55% in Table 7.XIV and 2.20% in Table 7.XV. The best performance is achieved on instance cl09 with over 6% improvement.

Another observation from the results shows the satisfactory behavior of the SS+ES+LMP procedure. On the 5 instances where CPLEX finds the optima, SS+ES+LMP achieves optima in 4 cases, and on the only one where we failed the optimal gap is rather small (0.13%). Statistics also show that on 28 out of 30 instances tested, the SS+ES+LMP heuristic finds solutions better than or equal to the CPLEX solution, with impressive improvements especially on large instances. The average CPLEX improvement is 0.63% on instances with 5 yards, and 22.56% on instances with 7 yards.

As a special case of ISND, DSND can also be tackled by the SS+ES+LMP heuristic. The performance of SS+ES+LMP and tabu search algorithm on D-S, D-M, D-L instances are compared in Table 7.XVI, 7.XVII, and 7.XVIII. Two experiments are implemented. The first 3 columns show the SS+ES+LMP solution in 10 hours, the comparison with tabu search results (TS(10h)Gap), and the comparison with CPLEX results (CplGap). The second experiment is conducted by the combination of SS+ES+LMP and tabu search, that is, SS+ES+LMP is first implemented for 300 iterations, and then tabu search continues with the best solution obtained. The results of the second experiment is shown in column SS+ES+LMP(300i)+TS.

Inst	SS+ES+LMP(10h)	TS(10h)Gap	CplGap	SS+ES+LMP(300i)+TS	TS(10h)Gap	CplGap
ds01	33670	0.00%	0.00%	33670	0.00%	0.00%
ds02	57412	-0.38%	0.00%	57631	0.00%	0.38%
ds03	88099	0.00%	0.00%	88099	0.00%	0.00%
ds04	87435	-0.21%	0.00%	87435	-0.21%	0.00%
ds05	117000	-0.69%	0.00%	117850	0.03%	0.73%
ds06	147273	-1.28%	0.00%	149071	-0.08%	1.22%
ds07	46944	0.00%	0.00%	47406	0.98%	0.98%
ds08	92664	-0.26%	0.07%	92664	-0.26%	0.07%
ds09	136493	-1.11%	0.00%	136602	-1.03%	0.08%
avg		-0.44%	0.01%		-0.06%	0.38%

Table 7.XVI: SS+ES+LMP Results on Instance Set D-S

The observation from the first experiment demonstrates that SS+ES+LMP is even

Inst	SS+ES+LMP(10h)	TS(10h)Gap	CplGap	SS+ES+LMP(300i)+TS	TS(10h)Gap	CplGap
dm01	139947	-2.02%	1.15%	140405	-1.69%	1.48%
dm02	206080	-1.44%	-0.14%	206586	-1.19%	0.11%
dm03	203484	-1.43%	-0.83%	204831	-0.78%	-0.18%
dm04	181104	-1.56%	-0.05%	182030	-1.06%	0.46%
dm05	196985	-1.55%	-0.26%	200101	0.01%	1.32%
dm06	213125	-0.46%	-0.17%	213923	-0.09%	0.20%
dm07	206944	-1.21%	-0.76%	209694	0.10%	0.56%
dm08	230054	-0.86%	-12.11%	235309	1.40%	-10.10%
dm09	281596	-1.07%	-1.79&	284306	-0.12%	-0.85%
avg		-1.29%	-1.66%		-0.38%	-0.78%

Table 7.XVII: SS+ES+LMP Results on Instance Set D-M

Inst	SS+ES+LMP(10h)	TS(10h)Gap	CplGap	SS+ES+LMP(300i)+TS	TS(10h)Gap	CplGap
dl01	277609	-2.43%	-8.92%	279212	-1.87%	-8.39%
dl02	286917	-1.07%	-15.10%	288710	-0.45%	-14.57%
dl03	238241	-2.42%	-15.77%	239593	-1.87%	-15.29%
dl04	346538	-3.79%	-11.85%	347521	-3.51%	-11.60%
dl05	398467	0.28%	-12.68%	394413	-0.74%	-13.57%
dl06	464758	1.21%		457521	-0.37%	
dl07	546171	-1.65%		546506	-1.59%	
dl08	509609	-0.15%		506227	-0.81%	
dl09	653138	2.08%		640347	0.08%	
avg		-0.88%			-1.24%	

Table 7.XVIII: SS+ES+LMP Results on Instance Set D-L

more robust than the tabu search. In almost all the instances (24 out of 27), SS+ES+LMP finds better solutions than tabu search, and the average improvement is reported as 0.44%, 1.29% and 0.88% in the 3 sets. The second experiment indicates that, by combining the SS+ES+LMP and tabu search, we also find better solutions than the tabu search. Compare the results from both experiments, SS+ES+LMP(10h) and SS+ES+LMP(300i)+TS solutions are rather comparable. The SS+ES+LMP(300i)+TS results from D-S and D-M are not as good as SS+ES+LMP(10h), but SS+ES+LMP(300i)+TS presents a better performance on large instances in D-L. We notice, however, the SS+ES+LMP(300i)+TS gives a smaller standard deviation (0.05) than SS+ES+LMP(10h) solutions (0.06) with respect to the tabu search improvement, which suggests SS+ES+LMP(300i)+TS has a more stable performance.

With the experiments on both ISND and DSND instances, we conclude that the proposed SS+ES+LMP algorithm is very efficient to provide good feasible solutions for both models. For all the instances tested, within a significant less solving time, SS generates feasible solutions effectively. Furthermore, LMP disturb the searching area, and direct the SS into promising domains. ES explore the ellipsoid space "between" feasible solutions with MIP solvers, and further improve the solution quality. Experiments show that SS+ES+LMP finds optimal or near-optimal solutions on small instances, and outperforms CPLEX on all instances with interesting size.

7.3 Results for BEST

In this section, we evaluate the performance of the BEST heuristic proposed. The same parameter setting of tabu search, slope scaling, factor perturbation, and ellipsoidal search, as calibrated before are applied. The initial solution of BEST is provided by a SS+ES+LMP phase on the preliminary block set terminates with maximal 300 iterations, or maximal 5 hours solution time, which ever reaches the first.

In order to test both efficiency and robustness, experiments are organized in two stages. In the first stage, we compare the BEST output with the best value obtained by CPLEX, as well as to the results from the SS+ES+LMP heuristic. The same two data sets C-S and C-L are used to verify the quality of BEST. In the second stage, we analyze a rail application and experience some very-large-scale instances.

The CPLEX results and SS+ES+LMP results are shown in Table 7.XIX and 7.XX. The first two columns comes from Table 7.XII and Table 7.XIII. Column CplSol indicates the CPLEX solution within 10 hours CPU time, and column OptGap displays the optimal gaps returned by CPLEX. The next two columns are from Table 7.XIV and Table 7.XV, and display the SS+ES+LMP results with the same CPU time, and their gap with the CPLEX solutions.

Column BEST displays the solution from the BEST heuristic proposed, where the same computing time as CPLEX and SS+ES+LMP is imposed. The BEST solution is compared with CPLEX solution, and the gap percentage is shown in column CplGap. We observe, BEST finds solutions rather close or a little better than CPLEX on 5-yard instances with an average 0.36% improvement, and improves CPLEX significantly on

Inst	CplSol	OptGap	SS+ES+LMP	CplGap	BEST	CplGap	SS+ES+LMP Gap
cs01	75157	0.00%	75157	0.00%	75157	0.00%	0.00%
cs02	72471	2.69%	72483	0.02%	72575	0.14%	0.13%
cs03	78629	0.00%	78731	0.13%	78743	0.14%	0.02%
cs04	82784	0.41%	82784	0.00%	82784	0.00%	0.00%
cs05	98705	0.00%	98705	0.00%	98760	0.06%	0.06%
cs06	110333	7.13%	107576	-2.50%	107785	-2.31%	0.19%
cs07	267198	39.17%	183527	-31.31%	186370	-30.25%	1.55%
cs08	230244	38.23%	166283	-27.78%	168112	-26.99%	1.10%
cs09	174471	32.79%	133110	-23.71%	133197	-23.66%	0.07%
cs10	176248	19.37%	158486	-10.08%	158716	-9.95%	0.15%
cs11	155526	22.97%	132723	-14.66%	134663	-13.41%	1.46%
cs12	183386	18.07%	167424	-8.70%	170195	-7.19%	1.66%
cs13	×	-	212204	-	212045	-	-0.07%
cs14	×	-	195208	-	185678	-	-4.88%
cs15	×	-	228770	-	224068	-	-2.06%
Avg							-0.04%

Table 7.XIX: BEST Results on Instance Set C-S

Inst	CplSol	OptGap	SS+ES+LMP	CplGap	BEST	CplGap	SS+ES+LMP Gap
cl01	94900	4.02%	94676	-0.24%	94929	0.03%	0.27%
cl02	90967	6.78%	89554	-1.55%	90129	-0.92%	0.64%
cl03	128099	2.68%	128097	0.00%	128505	0.32%	0.32%
cl04	79418	0.00%	79418	0.00%	79563	0.18%	0.18%
cl05	129305	0.00%	129305	0.00%	129307	0.00%	0.00%
cl06	105275	9.30%	101656	-3.44%	103193	-1.98%	1.51%
cl07	225581	50.09%	142283	-36.93%	144070	-36.13%	1.26%
cl08	169187	19.04%	158157	-6.52%	158719	-6.19%	0.36%
cl09	293613	53.99%	167934	-42.80%	169997	-42.10%	1.23%
cl10	208201	32.98%	155919	-25.11%	156230	-24.96%	0.20%
cl11	187314	35.07%	145052	-22.56%	147253	-21.39%	1.52%
cl12	226847	29.12%	180356	-20.49%	182786	-19.42%	1.35%
cl13	×	-	271156	-	261401	-	-3.60%
cl14	×	-	257230	-	252635	-	-1.79%
cl15	×	-	326575	-	316503	-	-3.08%
Avg							0.02%

Table 7.XX: BEST Results on Instance Set C-L

7-yard instances with an average 21.80% improvement.

The gap with SS+ES+LMP solution is shown in column SS+ES+LMP Gap, which shows BEST finds very close solutions with the same computing effort. Results of the

BEST algorithm is a bit weak than the SS+ES+LMP heuristic on small and medium instances, however, the difference is minor. On 5-yard instances, the average gap is reported 0.28% and 0.99% on 7-yard instances. These results may be explained by the less BEST iteration number in the total computing time due to the complexity of each approximation problem and additional computation for block generations. When with bigger instances, BEST starts to outperform SS+ES+LMP. These results clearly indicate the trade-off between the complex-but-small AP2 in BEST and simple-but-large AP1 problem in SS+ES+LMP. The overall performance of SS+ES+LMP and BEST is rather comparable. The average gap with SS+ES+LMP is reported -0.04% for instance set C-S and 0.02% for instance set C-L.

To analyze the real instance scale, we study a case with a Class-I rail carrier in North America. The rail carrier operates about 120 stations countrywide, which are aggregated into 27 station clusters. Providing coast-to-coast services, the rail carrier handles roughly 4 million carloads annually, and is facing to around $1,000$ unit requirements (for core and bulk plan) per week.

In the second experiment stage, we generate another set of instances, marked as C-XL, with 15 to 30 yards, 7 time periods and $1,000$ demands. The parameters of instance set C-XL is shown in Table 7.XXI. Apparently, instances in set C-XL are much larger than the ones in C-S and C-L. Any instance in C-XL leads to a huge number of potential blocks and it is impossible to load the whole model to the computers employed.

Inst	Service	Yard	Track	Time	Demand	Block(ini)	Block(final)	BEST
cxl01	3654	15	60	7	1000	10031	92092	1003753
cxl02	4816	15	90	7	1000	13587	73717	868602
cxl03	4116	20	60	7	1000	10969	87514	1101111
cxl04	6048	20	80	7	1000	16212	62587	1071811
cxl05	9779	20	120	7	1000	30940	41538	1103703
cxl06	7434	25	90	7	1000	19978	45143	1155523
cxl07	13342	25	120	7	1000	38339	44366	1137879
cxl08	14896	25	150	7	1000	44653	49399	1089380
cxl09	14784	30	135	7	1000	41258	49595	1428480

Table 7.XXI: Instance Parameters and Results, Set C-XL

The very-large-size network structure of C-XL instances causes to a much longer solving time. We extend the total solving time to 7 days. The initial and final numbers of potential blocks generated are indicated in the column Block(ini) and Block(final), which show the proper performance of the block generation procedure. One observes, comparing with the huge block number in C-S and C-L instances, which could be hundreds of thousands, only a rather small portion of blocks are generated and considered in BEST. The final BEST solutions are listed in the last column of Table 7.XXI.

The two experiments evaluate the performance of BEST heuristic proposed. Computational results from instances of three data sets are satisfactory, and BEST heuristic closely approaches or improves the CPLEX solutions, and finds solutions comparable with the previous SS+ES+LMP heuristic. Without considering all blocks explicitly in the very beginning, the algorithm is capable to address very-large-scale instances and is applicable to solve real-life size instances deriving from the rail transportation.

CHAPTER 8

CONCLUSION

The major objective of this thesis is to build an integrated tactical planning tool for rail freight transportation. Precisely, we study a scheduled service network design problem, in which two consolidation processes, as well as the operation scheduling are considered. By optimizing this problem, an outline of the whole operating plan is produced.

As shown in the literature review, the rail planning issues are generally studied individually, and the overall operating plan is the output of sequential solutions of blocking policy, train routing, scheduling, freight distribution and resource allocation. Some endeavors have been made by previous researchers to combine some aspects, but none of them present synchronous decision-makings for both consolidation procedures (service decision and block decision) with the associated temporal assessment. As far to the author's knowledge, this is the first study to integrate all these most important rail operations in a time-dependent context.

The main contribution of this dissertation comes from two facets. First, from the modeling perspective, we extend the physical network and present a 3-layer time-space structure to describe the complex rail operations. Comparing with the rail network, the proposed network structure is extended in both temporal (multiple time period) and polymorphisial (multiple objects) ways. The new structure allows us to analyze the time-related operations on trains, blocks and cars, and address the traffic itinerary as well as the journey of blocks and services at the same time. Second, from the solution perspective, we developed several solution algorithms, according to different models.

The difficultness of the models comes from two ways, the extreme mathematical complexity and very large applications. The mathematical complexity derives from the trade-off between fixed costs on blocks/services and flow costs on links, trade-off between operations on different layers (block and service), and trade-off between decisions along the time dimension (scheduling). Furthermore, any reasonable application of the models in rails generates thousands of decision variables and millions of flow variables. We study the characteristics of each version of the problem, and developed heuristic algorithms respecting to the DSND model, ISND model, and very-large-scale ISND model.

The DSND, which is a special case of ISND with only non-stop services, is first studied. The case can be often found in many applications such as industrial transportation (e.g. coal, green products). By removing the multi-stop services in the system, the problem is simplified and keeps a reduced set of potential services. The non-stop service case also holds a favorable property that each block is taken by several direct services, rather than service sections of multi-stop services. The good property allows us to focus on the block network design, instead of two level designs synchronously. According to the DSND problem, the network is degenerated to a 2-layer time-space network.

A tabu search algorithm on cycle-based neighborhood is proposed for DSND. Compared with Ghamlouche et al. [57], improvements have been made through several key setups. First, we expend the cycle-based concept and consider both block fixed costs and direct service fixed cost in each local search move. Second, we focus on the block decisions and only concern the cycles deriving from open blocks. Numerical experiments demonstrate that the proposed algorithm is robust for providing near-optimal solutions within reasonable solving efforts and outperforms CPLEX on large instances.

We developed a SS+ES+LMP algorithm to solve the ISND model. The heuristic algorithm is based on slope scaling idea, and the solving procedure is guided by long-term memories which keep a record of the history search track. A factor perturbation phase is adopted to further intensify/diversify the procedure. We also proposed an ellipsoidal search algorithm, to integrate the good solutions found during the slope scaling procedures.

The algorithmic contribution of the SS+ES+LMS exists in the modification of slope scaling and proposition of ellipsoidal search. Given the characteristics of the model, in the slope scaling, fixed costs from both blocks and services are approximated by linearization factors. The service fixed costs are described by total workload given the variant car flows on different service sections. The slope scaling process is accelerated by adopting heuristic solutions for each linear approximation problem. The ellipsoidal search we proposed can be viewed as a new effort to combine heuristic ideas and exact solvers, in which a small "ellipsoid" in the solution space is shaped by variable selection and fixing, and explored by MIP solver. The SS+ES+LMP algorithm performs robustly in experiments, where it either finds a near-optimal solution or discovers feasible solutions with substantial improvements than the mathematical solver.

We also propose a hybrid solution to solve the very-large-scale ISND problems. The BEST algorithm integrates the slope scaling, ellipsoidal search, tabu search and column generation ideas. A new slope scaling scheme is proposed, which naturally decompose the model into two subproblems: one block network design problem in the bottom two layers, and a block generation problem in the top two layers. The decomposition allows blending tabu search with column generation ideas: the previous tabu search algorithm is modified to tackle the approximation problems, in which the residual values in each local search move conduct the block generation process. By starting with only a very small set of block decision variables, the algorithm is capable to address very large instances. The result from the largest experiment (with 30 yards, 7 time periods, $1,000$ demands) makes it possible to identify national operating plan with weekly schedule for large-scale railways.

Other contributions behind the research derive from its generalization. Modeling wise, the 3-layer network structure can be further applied to address the operations in related transportation modes, such as LTL, aviation and shipping lines where consolidation takes an important part, or intermodal transports where the long-haul legs can be viewed as blocks in rail. Furthermore, although presented in a monograph for rail transportation, the powerful performance of the methodologies proposed inspires the algorithmic generalization, to the general multi-layer time-space network design problems, or even more general network design problems. For example, the BEST algorithm can be intuitively applied to a multi-level network design problem which consists of several layers, where links in a layer present both fixed design cost and linear flow cost, and a path in one layer is associated with fixed cost and flow cost deriving from its component links, and supplies a conceptual link for the next layer. Another generalization avenue comes from ellipsoidal search. The ellipsoidal search generates new solutions by exploring an ellipsoid space between reference solutions. In this research, the idea is particularly implemented on the ISND model studied. The outstanding performance suggests a broader range of applications in solving combinatorial optimization problems (e.g. FMCND) cooperating with a heuristic scheme (e.g. tabu search).

Aside from generalization and more applications of the proposed models and methodologies, there are several issues that we are recommending for the future research in the rail tactical planning problem. One might want to further integrate asset (locomotives, and crew) allocation into the ISND. This could be realized with a network with more layers to represent operations on additional resources. Another potential avenue is inspired by the stochastic nature of reality, where demand expectation is generally presented with possibilities. A stochastic model which balances the scenarios might attract more attention from industry. The models and methodologies proposed in this thesis present alternatives for solving each individual scenario, however, the cooperation of threads is up to be studied. This brings another interesting topic, parallel programming or distributed computing. Other than the application on stochastic models, parallel computing and distributed computing are also possibly applicable to the deterministic models we studied. For example, in slope scaling, one may extend the factor perturbation phase, and adopt several perturbation policies in parallel. Multiple approximation problems can be produced, which could be parallelized and computed on different computers to find many reference solutions simultaneously. Ellipsoidal search phase can also be parallelized by solving elite problems each derives from a pair of reference solutions in the reference set. A further study of such issues and development of approximation schemes to permit real-time implementations are suggested as future research topics.

BIBLIOGRAPHY

[1] Class I railroad statistics. *Association of American Railroads*, 2009.

[2] R. K. Ahuja, C. B. Cunha, and G. Şahin. Network models in railroad planning and scheduling. *Tutorials in Operations Research*, 1:54–101, 2005.

[3] R. K. Ahuja, J. Liu, J. B. Orlin, D. Sharma, and L. A. Shughart. Solving real-life locomotive-scheduling problem. *Transportation Sci.*, 39(4):503–517, 2005.

[4] R. K. Ahuja, K. C. Jha, and J. Liu. Solving real-life railroad blocking problems. *Interfaces*, 37:404–419, 2007.

[5] M. Al-Amin, M. A. Forbes, and D. H. Noble. Production of locomotive rosters for a multi-class single-locomotive problem. *Journal of the Operational Research Society*, 50(10):1004–1010, 1999.

[6] A. M. Alvarez, J. Gonzalez-Velarde, and K. De-Alba. GRASP embedded scatter search for the multicommodity capacitated network design problem. *J. of Heuristics*, 11:233–257, 2005.

[7] A. M. Alvarez, J. Gonzalez-Velarde, and K. De-Alba. Scatter search for network design problem. *Annals of Oper. Res.*, 138:159–178, 2005.

[8] J. Andersen, T. G. Crainic, and M. Christiansen. Service network design with management and coordination of multiple fleets. *Eur. J. Oper. Res.*, 193(2):377–389, 2009.

[9] J. Andersen, T. G. Crainic, and M. Christiansen. Service network design with asset management: Formulations and comparative analyzes. *Transportation Res. C*, 17(2):397–207, 2009.

[10] A. Armacost, C. Barnhart, and K. Ware. Composite variable formulations for express shipment service network design. *Transportation Sci.*, 36(1):1–20, 2002.

[11] A. A. Assad. Modelling of rail networks: Toward a routing/make-up model. *Transportation Res. B*, 14(1-2):101–114, 1980.

[12] A. Balakrishnan. LP extreme points and cuts for the fixed charge network design problem. *Math. Programming*, 39:263–284, 1987.

[13] A. Balakrishnan, T. L. Magnanti, and R. T. Wong. A dual-ascent procedure for large-scale uncapacitated network design. *Oper. Res.*, 37(5):716–740, 1989.

[14] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani. Network design. In F. Maffioli M. Dell'Amico and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 311–334. John Wiley & Sons, 1997.

[15] C. Barnhart and S. Shen. Logistics service network design for time-critical delivery. *Lecture Notes in Computer Science*, 3616:86–105, 2005.

[16] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, 46(3):316–329, 1998.

[17] C. Barnhart, H. Jin, and P. H. Vance. Railroad blocking: A network design application. *Oper. Res.*, 48(4):603–614, 2000.

[18] C. Barnhart, N. Krishnan, D. Kim, and K. Ware. Network design for express shipment delivery. *Computational Optimization and Applications*, 21:239–262, 2002.

[19] T. Bektas, T. G. Crainic, and V. Morency. Improving the performance of rail yards through dynamic reassignments of empty cars. *Transportation Res. C*, 17 (3):259–273, 2009.

[20] D. Bienstock and O. Günlük. Capacitated network design - polyhedral structure and computation. *INFORMS J. Comput.*, 43(8):243–259, 1996.

[21] S. Binato, M. Pereira, and S. Granville. A new Benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems*, 16:235–240, 2001.

[22] L. D. Bodin, B. L. Golden, A. D. Schuster, and W. Romig. A model for the blocking of trains. *Transportation Res. B*, 14(1-2):115–120, 1980.

[23] N. Bojović. A general system theory approach to rail freight car fleet sizing. *Eur. J. Oper. Res.*, 136:136–172, 2002.

[24] U. Brännlund, P. O. Lindberg, A. Nou, and J.-E. Nilsson. Railway timetabling using Lagrangian relaxation. *Transportation Sci.*, 32(4):358–369, 1998.

[25] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Oper. Res.*, 50(5):851–861, 2002.

[26] A. Caprara, M. Monaci, P. Toth, and P. L. Guida. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154: 738–753, 2006.

[27] R. K. Cheung and B. Muralidharan. Dynamic routing for priority shipments in LTL service networks. *Transportation Sci.*, 34(1):86–98, 2000.

[28] M. Chouman, T. G. Crainic, and B. Gendron. Revue des inégalités valides pertinentes aux problèmes de conception de réseaux. *INFOR*, 41:5–33, 2003.

[29] M. Chouman, T. G. Crainic, and B. Gendron. A cutting-plane algorithm based on cutset inequalities for multicommodity capacitated fixed charge network design. Research Report CRT-2003-16, Centre de recherche sur les transports, Montréal, Canada, 2003.

[30] M. Chouman, T. G. Crainic, and B. Gendron. A cutting-plane algorithm for multicommodity capacitated fixed-charge network design. Research Report CIRRELT-2009-20, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Montréal, Canada, 2009.

[31] J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation Sci.*, 32(4):380–404, 1998.

[32] A. M. Costa. A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Oper. Res.*, 32:1429–1450, 2005.

[33] A. M. Costa, J.-F. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392, 2009.

[34] T. G. Crainic. Service network design in freight transportation. *Eur. J. Oper. Res.*, 122:272–288, 2000.

[35] T. G. Crainic. Parallel computation, co-operation, tabu search. Research Report CRT-2001-32, Centre de recherche sur les transports, Montréal, Canada, 2001.

[36] T. G. Crainic. Long-haul freight transportation. In R. W. Hall, editor, *Handbook of Transportation Science*. Kluwer, Norwell, MA, 1999.

[37] T. G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627, 2002.

[38] T. G. Crainic and G. Laporte. Planning models for freight transportation. *Eur. J. Oper. Res.*, 97:409–438, 1997.

[39] T. G. Crainic and J.-M. Rousseau. Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Res. B*, 20B(3):225–242, 1986.

[40] T. G. Crainic and M. Toulouse. Parallel strategies for meta-heuristics. Research Report CRT-2001-06, Centre de recherche sur les transports, Montréal, Canada, 2001.

[41] T. G. Crainic, J. A. Ferland, and J.-M. Rousseau. A tactical planning model for rail freight transportation. *Transportation Sci.*, 18(2):165–184, 1984.

[42] T. G. Crainic, M. Gendreau, and J. M. Farvolden. A simplex-based tabu search method for capacitated network design. *INFORMS J. Comput.*, 12(3):223–236, 2000.

[43] T. G. Crainic, A. Frangioni, and B. Gendron. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1-3):73–79, 2001.

[44] T. G. Crainic, B. Gendron, and G. Hernu. A slope scaling/Lagrangean perturbation heuristic with long-term memory for multicommodity capacitated fixed-charge network design. *J. of Heuristics*, 10(5):525–545, 2004.

[45] T. G. Crainic, Y. Li, and M. Toulouse. A first multilevel cooperative algorithm for capacitated multicommodity network design. *Computers & Oper. Res.*, 33: 2602–2622, 2006.

[46] P. J. Dejax and T. G. Crainic. A review of empty flows and fleet management models in freight transportation. *Transportation Sci.*, 21(4):227–247, 1987.

[47] J. M. Farvolden and W. B. Powell. Subgradient methods for the service network design problem. *Transportation Sci.*, 28:256–272, 1994.

[48] M. Fischetti and A. Lodi. Local branching. *Math. Programming, Ser. B*, 98: 23–47, 2003.

[49] M. Fischetti, C. Polo, and M. Scantamburlo. A local branching heuristic for mixed-integer programs with 2-level variables,with an application to a telecommunication network design problem. *Networks*, 44:61–72, 2004.

[50] B. Fortz and M. Poss. An improved Benders decomposition applied to a multilayer network design problem. *Oper. Res. Letters*, 37(5):359–364, 2009.

[51] A. Frangioni and B. Gendron. 0-1 reformulations of the multicommodity capacitated network design problem. *Discrete Applied Mathematics*, 157(6):1229–1241, 2009.

[52] B. Gendron and T. G. Crainic. Relaxations for multicommodity network design problems. Research Report CRT-965, Centre de recherche sur les transports, Montréal, Canada, 1994.

[53] B. Gendron and T. G. Crainic. Parallel implementations of bounding procedures for multicommodity capacitated network design problems. Research Report CRT-94-45, Centre de recherche sur les transports, Montréal, Canada, 1994.

[54] B. Gendron and T. G. Crainic. Bounding procedures for multicommodity capacitated fixed charge network design problems. Research Report CRT-96-06, Centre de recherche sur les transports, Montréal, Canada, 1996.

[55] B. Gendron, T. G. Crainic, and A. Frangioni. Multicommodity capacitated network design. In P. Soriano and B. Sansò, editors, *Telecommunications Network Planning*, pages 1–19. Kluwer Academic, 1999.

[56] A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Sci.*, 20:822–844, 1974.

[57] I. Ghamlouche, T. G. Crainic, and M. Gendreau. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Oper. Res.*, 51(4): 655–667, 2003.

[58] I. Ghamlouche, T. G. Crainic, and M. Gendreau. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Oper. Res.*, 131(1-4):109–133, 2004.

[59] M. F. Gorman. An application of genetic and tabu searches to the freight railroad operating plan problem. *Annals of Oper. Res.*, 78:51–69, 1998.

[60] M. F. Gorman. Santa Fe Railway uses an operating plan model to improve its service design. *Interfaces*, 28(4):1–12, 1998.

[61] T. Grünert and H.-J. Sebastian. Planning models for long-haul operations of postal and express shipment companies. *Eur. J. Oper. Res.*, 122:289–309, 2000.

[62] O. Günlük. A branch-and-cut algorithm for capacitated network design problems. *Math. Programming, Ser. A*, 86:17–39, 1999.

[63] A. E. Haghani. Formulation and solution of combined train routing and makeup, and empty car distribution model. *Transportation Res. B*, 23B(6):433–452, 1989.

[64] J. Hellstrand, T. Larson, and A. Migdalas. A characterization of the uncapacitated network design polytope. *Oper. Res. Letters*, 12(3):159–163, 1992.

[65] M. Hewitt, G. L. Nemhauser, and M. W. P. Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed charge network flow problem. *INFORMS J. Comput.*, 22(2):314–325, 2010.

[66] K. Holmberg and J. Hellstrand. Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound. *Oper. Res.*, 46(2):247–259, 1998.

[67] K. Holmberg and D. Yuan. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design. *Oper. Res.*, 48(3):461–481, 2000.

[68] C. L. Huntley, D. E. Brown, D. E. Sappington, and B. P. Markowicz. Freight routing and scheduling at CSX transportation. *Interfaces*, 25(3):58–71, 1995.

[69] P. Ireland, R. Case, J. Fallis, C. Van Dyke, J. Kuehn, and M. Meketon. The Canadian Pacific Railway transforms operations by using models to develop its operating plans. *Interfaces*, 34(1):5–14, 2004.

[70] B. Jansen, P. C. J. Swinkels, G. J. A. Teeuwen, B. van Antwerpen de Fluiter, and H. A. Fleuren. Operational planning of a large-scale multi-modal transportation system. *Eur. J. Oper. Res.*, 156:41–53, 2004.

[71] K. C. Jha, R. K. Ahuja, and G. Şahin. New approaches for solving the block-to-train assignment problem. *Network*, 51:48–62, 2008.

[72] M. Joborn, T. G. Crainic, M. Gendreau, K. Holmberg, and J. T. Lundgren. Economies of scale in empty freight car distribution in scheduled railways. *Transportation Sci.*, 38(2):121–134, 2004.

[73] N. Katayama, M. Chen, and M. Kupo. A capacity scaling heuristic for the multicommodity capacitated network design problem. *Journal of Computational and Applied Mathematics*, 232(1):90–101, 2009.

[74] M. H. Keaton. Designing optimal railroad operating plans: Lagrangian relaxation and heuristic approaches. *Transportation Res.*, 23B:415–431, 1989.

[75] M. H. Keaton. Designing railroad operating plans: A dual adjustment method for implementing Lagrangian relaxation. *Transportation Sci.*, 26(4):263–279, 1992.

[76] A. Kershenbaum and R. R. Boorstyn. Centralized teleprocessing network design. *Networks*, 13:279–293, 1983.

[77] D. Kim and P. M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Oper. Res. Letters*, 24: 195–203, 1999.

[78] D. Kim, C. Barnhart, K. Ware, and G. Reinhardt. Multimodal express package delivery: A service network design application. *Transportation Sci.*, 33(4):391–407, 1999.

[79] D. Kim, X. Pan, and P. M. Pardalos. An enhanced dynamic slope scaling procedure with tabu scheme for fixed charge network flow problem. *Computational Economics*, 27:273–293, 2006.

[80] G. Kliewer and L. Timajev. Relax-and-cut for capacitated network design. *Lecture Notes in Computer Science*, 3669:47–58, 2005.

[81] M. J. Kuby and R. G. Gray. The hub network design problem with stopovers and feeders: The case of federal express. *Transportation Res. A*, 27A:1–12, 1993.

[82] O. K. Kwon, C. D. Martland, and J. M. Sussman. Routing and scheduling temporal and heterogeneous freight car traffic on rail network. *Transportation Res. E*, 34(2):101–115, 1998.

[83] M. W. Lewis. On the use of guided design search for discovering significant decision variables in the fixed-charge capacitated multicommodity network design problem. *Networks*, 53(1):6–18, 2009.

[84] S. Livramento, A. V. Moura, F. K. Miyazawa, M. M. Harada, and R. A. Miranda. A genetic algorithm for telecommunication network design. *Applications of Evolutionary Computing Lecture Notes in Computer Science*, 3005:140–149, 2004.

[85] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Sci.*, 18(1):1–55, 1984.

[86] T. L. Magnanti and R. T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 26:112–154, 1986.

[87] A. Marín and J. Salmerón. Tactical design of rail freight network. part i: Exact and heuristic methods. *Eur. J. Oper. Res.*, 90(1):26–44, 1996.

[88] M. Minoux. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, 19:313–360, 1989.

[89] E. K. Morlok and R. B. Peterson. Final report on a development of a geographic transportation network generation and evaluation model. In *Processing of the Eleventh Annual Meeting*, pages 71–105. Transportation Research Forum, 1970.

[90] A. K. Narisetty, J.-P. P. Richard, D. Ramcharan, D. Murphy, G. Minks, and J. Fuller. An optimization model for empty freight car assignment at Union Pacific Railroad. *Interaces*, 38:89–102, 2008.

[91] A. M. Newman and C. A. Yano. Centralized and decentralized train scheduling for intermodal operations. *IIE Transactions*, 32:743–754, 2000.

[92] A. M. Newman, L. K. Nozick, and C. A. Yano. Optimization in the rail industry. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*, pages 704–718. Oxford University Press, New York, NY, 2002.

[93] H. N. Newton, C. Barnhart, and P. H. Vance. Constructing railroad blocking plans to minimize handling costs. *Transportation Sci.*, 32(4):330–345, 1998.

[94] L. K. Nozick and E. K. Morlok. A model for medium-term operations planning in an intermodal rail-truck service. *Transportation Res. A*, 31(2):91–107, 1997.

[95] M. B. Pedersen and T. G. Crainic. Optimization of intermodal freight service schedules on train canals. Research Report CIRRELT-2007-51, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Montréal, Canada, 2007.

[96] M. B. Pedersen, T. G. Crainic, and O.B.G.Madsen. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Sci.*, 43(2):158–177, 2009.

[97] W. B. Powell. A local improvement heuristic for the design of less-than-truckload motor carrier networks. *Transportation Sci.*, 20(4):246–257, 1986.

[98] W. B. Powell and T. A. Carvalho. Dynamic control of logistics queueing networks for large-scale fleet management. *Transportation Sci.*, 32:90–109, 1998.

[99] W. B. Powell and T. A. Carvalho. Real-time optimization of containers and flatcars for intermodal operations. *Transportation Sci.*, 32:110–126, 1998.

[100] W. B. Powell and I. A. Koskosidis. Shipment routing algorithms with tree constraints. *Transportation Sci.*, 26(3):230–245, 1992.

[101] W. B. Powell and Y. Sheffi. The loading plan problem of motor carriers: Problem description and a proposed solution approach. *Transportation Res.*, 17A(6):471–480, 1983.

[102] W. B. Powell and Y. Sheffi. Design and implementation of an interactive optimization system for the network design in the motor carrier industry. *Oper. Res.*, 20:12–29, 1989.

[103] W. Rei, J.-F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. *INFORMS J. Comput.*, 21(2):333–345, 2009.

[104] I. Rodríguez-Martín and J. J. Salazar-González. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Oper. Res.*, 37(3):575–581, 2010.

[105] J. Roy and T. G. Crainic. Improving intercity freight routing with a tactical planning model. *Interfaces*, 22(3):31–44, 1992.

[106] J. Roy and L. Delorme. NETPLAN: A network optimization model for tactical planning in the less-than-truckload motor-carrier industry. *INFOR*, 27(1):22–35, 1989.

[107] M. Sellmann, G. Kliewer, and A. Koberstein. Lagrangian cardinality cuts and variable fixing for capacitated network design. In *Processing Lecture Notes in Computer Science*, volume 2461, pages 845–858. Algorithms-ESA, 2002.

[108] H. D. Sherali and A. B. Suharko. A tactical decision support system for empty railcar management. *Transportation Sci.*, 32(4):306–329, 1998.

[109] V. Sridhar and J. S. Park. Benders-and-cut algorithm for fixed-charge capacitated network design problem. *Eur. J. Oper. Res.*, 125:622–632, 2000.

[110] J. Stallaert. Valid inequalities and separation for capacitated fixed charge flow problems. *Discrete Applied Mathematics*, 98:265–274, 2000.

[111] J. Stallaert. The complementary class of generalized flow cover inequalities. *Discrete Applied Mathematics*, 77:73–80, 1997.

[112] M. Sun, J. E. Aronson, P. G. McKeown, and D. Drinka. A tabu search heuristic procedure for the fixed charge transportation problem. *Eur. J. Oper. Res.*, 106 (2-3):441–456, 1998.

[113] C. A. Yano and A. M. Newman. Scheduling trains and containers with due dates and dynamic arrivals. *Transportation Sci.*, 35(2):181–191, 2001.

[114] N. C. Zaleta and A. M. A. Socarrás. Tabu Search-based algorithm for capacitated multicommodity network design problem. In *Proceedings of the 14th International Conference on Electronics*, pages 144–148. Communications and Computers, 2004.

Appendix I

Instance Parameters: Set C-C

Inst	Block	Service	Yard	Track	Time	Demand
cc01	3350	450	4	10	10	90
cc02	3340	460	4	10	10	120
cc03	14970	1280	5	14	10	100
cc04	9490	890	5	14	10	150
cc05	27350	1200	5	14	10	200
cc06	5790	810	5	18	10	120
cc07	19970	1590	5	18	10	150
cc08	75960	2550	5	18	10	200
cc09	106310	2090	7	20	10	150

Table I.I: Instance Parameters: Set C-C