MODELLING AND ADVANCED OPTIMISATION METHODS FOR THE MULTI-SHIFT FULL TRUCKLOAD VEHICLE ROUTING PROBLEM

by

Ning Xue, BSc (Hons), MSc

Thesis submitted to the University of Nottingham

for the degree of Doctor of Philosophy

School of Computer Science

The University of Nottingham

September 2016

Abstract

This thesis is concerned with a real-world multi-shift drayage problem at a large international port with multiple docks being operated simultaneously. Several important issues in the drayage problem are identified and a set covering model is developed based on a novel route representation. The model adopts an implicit solution representation to reduce the problem size and aims to find a set of vehicle routes with minimum total cost to deliver all commodities within their time windows. As accurate travel time prediction is necessary to construct the vehicle routes, a short-haul travel time prediction model and an algorithm using real-life GPS data are studied. The output of the prediction model can be used as an input for the set covering model.

The set covering model for the multi-shift full truckload transportation problem can be directly solved by a commercial solver for small problems, but results in prohibitive computation time for even moderate-sized problems. In order to solve medium- and large-sized instances, we proposed a 3-stage hybrid solution method and applied it to solve real-life instances at a large international port in China. It was shown that the method is able to find solutions that are very close to the lower bounds. In addition, we also proposed a more efficient hybrid branch-and-price approach. Results show the method performed well and is more suited for solving real-life, large-sized drayage operation problems.

Acknowledgements

My sincerest thanks go to my supervisor, Prof. Ruibin Bai. The valuable advice and support he has given me over the last few years has enabled me to finish this thesis. I am also grateful to my supervisor, Prof. Gethin Wyn Roberts, for his help and assistance.

I am highly grateful for the financial funding from the IDIC and the School of Computer Science, The University of Nottingham. It would have not been possible for me to enter this PhD programme without their support.

Finally, I would like to thank my family and friends for their unconditional support throughout my academic career and life in general. There is no way I would have reached this point if it wasn't for you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Freight transportation is of great importance to the economy. It supports production, trade, and consumption by ensuring the timely provision of raw materials and consumer products. Freight transportation accounts for a large proportion of the cost of a product and the national expenditures of any country [45]. Analysis results suggest that, the volumes of freight traffic and freight turnover in China are positively correlated with GDP [74].

In China, ports are important gateways for domestic and foreign trade. According to statistics [3] from the largest (based on throughput) container ports worldwide in 2015, seven of the top 10 ports in the world are located in China. In 2015, the major ports in China handling a total of 212 million TEUs (twenty-foot equivalent unit) increased 4.5% compared with 2014 [2]. This resulted in a highly competitive environment for container transportation firms that needed to offer high quality, low cost services to their customers in order to remain profitable.

The performance of containerised shipping as a whole depends on intermodal container transportation [86], which has experienced rapid development since the 1980s [131]. It concerns container transportation by multiple modes (i.e. road, rail, and maritime) within a single transportation chain [149]. In intermodal container transportation, the longest journey (e.g. between countries, cities) is usually by sea or rail, while connections between ports, docks or connections from rural to urban areas mainly rely on truck drayage (here referred to as drayage). Despite the relatively short distances compared to maritime or rail hauls, drayage accounts for a large percentage of origin-to-destination expenses, as transportation per TEU per kilometre is higher compared with other modes [175].

While freight transportation by truck is indispensable for economic development, it also comes with environmental hazards and human health risks. Drayage trucks powered by diesel engines are a major contributor to poor air quality [148], as they are typically older and more polluting than the average long haul truck. Hence, even though it is critical to improve drayage operations in order to keep costs low, it is also necessary to reduce their deleterious emissions impacts on the environment.

The study of drayage is relatively new due to the growth of globalised trade. The term "drayage" was used to describe the overland transport of cargo to and from barges or rail yards [157]. In a broader sense, drayage includes regional movements of loaded or empty containers by trucks between rail yards, shippers, consignees, and equipment yards [142]. A typical drayage move involves either the pickup of an import container from or delivery of an export container to a terminal. However, the complexities of the business usually require more than a single drayage trip for each container moved, owing to the movement preferences, policies and capabilities of carriers.

Scheduling for drayage, in essence, is a full truckload vehicle routing problem (FTLVRP). This is a challenging and time-consuming process, because the simplest form of truck scheduling is already similar to vehicle routing problem with pickup and delivery (VRPPD), a well-known NP-hard combinatorial problem that is very difficult to solve. The difficulty increases if more factors are considered. Besides vehicle capacity and time-related constraints (e.g. available time and deadline of transporting a container, working time of driver), the terminal storage facilities and number of trucks accessing the terminal within a given time slot may also be limited [144]. The complexity of their task leads to inefficiencies, delays and unnecessary emissions. Reducing these problems is a matter of great importance to the drayage industry.

The Port of Ningbo-Zhoushan is one of the busiest (based on cargo tonnage) ports in China. It handled over 20 million TEUs in 2015, an increase of 6.36% compared with 2014 [2]. To meet growing service demands, Ningbo Port Co., Ltd is making efforts to increase drayage efficiency and maximise throughput, while at the same time reduce air pollution. Compared with other drayage problems, transport distances in the cross-dock container shipment problem for the Port of Ningbo are relatively short. Another unique concern of the company is that its schedules are shift-based.

In accordance with labour laws, each truck must return to a depot for driver changeovers after a shift (12 hours). A schedule typically spans from 4 to 8 shifts in order to maximise efficiency. Due to the unique factors of the problem, in particular the much longer planning horizons, existing methods are either not efficient or not applicable.

The real-life problem has the following characteristics:

- The unit size of the commodity (container) is equivalent of that of the trucks. Therefore one unit of a commodity is shipped directly to its destination without transfers or consolidations.
- Schedules are shift based. Each truck has to go back to a depot for driver changeovers after a shift due to labour law related regulations. Thus this is a multi-shift vehicle routing problem.
- All docks are within a short distance of each other and a unit of any commodity could be completed within a shift.
- The time window for each commodity varies considerably from 1-2 hours, up to 6 shifts.
- The total quantities of all the commodities within a planning horizon can be very large (up to 2000) but the number of distinct physical nodes is relatively small (less than 10).

1.2 Thesis Structure

This thesis studies a real-world container inter-dock transportation problem. In this study, there is a set of docks with short distance (e.g. 5-15 minutes) between each other and a large number (e.g. 2000) of commodity (container) transportation demand between these docks within a planning horizon (from 4 shifts to 8 shifts). The container transportation is conducted by truck and each truck has to go back to a depot for driver changeovers after a shift due to labour law related regulations. Each commodity has a time window (i.e. from time when the commodity becomes available to the time when it has to be delivered to its destination), which varies considerably from 1 to 2 h, up to 6 shifts. Our objective is to minimise the total vehicle travel distance for transporting a large number of non-consolidatable commodities between a relatively small number (less than 10) of nodes (docks), satisfying various time window constraints concerning commodities and drivers. It is a combinatorial optimisation problem and the real-world problem is very hard to be solved. The structure of the thesis, in the form of a brief overview of each of the individual chapters, will be presented in the following.

Chapter 2 introduces several terms and optimisation technologies that appears in the rest of this thesis. Some technologies which have been successful for solving combinatorial optimisation problems are also reviewed and promising techniques such as genetic algorithm (GA), variable neightbourhood search (VNS) and branch-and-price are highlighted.

Chapter 3 firstly provides a detailed description of the problem concerned in this thesis. Secondly, the terms and models of several typical vehicle routing problems are introduced. Thirdly, we explain where our work is situated in the broader field of freight transportation research. The problem is formally defined as a multi-shift full truckload vehicle routing problem (MFTLVRP). After that, relevant vehicle routing problems that share similar properties of our research and their solution methods are highlighted and reviewed.

In Chapter 4, a set covering model is developed for the problem based on a novel route representation and a container-flow mapping. A lower bound of the problem is also obtained by relaxing the time window constraints to the nearest shifts and transforming the problem into a service network design problem. Finally, the features and merits of the model are discussed.

As travel times are necessary and important parameters for the set covering model formulated here, in Chapter 5, we analyse real-life GPS data were obtained from a container truck fleet at the Port of Ningbo and estimate accurate travel times between different docks. By analysing the data, we observed an increase in travel time patterns during peak times. This motivated us to investigate further to try to estimate travel times more accurately and efficiently. We ultimately develop a real-world short-haul travel time prediction model. The processes of data preparation and variable and model selection are also illustrated.

Instead of using fixed travel times when generating feasible routes for the set covering model in practice, this prediction model is suggested to estimate travel times due to the variability of traffic and driving conditions. Hence, even though the set covering model studied here is deterministic, its travel time parameters can be non-deterministic. The idea of this approach is an alternative method to other non-deterministic approaches (e.g. stochastic programming, robust scheduling). Unfortunately, the travel time prediction model is not integrated with the set covering model for further analysis, as only limited size of data obtained so that we are not able to obtain a connected graph with travel time between every pair of docks. For the travel time parameters used in experiments in Chapter 6 and 7, only part of travel times were estimated by analysing the GPS data, the rest were estimated by experience (i.e. travel time suggested by experienced manager in Port).

Chapter 6 presents a hybrid solution method for the set covering model presented in Chapter 4. In order to evaluate the feasibility and performance of the model, we applied it to solve real-life instances at the Port of Ningbo. In addition, test instances with certain features were created in order to fully assess the approach and to gain knowledge that may not be discovered from real-life instances. The results are also compared against a reactive shaking variable neighbourhood search (VNS) and a simulated annealing hyperheuristic method (SAHH).

The real-life problem has some special features to permit the hybrid solution method being used. However, in addition to the excessive computational time by the hybrid algorithm, it may become even invalid for problems that do not possess the features present in this problem. To address this issue, in chapter 7, a more efficient hybrid branch-and-price approach is studied. This chapter extends the previous work and presents a significantly more efficient approach by hybridising metaheuristics (VNS and GA) with an hybrid branch-and-price approach.

Chapter 8 provides a summary of the work undertaken in the thesis and the contributions contained therein. An outline of possible directions for future research is given.

1.3 Aims and Scope

The drayage problem discussed in this thesis is drawn from a real-world container inter-dock transportation problem raised in the Port of Ningbo. The primary objective of this study is to minimise the total vehicle travel distance for transporting a large number of non-consolidatable commodities between a relatively small number of nodes (docks), satisfying various time window constraints concerning commodities and drivers. One of the unique characteristics of this problem is that schedules are shift based. This problem will be described in detail in Section 3.2 and formally defined as a multi-shift full truckload vehicle routing problem (MFTLVRP). Overall, the aim of this research is to develop models and algorithms for the problem that: 1. Can be utilised by any large port with multiple docks being operated simultaneously; 2. Should be capable of producing optimised truck scheduling solutions for the given requirements. Specifically, we want to:

1. Identify potentially important issues in an MFTLVRP;
2. Formulate a practical model that captures the main characteristics of the MFTLVRP so that it can be used in practice;

3. Analyse fleet GPS data and estimate accurate travel times between different docks, as travel times are necessary and important parameters for the model formulated in 2;

4. Investigate both exact and metaheuristics approaches for the model formulated in 2.

1.4 Contributions

A number of academic publications have been produced as a result of completing the research presented in this thesis. These publications are listed in order of the relevant chapter in which this research is contained:

Chapter 4, 6:

- Ruibin Bai, Ning Xue, Jianjun Chen, and Gethin Wyn Roberts. A set-covering model for a bidirectional multi-shift full truckload vehicle routing problem. Transportation Research Part B: Methodological, 79(0):134 - 148, 2015.

Chapter 5:

- Ning Xue, Ruibin Bai, and Gethin Wyn Roberts. A Study of Automated Container Truck Travel Time Prediction Based on Real-life GPS data Using ARIMA. The Ninth International Conference on Operations and Supply Chain Management, Ningbo, China. 2015. (**Best paper award**)

Chapter 7:

- Ning Xue, Ruibin Bai. A Branch-and-price Approach for the Bidirectional Multi-shift Full Truckload Vehicle Routing Problem. The 8th International Annual Conference of Chinese Scholars Association for Management Science and Engineering, Shenyang, China. 2015.
- Ning Xue, Ruibin Bai, Rong Qu, Uwe Aickelin. A Hybrid Branch-and-price Method for the Multi-shift Full Truckload Vehicle Routing Problem. Submitted to INFORMS Journal on Computing.

Chapter 2

Optimisation Techniques: An Overview

2.1 Introduction

In computational complexity theory, the problems are usually classified by two distinct headings: P and NP. P (standing for polynomial) represents the class of the problems that are solvable by a deterministic algorithm with polynomial time complexity. While NP is the class of the problems that can be solved in polynomial time by a nondeterministic algorithm (NP stands for non-deterministic polynomial). NP-hard (non-deterministic polynomial-time hard) is a type of problems that are at least as hard as any problem in NP [7]. A nondeterministic algorithm is an algorithm that, even for the same input, can exhibit different behaviors (i.e. solving time and solution) on different runs, which is opposed to a deterministic algorithm [75].

In combinatorial problems, we look for *"an object from a finite (or possibly countable infinite) set, typically an integer set, permutation, or graph"* [154]. The number of possible solutions in the search space of a combinatorial problem is usually so large as to forbid an exhaustive search for the best answer. A typical example of a combinatorial problem is the travelling salesman problem (TSP), which is NP-hard [123]. The TSP determines the shortest path that starts from a given city, then passes through all the other cities and returns to the initial city. A 10-city TSP has around 181,000 possible solutions, however, a 20-city TSP has about 10 quadrillion possible solutions. This exponential increase in problem size makes some exact algorithms impractical for solving such large instances. Therefore, one may apply an approximation approach (heuristics or metaheuristics) that can obtain satisfactory solution quality within a reasonable computation time.

The truck drayage problem is an NP-hard [123] problem. There is no yet known polynomial time bounded algorithm that can solve every instance to optimality. This chapter try to introduce several terms and optimisation technologies that appears in this thesis. Some technologies which have been successful for NP-hard combinatorial optimisation problems are also reviewed and promising techniques are highlighted.

2.2 Exact Methods

Exact methods are able to find the optimal solution and assess its optimality. The computation time, however, often grows considerably with the instance size. In general, small or moderate instances can be solved to provable optimality, but they may suffer in large scale problems, especially those known to be NP-hard. This section reviews some well-known exact methods that include: dynamic programming, branch-and-bound, branch-and-cut, and branch-and-price.

2.2.1 Linear programming and the simplex method

The development of linear programming (LP) has been ranked among the most important scientific advances of the mid-20th century [95]. LP uses a mathematical model to describe a problem and solving it requires finding the extremum of a linear combination of variables.

A conventional procedure for solving LP problems is the simplex method, which was developed by George Dantzig in 1947 [50] and has proved to be an efficient method for solving LP problems. The underlying concepts of the simplex method are geometric, where each constraint can be interpreted as a boundary line and the points of intersection are corner-point solutions. When points fall into a feasible region, they become corner-point feasible (CPF) solutions. An augmented solution is a solution for the original variables that has been augmented by the corresponding values of the slack variables. A basic feasible solution is an augmented CPF solution. The simplex method is an iterative algorithm with the following steps (see Figure 2.1):

The processes start by finding an initial CPF solution. The iterative procedure selects an *entering variable* (i.e. the variable enters the set of basic variables) using an optimality condition and selects a *leaving variable* (i.e. the variable being replaced leaves the set of basic variables) using a feasibility condition. The pivot column represents the line (plane) that

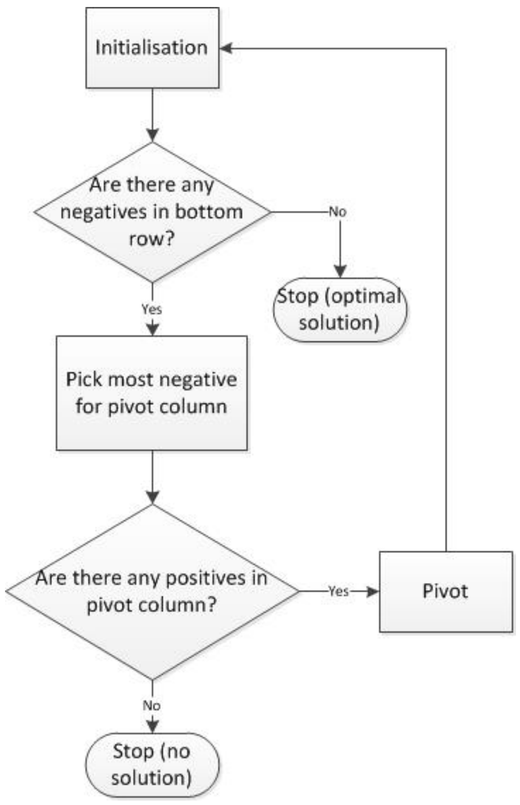
Initialisation
Are there any negatives in bottom row?
No
Stop (optimal solution)
Yes
Pick most negative for pivot column
Are there any positives in pivot column?
Yes
Pivot
No
Stop (no solution)

Figure 2.1: Structure of simplex method.

is leaving and the pivot row represents the line (plane) that is entering. After that, the algorithm checks if the current CPF solution is optimal or not. If yes, the optimal solution is obtained, otherwise, another iteration is performed to find a better CPF solution. This is repeated until it is clear that the current CPF can't be improved for a better objective value. In this way, the optimal solution is achieved.

A key step in solving a linear program with the simplex method is the pivot selection. Methods that make this selection are generally known as a pivot rules. The two major goals of a pivot rule are: prevent cycling between states and enhance the speed of search by choosing good edges to traverse. Good choices can lead to a significant speedup in finding a solution to a linear program, while poor choices lead to very slow or even nonterminal progress. Dantzig rule is a widely used pivot method.

Take the Dantzig rule as an example: Let a linear program be given by a canonical tableau. In Figure 2.1, the initialisation stage converts the problem into an initial system by adding slack variables. Then the initial system is transformed into a tableau. The value of the objective function is decreased if the pivot column is selected so that the corresponding entry in the objective row of the tableau is positive. If all the entries in the objective row are less than or equal to 0 then no choice of entering variable can be made and the solution is optimal. The pivot process computes the ratios between the non-negative entries in the right hand side and the positive entries in the pivot column. The pivot row is the row with the smallest non-negative ratio. Pivoting occurs where the pivot row and pivot column meet.

In addition to the simplex method, there are other (or variants of the simplex method) algorithms for linear programming, such as the dual simplex method [128], parametric linear programming [179], and interior-point algorithm [112].

2.2.2 Dynamic programming

Dynamic programming is an important tool for making a sequence of interrelated decisions. The term was introduced by Richard Ernest Bellman in 1953 [12]. In contrast to the simplex method, which is formed by standard mathematical formulations, dynamic programming is only a general program solving concept. Dynamic programming can be applied in many fields, such as mathematics, management science, economics, and computer science for solving complex problems.

The basic idea of dynamic programming is to break a complex problem into a collection of subproblems and solve each subproblem with a recursive method. It is a useful technique for making a sequence of interrelated decisions when the problem can be solved in "stages". The decisions made in a stage will affect the decisions at the subsequent stages but are dependent on each other. One usually follow a backward recursive approach where the make decisions from the end stages rather than from the beginning stages. Intermediate solutions are usually saved in a memory-based data structure (e.g. a table) so that the low efficiency of repetitive computation in a recursive method is avoided. A dynamic programming algorithm looks up and compares the previously solved solutions until the best solution is found.

Stage, state, decision variable and criterion of effectiveness parameters are common for every dynamic programming problem. Please refer to [13] for more detailed information about dynamic programming. A high degree of ingenuity is required to design a recursive algorithm that solves problems efficiently. Generally, dynamic programming fails to solve large instances within a reasonable computation time, owing to the recursive structure of the algorithm.

2.2.3 Branch-and-bound

Similar with dynamic programming, branch-and-bound only examines a fraction of the feasible solutions. Branch-and-bound and its variations have been successfully applied to a variety of combinatorial problems, especially integer programming problems. This method was first proposed by A.H. Land in 1960 [119]. Branch-and-bound, as its name suggests, branches a complex problem into smaller and smaller subproblems and checks the bound of each subset, then discards the subset if the bound indicates it cannot contain an optimal solution so that only a small fraction of the feasible solutions need be examined.

Branching, bounding and fathoming are three basic steps in this algorithm. The **branching** step divides (branches) a problem into a tree of subproblems, its nodes corresponding to candidate solutions. This step can be conducted by setting the branching variable at a fixed value or to specify a range of values if the number of branching variables is greater than two. The **bounding** step computes a bound on how good the solution of a subproblem can be. Conventional methods for doing this are LP relaxation and Lagrangian relaxation [69]. The **fathoming** step dismisses a subproblem from further consideration if it indicates that the result cannot contain an optimal solution or its relaxation has no feasible solutions. Please refer to [95] and [207] for detailed information about this algorithm.

2.2.4 Branch-and-cut

After the development of the branch-and-bound approach, another breakthrough in operations research is the introduction of the branch-and-cut method in the 1980s. In fact, cutting plane (cut) algorithms for integer programming problems were first proposed by Gomory (1963) [84], but the algorithms were proven ineffective and have since fallen out of use [138].

Branch-and-cut is the branch-and-bound algorithm combined with cutting planes, which is also called row generation. A cut for any linear programming program is a new linear constraint that tightens a feasible region of the linear programming relaxation of a branch-and-bound search. Consequently, the performance of the branch-and-bound algorithm is improved. Within the branch-and-cut process, if an optimal solution to an LP relaxation is infeasible, violated inequalities are found and added to the LP to cut off the infeasible solution. After that, the LP is re-optimised. Branching occurs when no violated inequalities can be found [11].

Branch-and-cut was initially limited to pure binary integer programming problems, but was soon extended to general linear programming problems where some or all the variables are restricted to integer values. Several types of cutting plane algorithms have been developed, such as generalised comb inequalities [145] and disjunctive inequalities [9]. The procedures for identifying violated inequalities also vary by problem, for example, the knapsack problem [48] and traveling salesman problem [152]. More information about cutting planes and implementation details can be found in [138].

2.2.5 Branch-and-price

Branch-and-price focuses on column generation rather than row generation. Pricing and cutting are complementary procedures for tightening a feasible region.

The branch-and-price approach integrates branching together with column generation guided by a pricing problem. Since only a small number of variables contribute to obtaining the objective value, a subproblem (pricing problem) is solved to identify the variables worthy of further processing. This procedure is referred to as column generation. If such columns are found, the LP is re-optimised until no more such columns can be found. The branching process occurs as the column generation does not automatically guarantee an integer solution. Cutting planes can also be added in order to further strengthen the relaxation, and this method is called branch-price-and-cut (see [56] for a detailed description of the branch-price-and-cut algorithm).

In the branch-and-price framework, the original problem is decomposed into a master problem and subproblems. This decomposition scheme is different according to various contextual settings. Application of branch-and-price has been particularly fruitful in the areas of routing and scheduling [11]. In many vehicle routing applications solved by column generation, the subproblem is usually viewed by a shortest path problem with resource constraints (SPPRC) or one of its variants. The elementary shortest path problem with resource constraints (ESPPRC) is an extension of the SPPRC in which all paths are elementary. The ESPPRC is to find the least cost elementary path with no repeated nodes between two specified nodes such that the accumulated quantity of each resource consumed on all arcs in the path does not exceed its limit [17].

Standard exact approaches for solving the SPPRC and its variants are dynamic programming, labelling algorithms, branch-and-bound, Lagrangean relaxation and constraint programming. Among them, dynamic programming is used extensively for generating columns and it is shown to be successful with tight resource constraints, but it becomes time consuming with increasing problem size. Heuristics are adopted for efficiency in solving the SPPRC approximately. If a graph contains negative costs, which always occur in the context of column generation, solving the SPPRC might become more complicated [67]. Extending the SPPRC to an ESPPRC exponentially increases the problem size and the ESPPRC is proven to be NP-hard [60].

The SPPRC was first introduced in the Ph.D dissertation of Desrochers as a subproblem of a bus driver scheduling problem [103]. The classic optimisation approach for the VRPTW was given in Desrochers et al. (1992) [54]. Feasible columns were added as needed by solving a SPPRC by considering time window and capacity constraints using dynamic programming. The algorithm is capable of optimally solving 100-customer problems. D. Feillet et al. (2004) [67] proposed an exact solution procedure for the E-SPPRC, which extended the classical label correcting algorithm originally developed by Desrochers et al. (1988) [55] for the relaxed (non-elementary) path version of this problem.

Rousseau et al. (2004) [170] presented a column generation approach that solved the elementary shortest path subproblem with constraint programming. A more efficient label setting algorithm for solving ESPPRC with consideration of the state-space augmenting approach was proposed by Boland et al. (2006) [17], an idea based on Kohl (1995) [111]. Righini et al. (2007) [167] compared dynamic programming to the branch-and-bound method and developed a decremental state-space relaxation method. The implementation and comparison of [17] and [167] can be found at [162].

Irnich et al. (2005) [103] provided a review on the subject. The review proposed a classification and a generic formulation for SPPRCs, and also briefly discussed complex modelling issues involving resources, and presented the most commonly employed SPPRC solution methods. A more recent survey of SPPRCs can be found at [163].

2.3 Heuristics and Metaheuristics

We introduce several exact algorithms that are guaranteed to find the optimal solution in a finite amount of time. These methods are well-studied and have proven to be valuable in solving many real-life problems, particularly smaller ones. For large and complex problems, finding optimal solutions with exact methods is prohibitively time-consuming. In some cases, it is not important to find the optimal solution as long as the computation time is reasonable and the solution quality is satisfied. For such cases, people often resort to heuristic or metaheuristic methods.

A heuristic method (heuristic) can be defined as: *"a procedure that is* *likely to discover a very good feasible solution, but not necessarily an optimal solution for the specific problem being considered [95]."* Typical examples of heuristics are the constructive method, greedy algorithm and local search algorithm. Heuristics are also referred to as problem-dependent techniques.

Metaheuristics, on the contrary, are problem-independent techniques, as they are general solution methods for a whole range of problems. A metaheuristic can be defined as *"a general solution method that provides both a general structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problem [95]."* The main framework of metaheuristics are problem-independent but the actual algorithm implementation is problem dependent (e.g. parameters, neighbourhoods, etc). According to the number of solutions, (meta)heuristics can be separated into two classes: single-point and population-based. Single-point algorithms only keep a single solution at each iteration, while population-based algorithms maintain a population of solutions. Typical examples of single-point search methods are the basic local search, tabu search, simulated annealing, and variable neighbourhood search, while the population-based approaches include genetic algorithms, evolutionary strategies, evolutionary programming, genetic programming, differential evolution, ant colony optimisation and particle swarm optimisation. Metaheuristics are deemed as advanced heuristics. For example, the simulated annealing technique may accept a temporary deterioration of the solution in order to explore a larger solution space.

Hyperheuristics are a step ahead of metaheuristics. Their main purpose is to *"devise new algorithms for solving problems by combining known heuristics in ways that allow each to compensate, to some extent, for the weaknesses of others. They might be thought of as heuristics to choose heuristics [169]."* Differing from the applications of metaheuristics that usually work with a search space of solutions, hyperheuristics work with a search space of heuristics [80]. A more detailed description of hyperheuristics can be found at [169].

In the following, some popular heuristic and metaheuristic methods are introduced.

2.3.1 Basic local search

Local search is an iterative algorithm that starts with an initial solution and constantly replaces the current solution with a better neighbourhood solution until the stopping condition is met or no further improvement can be found. A neighbourhood of a point is a set of points containing that point where one can move some amount away from that point without leaving the set. The initial solution can be generated randomly or by a certain constructive heuristic (i.e. a heuristic that starts from an empty solution and is gradually constructed until a full solution is obtained). Neighbourhood solutions are usually achieved by moves that transform the current solution by neighbourhood function(s). The most commonly used stopping conditions are maximum CPU time and maximum number of iterations. Local search is a commonly used heuristic that improves an initial solution by a set of local changes, and during the process only one neighbourhood is employed. This local search heuristic may fall into the trap of local optimum. When the choice of the neighbourhood solution is based on a maximising objective, local search is also known as *hill-climbing* (or *descent* for the minimising objective).

2.3.2 Tabu search

The idea of tabu search (TS) was initially proposed by Glover [77] in 1977 but it was formally introduced by Glover and McMillan in 1986 [78]. A few years later, Glover further investigated TS (e.g. in 1989 [78] and 1990 [79]). TS is a single-point metaheuristic search method that includes the local search procedure used for solving combinatorial problems. It is able to enhance the performance of local search, as it uses memory structures to store the visited solutions in order to avoid the search process cycling back to the recently visited local optimum. TS also allows the search to continue by accepting non-improving moves from the neighbourhood of the local optimum. Hansen et al. (1986) [89] proposed a similar idea to TS and named it the steepest ascent mildest descent approach.

Tabu list is a short-term memory that records a limited number of full or partial solutions (or tabu moves) that are prohibited from being revisited. *Tabu tenure* restricts the number of iterations of each tabu move. A long-term memory could also serve to record solution attributes for diversification and intensification in order to explore different neighbourhoods. There are many methods for implementing these two functions (see [178]). An example of conducting diversification and intensification is to give penalties or incentives to the attributes of some solutions in the process of move selection. *Aspiration criteria* is adopted to accept promising moves even if they are restricted by the tabu list. TS can also incorporate some advanced concepts, such as surrogate and auxiliary objectives. Please refer to [76] for more details.

2.3.3 Simulated annealing

Simulated annealing (SA) is another metaheuristic that allows the search process to escape from the local optimum. This algorithm was proposed by Metropolis et al. in 1953 [135] and the idea originated from the annealing process in metallurgy. Similar with TS, SA is a single-point metaheuristic search method that not only always accepts improving solutions, but also non-improving solutions depending on a specified probability. The probability is denoted by P and is calculated by the following equation:

$$p = e^{c/t} > r \tag{2.1}$$

where c denotes change (i.e. difference between current and candidate objective values). t is the temperature, which measures the tendency to accept the candidate. r is a random number between 0 and 1. As in metallurgy, the SA search process starts with a relatively large value of t in order to increase the value of P, so that the search goes in relatively random directions. With the cooling of t, the search continues but the P decreases gradually. Therefore, the choice of t controls the degree of randomness. The selection of appropriate starting temperature, cooling schedule (e.g. linear, geometric function), neighbours and stopping criteria is necessary in order to tune an SA. For more on SA please see [180], [122] and [216].

2.3.4 Variable neighbourhood search

As discussed, the basic local search heuristic may fall into the trap of local optimum. In order to avoid this, different strategies are adopted for TS and SA. In contrast, the variable neighbourhood search (VNS) systematically exploits the idea of neighbourhood change, both in the descent to local minima and in the escape from the valleys that contain them [140]. The VNS algorithm is a single-point metaheuristic that contains an initialisation and an iteration step. According to [90], the initialisation step involves the selection of a set of neighbourhood structures $N_k, k = 1, ..., k_{max}$, finding an initial solution x and choosing a stopping condition. The iteration step contains three processes to change neighbourhood: *shaking*, *local search*, and *move decision*. The shaking function is a diversification process that generates a solution randomly from the k_{th} neighbourhood $x' \in N_k(x)$. The local search process applies a local search method with x' as the initial solution, while the obtained local optimum solution is denoted as x''. The move decision process decides whether to move the search to the new incumbent x''. In addition to the basic VNS that employs the first improvement method with randomisation strategy, there are many variants and extensions of the VNS: variable neighbourhood descent (VND), reduced VNS (RVNS), and variable neighbourhood decomposition search (VNDS). More detailed information about VNS variants and their applications can be found in [92], [93] and [91].

2.3.5 Genetic algorithms

Genetic algorithm (GA) was first developed by Fraser (1960) [70]. They belong to population-based metaheuristics, meaning that rather than processing a single solution at each iteration, GAs work with a set of solutions simultaneously. Similar to SA, which is inspired by natural phenomena, GAs originate from the *theory of evolution*. This theory suggests that offspring with advantageous mutations are more likely to survive in order to cope with the environment. The process is also referred to as *survival of the fittest*. In terms of applications in operations research, a feasible solution corresponds to an offspring (i.e. *individual* while a number of individuals are called *population*) and the natural *fitness* is equivalent to objective value. Therefore, GAs tend to generate improving solutions. More specifically, through iterative evolutions, the current surviving population passes their genes (features by *crossover* functions) to their children (new candidate solutions) who may share the merits of their parents. In addition, advantageous mutations may occur in children so that they process the feature that their parents do not have and are more likely to survive and spawn the next generation. However, in order to increase diversity among individuals, the less fit offsprings could also be selected to survive and act as parents in future generations. This step can be achieved by selection strategy such as *tournament selection* and *roulette wheel selection*. The algorithm stops once the termination criteria are satisfied.

More specifically, the search space of a problem in GA is represented as a collection of *individuals*. These individuals are represented by character strings (or matrices), which are often referred to as *chromosomes*. The fitness of an individual is measured with an evaluation function. The part of the search space to be examined is called the population. Normally, a genetic algorithm works as follows: First, an initial population is chosen, and the quality of each individual in this population is determined through a fitness function. Next, at each iteration, children are generated form some selected parents through recombinations and mutations. Newly created individuals are subject to mutations at a small probability, that is, they will change some of their heriditary distinctions. After that, both the parents and children go through a selection process to determine whether they can enter to the next generation successfully based on a selection criterion.

In order to implement a GA, one of the most important decisions is to decide the solution encoding scheme. Common examples include binary encoding, path representation, adjacency representation, ordinal representation and matrix representation. The path representation is probably the most natural representation of a tour for vehicle routing problems. However, the biggest problem of this encoding is the presence of repeating/missing nodes after crossovers. A feasibility repair procedure is often required to restore the feasibility.

The operators which generates offsprings are called the crossover operator and the mutation operator. Mutation and crossover play different roles in the genetic algorithm. Crossover helps the evolution to pass beneficial features to the next generation. Mutation is applied to explore new states and helps the algorithm to gain new features that may not exist in parents. By choosing appropriate crossover and mutation operators, the probability that the genetic algorithm converges to a near-optimal solution in a reasonable number of iterations is increased. There are many types of crossover operators that have been designed for various encoding schemes but one-point, two-point and uniform crossover are commonly used. To implement one-point crossover, a point on both parents' strings (depends on encoding scheme) is selected, a string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent. Two-point crossover calls for two points to be selected on the parent strings, string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent. The uniform crossover uses a mixing ratio between two parents, unlike one- and two-point crossover, genes are partially copied from both parents. Common methods of implementing the mutation operator in permutation problems are swaps, inversions, and scrambles.

There can be various criteria for stopping a GA. For example, if it is possible to determine previously the number of iterations needed. But the stopping criteria should normally take into account the uniformity of the population, the relationship between the average objective function with respect to the objective function of the best individual, as well as not producing an increase in the objective function of the best individual during a fixed number of cycles. For more details about the implementation of GAs please see [95], [81], [203] and [114].

Population diversity is an important issue in the theory of natural selection and it indicates the difference in structures of individual in a population. It is crucial to the GAs ability to continue fruitful exploration as it may be used in choosing an initial population, in defining a stopping criterion, and in making the search more efficient throughout the selection of crossover operators or the adjustment of various control parameters (e.g., crossover or mutation rate, population size) [150]. Normally, GA are robust when the population contains more various individuals (i.e. high population diversity), as it will encourage the exploration phase of the GA search and prevent the population from converging prematurely to local optima [94]. Population diversity is commonly defined by a diversity metric which is measured by features such as the individual fitness values, structures, or the combination of the two [25].

A large amount of work has been devoted to diversity measures, which includes early study of variance of fitness and uncertainty. Recently, other measures such as evolution history, distance and measures in the genotype and phenotype space are also introduced [223]. The genotypic diversity measures the structural differences between individual genotypes, while the phenotypic diversity measures the differences in individual phenotypes [10].

Population Diversity can be maintained by a means of ways such as fitness sharing, deterministic crowding, self-adapting mutation rates, etc. A recent survey about maintenance of diversity can be found in [87].

2.3.6 Swarm intelligence metaheuristics

According to Blum et al. (2008) [15], swarm intelligence (SI) is the discipline that *"concerns the design of an intelligent multi-agent system by taking inspiration from the collective behaviour of social insects such as ants, termites, bees, and wasps, as well as from other animal societies such as flocks of birds or schools of fish."* SI refers to a general set of algorithms, such as particle swarm optimisation [106], ant colony optimisation [58], bee colony optimisation [104], bat algorithm [212], etc. The ant colony optimisation (ACO) algorithm is a typical example of the SI family.

The ACO algorithm was proposed in 1991 by Marco Dorigo in his PhD thesis [57] for solving combinatorial optimisation problems. When seeking food, ants communicate with each other by using pheromones. Initially they travel independently and lay down pheromone trails from the food source to their colony. If other ants find the pheromone trails, instead of traveling at random, they tend to move along the path of pheromones. The pheromone trails evaporate over time. But during that time, the ants travel shorter paths to the food and leave pheromone scents. Eventually, the ants find the shortest path to their food.

ACO algorithms have been successfully applied to many combinatorial optimisation problems (e.g. [200] and [146]). ACO can easily to be implemented to handle dynamic operations. For instance, when the graph changes during the search, the ACO algorithm is changed accordingly to adapt to the latest graph. Please refer to [182], [58], [59] and [64] for more details about ACO algorithms, their history and applications in dynamic operations. In addition to ACOs, there are many other SI-based algorithms. The basic concepts of all these algorithms originate from the intelligence of different swarms in nature, such as: particle swarm optimisation [106], artificial bee colony algorithm [104] and glowworm swarm optimisation [113].

2.4 Hybridising Exact Methods and Metaheuristics

Recently, many works have been implemented by the hybridisation of optimisation approaches. Initially, research mainly focused on hybridisation of metaheuristics. For example, although the greedy randomised adaptive search procedure (GRASP) is considered a single-point algorithm, it can cooperate with path-relinking, which requires maintaining a population of solutions. Nowadays, an increasing number of hybridisations between metaheuristics with exact approaches are being developed. These methods are able to provide good results as they adopt advantages from both types of methods.

Dumitrescu et al. (2003) [61] investigated local search approaches that are strengthened by the use of exact algorithms. This study demonstrates that hybridisation can serve to solve some linear programming subproblems in order to reduce the search space for the local search algorithm. The subproblems can be the relaxations of integer programming models, and the optimal solutions of these subproblems are then used to define the search space for the local search algorithm. Hybridisation is especially useful when the neighbourhoods are very large so that the subproblems can guide neighbourhood moves.

Raidl et al. (2005) [160] classified hybridisation of exact algorithms and (meta)-heuristics into four types shown in Figure 2.2. Collaborative combinations are high-level combinations as the algorithms are not part of each other, while integrative combinations are low-level combinations because one algorithm is embedded within another one.

We briefly introduce the four types of hybridisation and give examples for each:

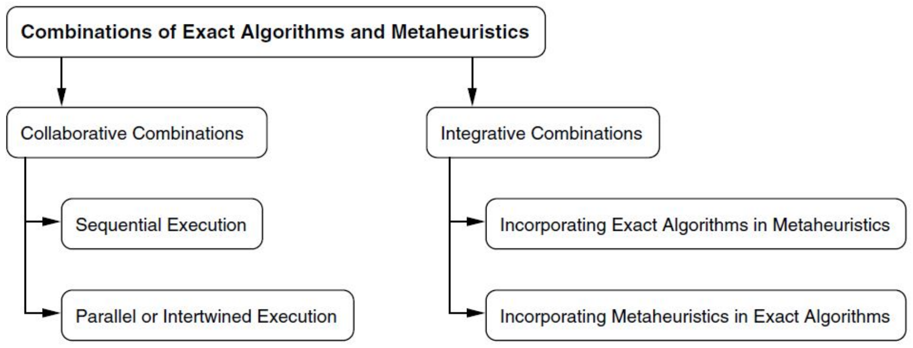
Combinations of Exact Algorithms and Metaheuristics
Collaborative Combinations
Integrative Combinations
Sequential Execution
Parallel or Intertwined Execution
Incorporating Exact Algorithms in Metaheuristics
Incorporating Metaheuristics in Exact Algorithms

Figure 2.2: Major classification of exact/metaheuristic combinations [160].

Collaborative Combinations - sequential execution: In this type of hybridisation, either the metaheuristic is executed before the exact method, or vice-versa. For example, when solving a set covering problem, a heuristic is used to generate a set of feasible columns and the exact method is used to find an optimal solution from the feasible columns. This type of hybridisation has been successfully applied in solving a variety of difficult combinatorial optimisation problems such as: travelling salesman problems [110], production line scheduling problems [39], and multidimensional knapsack problems [193].

Collaborative Combinations - parallel or intertwined execution: Instead of executing either metaheuristics or exact methods sequentially, this type of method implements the algorithms in a parallel or intertwined way. Cluster or multi-processor is used to deploy the parallel implementations. There are several frameworks proposed to facilitate such implementations: A-Teams [184], TECHS [52], and MALLBA [5]. These frameworks have been utilised in many combinatorial optimisation problems, such as: job scheduling problems solved by A-Teams [33], fuel optimisation problems solved by TECHS [141], and multidimensional knapsack problems solved by MALLBA [5]. More recently applications can be found at [195] and [115].

Integrative Combinations - incorporating exact algorithms in (meta)-heuristics: In this type of hybridisation, exact algorithms are subordinately embedded within (meta)heuristics. For example, the solution of LP-relaxation and its dual values can be utilised in heuristically guiding neighbourhood search, mutation, and local improvement. This type of combination has been implemented to solve multi-constrained knapsack problems [37], glass cutting problems [161], and linear assignment problems [130].

Integrative Combinations - incorporating (meta)heuristics in exact algorithms: This type of hybridisation is analogous with the previous one, but (meta)heuristics are embedded within exact algorithms. For example, (meta)heuristics can be used to determine bounds in branch-and-bound algorithms. In addition, in the branch-and-price approach, (meta)heuristics are often used to search for columns with negative costs. Applications of this hybridisation method include graph colouring problems [165], 2D bin packing [159], and vehicle routing problems [172].

As mentioned, the main motivation behind the hybridisation of (meta)-heuristics with exact algorithms is to exploit the advantages of both types of methods. Developing an effective hybrid method is generally a difficult task, as it requires knowledge of both (meta)heuristics and exact methods. Moreover, Blum et al. (2011) [16] shows that a hybridisation strategy might work well for specific problems, but performs poorly for others. Jourdan et al. (2009) [101] proposed taxonomy of hybridisation methods. A more recent review of the hybridisation approach can be found at [16].

2.5 Summary

This chapter gives a general overview of the current popular optimisation techniques which may be promising for the optimisation of the truck drayage problem studied in this thesis. Exact methods, (meta)heuristic methods, and hybridisations of them are reviewed. Some methods germane to this thesis are emphasised.

NP-hard problems are difficult to solve and no polynomial time algorithms are known for solving them optimally. In fact, a majority of combinatorial optimisation problems, such as bin packing, knapsack and vehicle routing problems, are NP-hard. Small instances are usually solved by exact algorithms, such as linear programming, dynamic programming, branch-and-bound, branch-and-cut and branch-and-price.

Exact algorithms can solve a problem optimally, but they are also known to be time expensive and not recommended to be adopted in solving large instances. In order to effectively solve large instances, researchers tend to consider (meta)heuristics. One commonly used classification of metaheuristics is to divide them into two main categories: single-solution algorithms and population-based algorithms. For example, the first category contains local search, tabu search, simulated annealing, and variable neighbourhood search, while the population-based approaches includes genetic algorithms, ant colony optimisation and practice swarm optimisation. Recently, many works have been implemented by cooperation (or hybridisation) of exact and (meta)heuristic approaches.

Previous research related to multi-shift full truckload vehicle routing problems (MFTLVRPs) is introduced and reviewed in the next chapter.

Chapter 3

The Multi-shift Full Truckload Vehicle Routing Problem and Its Literature Review

3.1 Introduction

Real-life vehicle routing problems exhibit a high degree of complexity, owing to various constraints concerning carrier capacity, route duration, time windows (subject to either customers or carriers), and compatibility between customer, carrier and commodity. This chapter firstly present the problem concerned in this thesis. Secondly, we introduce terms and models of several typical vehicle routing problems. Thirdly, we explained where our work situated within the broader field of freight transportation research. Finally, relevant vehicle routing problems that sharing similar properties of our research and their solution methods are highlighted and reviewed.

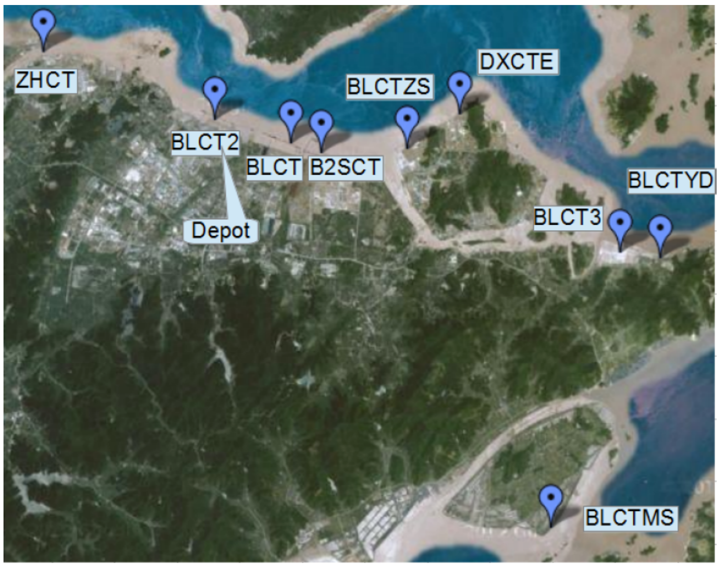

Figure 3.1: Positions of the container terminals in the port of Ningbo.

3.2 Problem Introduction

This study concerns the operations of container transshipments between nine different docks located in Port of Ningbo. The objective is to minimise the total vehicle travel distance for transporting a large number of non-consolidatable commodities (containers) between a relatively small number of nodes (docks), satisfying various time window constraints concerning commodities and drivers. Typical time window constraints are the available time and deadline for commodities and work shifts for drivers. Thanks to internationally adopted EDI (Electronic Data Interchange) systems and GPS (Global Positioning Systems) sensors, the available and deadline times of commodities are generally known 1 to 2 days in advance with some tolerable estimation errors. The physical locations of different nodes are shown in Figure 3.1.

The problem has the following characteristics:

- The unit size of the commodity is equivalent of that of the trucks. Therefore one unit of a commodity is shipped directly to its destination without transfers or consolidations.

- Schedules are shift based. Each truck has to go back to a depot for driver changeovers after a shift due to labour law related regulations. For this particular problem, a shift is 12 hours. A schedule typically spans from 4-8 shifts in order to maximise the efficiency.

- All docks are within a short distance of each other and a unit of any commodity could be completed within a shift. The service time at each node (see Table 3.1) (loading time at the source or unloading time at the destination of a shipment) is comparable to the truck travel times between nodes. Note that in practice, travel times are fluctuating with the time of day and will be estimated in Chapter 5.

- The time window for each commodity (i.e. from time when the commodity becomes available to the time when it has to be delivered to its destination) varies considerably from 1-2 hours, up to 6 shifts.

- The total quantities of all the commodities within a planning horizon can be very large (up to 2000) but the number of distinct physical nodes is relatively small (less than 10).

Table 3.1: Container service time on the ports

Port name	Load time	Unload time
BLCT	30	30
BLCT2	30	30
BLCT3	30	30
BLCTYD	40	40
BLCTZS	60	60
DXCTE	5	5
BLCTMS	60	60
ZHCT	180	180
B2SCT	5	5

3.3 Typical Vehicle Routing Problems and Their Models

As there are many variants of the basic problems owing to differences in real-life cases and each of them may also have diverse modelling formulations, we select those that have received a relatively greater amount of attention in the literature. In general, the formulations can be divided into: vehicle flow formulations, commodity flow formulations and set-partitioning based formulations.

3.3.1 VRP

The VRP introduced by Dantzig et al. (1959) [51] is one of the most s-tudied problems in the field of operations research. Although the original basic problem has been proposed for over five decades, enthusiasm for VRP research has not waned and numerous variants of VRP have been proposed in the literature (see [189], [41], [121] and [120]). A more recent taxonomy for VRP literature can be found in [63], [116], [108] and [23].

Vehicle routing problem (VRP) is a generic name given to a whole class of problems in which a fleet of vehicles based at one or several depots have to determine a set of routes for a number of geographically dispersed customers [1]. The objective of VRP is to satisfy the demands of customers at minimal cost. VRP is a well-known NP-hard problem of integer programming, as the computational effort required to optimally solve VRP increases exponentially with problem size.

The difficulty of solving VRPs, for example capacitated vehicle routing problem (CVRP) (see Section 3.3.1), is rooted in the overlapping of two well-studied combinatorial problems: travelling salesman problems (TSPs) and bin packing problems (BPPs). Both are NP-hard. A TSP can be defined as: a salesman who leaves from the source of product (depot) that has to visit $n-1$ number of cities (exclude the depot) to sell product, visit each city exactly once, and eventually return to the depot. The objective is to minimise total travel distance. A TSP is also an NP-hard problem that can be viewed as the simplest version of a VRP involving only one uncapacitated vehicle. A BPP, another NP-hard problem, can be defined as: a set of items with different volumes that has to be packed into a number of bins so that the number of bins used is minimal. A feasible solution for the VRP is one or several TSP tours satisfying bin packing constraints (i.e. the total demand along each of the arcs joining successive copies of the depot does not exceed vehicle capacity) [1]. Since both TSPs and BPPs are NP-hard, CVRP are more difficult to be solved than either TSPs or BPPs.

The formulation of TSP and BPP are given below:

TSP

The travelling salesman problem (TSP) can be defined on a graph $G = (A, V)$ where A is the arc set and V is the set of nodes (e.g. city, customer), $\{0\}$ is the depot. Let c_{ij} be the travelling cost (or distance) between an arc(i, j), $i, j \in V$. x_{ij} is the decision variable and it equals to 1 if arc (i, j) is included in the solution and 0 otherwise. The formulation [222] and [49] of TSP can be described as following:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.1}$$

subject to

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \tag{3.2}$$

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \tag{3.3}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \leq |S| - 1, 2 \leq |S| \leq n - 1 \quad \forall S \subset V \setminus \{0\}, S \neq \emptyset \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V, i \neq j \tag{3.5}$$

The objective (3.1) is to minimise total travel cost (or distance) and visit all customers. Constraint (3.2) and (3.3) ensure any node is visited exactly once. Constraint (3.4) is a sub-tour breaking constraint.

BPP

Given a set of bins L, each bin is associated with a size $l_j \in L$. Let V denote the set of items and each having a size $v_j \in V$. z_j is a decision variable, it equals to 1 if bin j is used and 0 otherwise. Another decision variable is

x_{ij}, its value is 1 if the item i is packed into the bin j and 0 otherwise. The formulation [222] of BPP can be described as following:

$$\min \sum_{j \in L} z_j l_j \tag{3.6}$$

subject to

$$\sum_{i \in V} v_i x_{ij} \leq l_j \quad \forall j \in V \tag{3.7}$$

$$\sum_{j \in L} x_{ij} = 1 \quad \forall i \in L \tag{3.8}$$

$$\sum_{i \in V} x_{ij} - M z_j \leq 0 \quad \forall j \in V \tag{3.9}$$

$$z_j \in \{0,1\} \quad \forall j \in V \tag{3.10}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in L, \forall j \in V \tag{3.11}$$

The objective (3.6) is to pack all the items into a minimal number of bins. Constraint (3.7) ensures that the total size of items should not exceed the capacity of each bin. Constraint (3.8) impose that each item must be packed into exactly one bin. Constraint (3.9) indicates if any item packed in a bin, then the bin is taken. M is a sufficiently large number. Let $d(v) = \sum_{j \in V} v_j$ denotes the total demand of the item set. Assuming that each bin is associated with a capacity C, then the trivial BPP lower bound can be obtained by:

$$\lceil d(v)/C \rceil \tag{3.12}$$

CVRP

For a single depot VRP, if the vehicle fleet only has capacity constraints, then the problem is also referred to as a capacitated vehicle routing problem

(CVRP). The CVRP is the most-studied problem of the VRP family. In the CVRP, vehicles are identical, based at a depot and, as the name suggests, capacity restrictions on vehicles are imposed. Given a graph $G = (A, V)$ where A is the arc set and V is the set of customers. Let c_{ij} be travelling cost between an arc(i, j), $i, j \in V$. x_{ij} is the decision variable and it equals to 1 if the arc (i, j) is included in the solution and 0 otherwise. K is the set of vehicles, each with capacity C. The formulation of this problem can be given as following [189]:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{3.13}$$

subject to

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\} \tag{3.14}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\} \tag{3.15}$$

$$\sum_{i \in V} x_{i0} = K \quad \forall j \in V \setminus \{0\} \tag{3.16}$$

$$\sum_{j \in V} x_{0j} = K \quad \forall i \in V \setminus \{0\} \tag{3.17}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \leq |S| - r(S) \quad \forall S \subset V \setminus \{0\}, S \neq \emptyset \tag{3.18}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V, i \neq j \tag{3.19}$$

The objective (3.13) is to minimise the total vehicle travel distance and satisfy all transportation demand of customers. Constraint (3.14) and (3.15) ensure that exactly one arc enters and leaves each node. Constraint (3.16) and (3.17) impose that for each route, only one arc enters and leaves the depot. (3.18) is a well-known generalized subtour elimination constraint. $r(S)$ denote the minimum number of vehicles used to serve set S and it can be obtained by the trivial BPP lower bound (3.12). However, solving (3.18) is practically hard as it has a cardinality growing exponentially in the n. Alternatively, a series of constraints that have similar function as (3.18) have been proposed, such as the sub-tour elimination constraint presented below:

$$u_i - u_j + Cx_{ij} \leq C - d_j \quad \forall i, j \in V \setminus \{0\}, i \neq j \tag{3.20}$$

$$d_i \leq u_i \leq C \quad \forall i \in V \setminus \{0\} \tag{3.21}$$

u_i denote the load of a vehicle after visiting customer i.

3.3.2 VRPTW

Given a set of customers, vehicles, and goods, to solve a vehicle routing problem with time windows (VRPTW) is to find a set of routes of minimal total length that start and end at a depot, such that each customer is visited by exactly one vehicle while also satisfying a number of constraints, such as time window of visit and vehicle capacity. Time window constraint could be either hard or soft. In hard time window constraint, a driver is allowed to wait in case of early arrival but late arrival is not allowed. In soft time window constraint, a driver is allowed to wait in case of early arrival but will be punished for late arrival.

The VRPTW is defined on the network $G = (V, A)$, where V is the set of nodes and A is the set of arcs between nodes. A nonnegative cost, c_{ij} is associated with each arc $(i, j) \in A$, which is usually defined as the *Euclidean distance* between a node i and j, but it can be also defined as other cost such as consumption of gasoline, travel time etc. The depot is denoted by two nodes 0 and n+1. A vehicle route is constructed by the set of arcs starting from node 0 and ending at node n+1. Each node is also associated with a time window $[a_i, b_i]$, where a_i denotes the earliest possible departure time at node i and b_i represents the latest possible arrival time at the node i. s_i denote the service time of the node i. Let E and L be the earliest possible departure time and latest possible arrival time at the depot respectively, that is, $[a_0, b_0] = [a_{n+1}, b_{n+1}] = [E, L]$. d_i represents transportation demands of the node i and C is the capacity of a vehicle. K denotes the vehicle set, decision variables $x_{ijk}, (i,j) \in A, k \in K$ equals to 1 if arc (i,j) is used by the vehicle k and 0 otherwise. Decision variables $w_{ik}, i \in V, k \in K$ indicates the starting time of service at the node i when it is serviced by the vehicle k.

The formulation [189] of VRPTW can be described as the following commodity flow formulation with time window and capacity constraints:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} C_{ij} X_{ijk} \tag{3.22}$$

subject to

$$\sum_{k\in K}\sum_{\{j\in V|(i,j)\in A\}} x_{ijk}=1 \quad \forall i\in N \tag{3.23}$$

$$\sum_{\{0\in V|(0,j)\in A\}} x_{0jk}=1 \quad \forall k\in K \tag{3.24}$$

$$\sum_{\{i\in V|(i,j)\in A\}} x_{ijk}-\sum_{\{i\in V|(j,i)\in A\}} x_{jik}=0 \quad \forall k\in K,\forall j\in N \tag{3.25}$$

$$\sum_{\{i\in V|(i,n+1)\in A\}} x_{i,n+1,k}=1 \quad \forall k\in K \tag{3.26}$$

$$x_{ijk}(w_{ik}+s_i+t_{ij}-w_{ij})\le 0 \quad \forall k\in K,(i,j)\in A \tag{3.27}$$

$$a_i\sum_{\{j\in V|(i,j)\in A\}} x_{ijk}\le w_{ik}\le b_i\sum_{\{j\in V|(i,j)\in A\}} x_{ijk} \quad \forall k\in K,\forall i\in N \tag{3.28}$$

$$E\le w_{ij}\le L \quad \forall k\in K, i\in\{0,n+1\} \tag{3.29}$$

$$\sum_{i\in N} d_i\sum_{\{j\in V|(i,j)\in A\}} x_{ijk}\le C \quad \forall k\in K \tag{3.30}$$

$$x_{ijk}\in\{0,1\} \quad \forall k\in K,(i,j)\in A \tag{3.31}$$

The objective (3.22) is to minimise the total vehicle travel distance and satisfies all the transportation demand of customers and various time windows and capacity constraints. Constraint (3.23) restricts that each customer is assigned by only one vehicle. Constraint (3.24) and (3.26) restrict each vehicle route starting from and ending at the depot. Constraint (3.25) concerns commodity flows. Constraints (3.27) (3.28) and (3.29) are the time window constraints regarding to schedule feasibility. Constraint (3.30) limit vehicle capacities.

3.3.3 m-TSPTW

A travelling salesman problem with time windows (TSPTW) is a VRPTW involving only one uncapacitated vehicle, while a multiple travelling salesman problem with time windows (m-TSPTW) belongs to the special case of VRPTW involving m uncapacitated vehicles. The m-TSPTW is a well-known NP-hard problem; when the number of vehicles is fixed it becomes an NP-complete problem [139]. We can obtain the commodity flow formulation of the m-TSPTW by simply eliminating the capacity constraint 3.30 from the VRPTW formulation presented in Section 3.3.2. In this section, we present an alternative formulation that applied for a container truck transportation problem [219]. The objective of this formulation is to minimise the total travel time.

Given a network graph $G = (V_D, V_C, A)$, where V_D is the depot node set and V_C is the container node set. V_C is the collection of container transportation activities mainly including the activities of pickup and drop off containers. V_D represents the nodes that vehicles have to initially leave from and finally return back. When there is only one depot node, then the problem falls into the m-TSPTW. The problem is called m-TSPTW with multiple depots when V_D contains more than one depot. Initially, each depot i has n_i vehicles. Let $[T_A(i), T_B(i)]$ denote the time window of the node $i \in V_C$ and $T(i,j)$ be the amount of time consumed by the arc (i,j). $T(i)$ is the start time of node $i \in V_C$. Let y_i represents the time when a truck has to serve the container node $i \in V_C$. Decision variable x_{ij} is 1 if arc (i,j) included in a route and 0 otherwise. The problem can be formulated as a mixed integer programming model as following.

$$\min \sum_{(i,j)\in A} T(i,j)x_{ij} \tag{3.32}$$

subject to

$$\sum_{j \in V_C} x_{ij} \leq n_i \quad \forall i \in V_D \tag{3.33}$$

$$\sum_{j \in V_D \cup V_C} x_{ij} = \sum_{j \in V_D \cup V_C} x_{ji} = 1 \quad \forall i \in V_D \tag{3.34}$$

$$\sum_{i \in Z, j \in Z} x_{ij} \leq |Z| - 1 \quad \forall Z \subseteq V_C, Z \neq \phi \tag{3.35}$$

$$T_A(i) \leq y_i \leq T_B(i) \quad \forall i \in V_C \tag{3.36}$$

$$y_i + T(i) + T(i,j) - y_i \leq (1 - x_{ij})M \quad \forall i \in V_C, j \in V_C \tag{3.37}$$

$$x_{ij} \in 0, 1 \quad \forall (i,j) \in A \tag{3.38}$$

$$y_i : real \quad variable \quad \forall i \in V_C \tag{3.39}$$

The objective function (3.32) is to minimise the total travel time. Constraint (3.33) imposes the number of vehicles. (3.34) is the flow conservation constraint that ensures that exactly one arc enters and leaves any container node. (3.35) eliminates sub-tours among container nodes. Z is the subset of V_C and ϕ is the empty set. (3.36) is the time window constraint. Constraint (3.37) updated the start time along the route, where M is a sufficiently large number.

3.3.4 VRPPD

A vehicle routing problem with pickup and delivery (VRPPD) can be defined as: A set of vehicles based at one or multiple depots (terminals) that has to satisfy a number of transportation requests (picking up goods or people) from a pick-up point and transport them to a delivery point. The transportation of people is also referred to as a dial-a-ride problem. The objective is to finish all requests and minimise total routing costs of vehicles.

Similar with the VRPTW investigated in Section 3.3.2, a VRPPD involves several types of constraints, including: 1. Each customer is allowed to be visited exactly once. 2. Vehicle capacity constraints. As many practical applications of the VRPPD consider time intervals of service that may be visited by a vehicle, it is thus useful to present a more general variant of the problem, which is VRPPD with time windows (VRPPDTW).

Let K denote the set of vehicles. As not all vehicles can service all requests, each vehicle is associated with a set $N = P \cup D$ where P is the set of pick up nodes and D is the set of delivery nodes, $P = \{1, ..., n\}, D = \{n+1, ..., 2n\}$. Let $o(k)$ and $d(k)$ be the origin and destination of a vehicle k. Each vehicle is defined on a network $G_k = (V_k, A_k)$, where the V_k ($V_k = N_k \cup \{o(k), d(k)\}$) is the set of nodes and A_k is the set of arcs between the nodes. Let C_k represents the capacity of a vehicle. Let t_{ijk} and c_{ijk} denote the travel time and cost from node i to j by vehicle k respectively. Each node is also associated with a time window $[a_i, b_i]$, where a_i denotes the earliest possible departure time at node i and b_i represents the latest possible arrival time at node i. s_i denotes the service time of node i. l_i represents the transportation demand of node i.

Decision variables $x_{ijk}, (i, j) \in A, k \in K$ equals to 1 if arc (i, j) is used by the vehicle k and 0 otherwise. Decision variables $T_{ik}, i \in V, k \in K$ indicates the starting time of service at the node i when it is serviced by the vehicle k. L_{ik} provides the load after a vehicle finish service at the node i.

The formulation [189] of VRPPDTW can be described as following:

$$\min \sum_{k \in K} \sum_{(i,j) \in A_k} C_{ij} X_{ijk} \tag{3.40}$$

subject to

$$\sum_{k \in K} \sum_{j \in N_k \cup \{d(k)\}} x_{ijk} = 1 \quad \forall i \in P \tag{3.41}$$

$$\sum_{j \in N_k} x_{ijk} - \sum_{j \in N_k} x_{j,n+i,k} = 0 \quad \forall k \in K, i \in P_k \tag{3.42}$$

$$\sum_{j \in P_k \cup \{d(k)\}} x_{o(k),j,k} = 1 \quad \forall k \in K \tag{3.43}$$

$$\sum_{i \in N_k \cup \{o(k)\}} x_{ijk} - \sum_{i \in N_k \cup \{d(k)\}} x_{jik} = 0 \quad \forall k \in K, j \in N_k \tag{3.44}$$

$$\sum_{i \in D_k \cup \{o(k)\}} x_{i,d(k),k} = 1 \quad \forall k \in K \tag{3.45}$$

$$x_{ijk}(T_{ik} + s_i + t_{ijk} - T_{jk}) \le 0 \quad \forall k \in K, (i,j) \in A_k \tag{3.46}$$

$$a_i \le T_{ik} \le b_i \quad \forall k \in K, i \in V_k \tag{3.47}$$

$$T_{ik} + t_{i,n+1,k} \le T_{n+i,k} \quad \forall k \in K, i \in P_k \tag{3.48}$$

$$x_{ijk}(L_{ik} + l_j - L_{jk}) = 0 \quad \forall k \in K, (i,j) \in A_k \tag{3.49}$$

$$l_i \le L_{ik} \le C_k \quad \forall k \in K, i \in P_k \tag{3.50}$$

$$0 \le L_{n+i,k} \le C_k - l_i \quad \forall k \in K, n+i \in D_k \tag{3.51}$$

$$L_{o(k),k} = 0 \quad \forall k \in K \tag{3.52}$$

$$x_{ijk} \in \{0,1\} \quad \forall k \in K, (i,j) \in A_k \tag{3.53}$$

The objective function (3.40) is to minimise the total vehicle travel cost. Constraints (3.41) and (3.42) ensure that each request of pickup and delivery is conducted by one and by the same vehicle. Constraints (3.43), (3.44) and (3.45) are commodity flow constraints imposing that each vehicle start from and return back to the depot. Constraint (3.46) and (3.47) are the time window constraints with respect to schedule feasibility. Constraint (3.48) is the precedence constraint that forces the pickup node to be serviced before the delivery node. Constraint (3.49) related with the compatibility requirement between vehicle loads and routes. (3.50) and (3.51) are the vehicle capacity constraints. Constraint (3.52) express the initial vehicle load at the depot.

3.3.5 SPP of VRP

The SPP is usually considered as the master problem if SPP is solved by a decomposition (e.g. column generation) method. Let R denotes the collection of all the feasible routes (or a set of feasible routes maintained by a column generator) of the network graph $G = (V, A)$. Construction of R depends on the practical problem. For example, for the VRPTW (see Section 3.3.2), R is the set of routes with schedules satisfying constraints (3.24), (3.26), (3.27), (3.28), and (3.29). Each route $r \in R$ is associated with a cost c_r. In addition, let a_{ri} be a binary constant that takes value 1 if route r includes node i and takes value 0 otherwise. The binary variable y_r is equal to 1 if the route r is used in the solution and 0 otherwise.

The formulation [55] of SPP can be described as following:

$$\min \sum_{r \in R} c_r y_r \tag{3.54}$$

subject to

$$\sum_{r \in R} a_{ri} y_r = 1 \quad \forall i \in V \tag{3.55}$$

$$\sum_{r \in R} y_r = K \quad \forall r \in R \tag{3.56}$$

$$y_r \in \{0, 1\} \quad \forall r \in R \tag{3.57}$$

The objective function (3.54) is to minimise the total vehicle travel cost. Constraint (3.55) ensures that each node i is visited only once by the selected routes. Constraint (3.56) imposes that K routes are selected.

3.3.6 FTPDPTW

A full truckload pickup and delivery problem with time windows (FT-PDPTW) can be viewed as an extension of VRPPDTW (see Section 3.3.4). In FTPDPTW, a vehicle carries a single load (full truckload). A fleet of vehicles has to complete assignments of pick-up and delivery container pairs with minimal traveling costs and satisfy various time window constraints. Depending on practical problems, the traveling costs include fixed vehicle costs (e.g. maintained cost of truck and payment to driver) and variable costs (e.g. gasoline consumption that is proportional to the travel distance). In this section, we present a particular case of formulation that applied for drayage of containers [28]. This model was specifically designed for drayage operations arising in the context of intermodal container transportation, but can also be applied to other problems involving full truckload transportation as well. The objective of this formulation is to minimise total cost including the fixed and travel costs of serving all customers.

The FTPDPTW is defined on a graph $G = (V_0, A)$, where V_0 is the node set, $\{0\}$ is the depot, $V_0 = V \cup \{0\}$. Let V be the customer set that includes delivery customers V_D and pickup customers V_P, that is $V = V_D \cup V_P$. The other parameters are summarised in Table 3.2.

Table 3.2: The list of notations.

Input Parameters	
K	the set of trucks, $k \in K$
C_{ijk}	the travelling cost of arc (i, j) by truck k.
FC_k	fixed cost of truck k for each route.
E_i	earliest possible start time of node i.
L_i	latest possible start time of node i.
s_i	starting service time of node i.
d_{ij}	travel time of arc (i, j).
Decision Variables	
x_{ijk}	equals to 1 if arc (i, j) is used by vehicle k and 0 otherwise.
b_i	actual starting service time at node i.

The formulation of FTPDPTW can be described as the following [28]:

$$\min \sum_{i \in V_0} \sum_{j \in V_0, i \neq j} \sum_{k \in K} C_{ijk} X_{ijk} + \sum_{k \in K} (FC_k \sum_{j \in V} x_{0jk}) \tag{3.58}$$

subject to

$$\sum_{j \in V_0} \sum_{k \in K} x_{ijk} = 1 \quad \forall i \in V \tag{3.59}$$

$$\sum_{i \in V_0} x_{ijk} - \sum_{i \in V_0} x_{jik} = 0 \quad \forall i \in V, k \in K \tag{3.60}$$

$$E_i \leq b_i \leq L_i \quad \forall i \in V \tag{3.61}$$

$$\sum_{k \in K} x_{ijk}(b_i + s_i + d_{ij} - b_j) \leq 0 \quad \forall i \in V, j \in V \tag{3.62}$$

$$\sum_{k \in K} x_{0jk} d_{0j} \leq b_j \quad \forall j \in V \tag{3.63}$$

$$x_{0jk}(b_i + s_i + d_{i0} - b_j) \leq 0 \quad \forall i \in V, k \in K \tag{3.64}$$

$$\sum_{j \in V} x_{0jk} \leq 1 \quad \forall k \in K \tag{3.65}$$

$$x_{ijk} \in \{0, 1\} \quad \forall i, j \in V_0, i \neq j, k \in K \tag{3.66}$$

$$b_i \geq 0 \quad \forall i \in V \tag{3.67}$$

The objective function (3.58) is to minimize total routing cost including the fixed and travel cost and satisfy all customers. Constraint (3.59) imposes that each node is visited only once. (3.60) is the flow conservation constraint. (3.61) and (3.64) are the time window constraint of customer location and total travel time of route k. Constraints (3.62) and (3.63) ensure the time consistency of b_i. Constraint (3.65) enforces that each vehicle is used at most once.

3.3.7 SNDP

A service network design problem (SNDP) is usually used to solve tactical planning of freight operations, ensuring optimal utilisation of resources. Tactical planning of operations mainly involve: the selection of service type, specification of terminal and routing of freight [45]. Typical applications of SNDP include less-than-truckload (LTL) problems, express package delivery, and container transshipment problems. Consolidations are widely adopted in order to maximise the utilisation of logistic resources.

Different from the VRPs discussed above, the focus of tactical planning is more on long-haul transportation, which is freight transportation between cities, firms and organisations by trains, trucks, barges or any combination of those modes. In SNDP, carriers usually establish regular service routes and adjust their characteristics to satisfy the expectations of the most number of customers. The service thus cannot be tailored for each customer individually as a carrier usually groups freight of different customers into certain service types [45]. Furthermore, VRP focuses on door-to-door transportation, while freights are consolidated at and moved between terminals by various service types in the transportation system of SNDP. The transportation system is constructed by a complex network of terminals, which are connected by physical and conceptual links.

A particular frequency-based path formulation is defined on a graph $G = (N, A)$, where N is a set of nodes and A is a set of arcs. A given set of service routes R is available. The set of all paths from origin to destination throughout this network is defined as P. There are two types of decision variables x_p^k(called a shipment flow variable) and y_r^f(design variable). There are also two indicators α_{ij}^r and β_i^r. The parameters used in the model are defined in Table 3.3:

Table 3.3: The list of notations.

Input Parameters	
K	the set of commodities.
$o(k)$	the origin of commodity, $k \in K$.
$d(k)$	the destination of commodity, $k \in K$.
s	the customer service type, $s \in S$.
d^k	the demand that has to be transported from origin $o(k)$ to destination $d(k)$ within the time window of its customer service type $s \in S$.
f	the fleet type, $f \in F$.
u^f	the capacity of fleet type, $f \in F$.
R	the set of feasible service routes.
P	the set of paths in the network.
R^f	the set of routes that serviced by fleet type $f \in F$.
α_{ij}^r	equals to 1 if $arc(i, j)$ is part of the route $r \in R$, 0 otherwise.
β_i^r	equals to 1 if route $r \in R$ starts from i and -1 when $r \in R$ ends in i.
h_r^f	the fixed costs of running the service route $r \in R$ by vehicle type $f \in F$.
c_p^k	the costs of delivering the one unit of commodity $k \in K$ in path $p \in P$.
Decision Variables	
x_p^k	the flow of commodity k is transported through path $p \in P$.
y_r^f	the number of vehicles used to run the service route $r \in R$.

The model can be formulated as following [205]:

$$\min \sum_{f \in F} \sum_{r \in R^f} h_r^f y_r^f + \sum_{k \in K} \sum_{p \in P^k} c_p^k x_p^k \tag{3.68}$$

subject to

$$\sum_{k\in K}\sum_{p\in P^k} x_p^k \leq \sum_{f\in F}\sum_{r\in R^f} u^f y_r^f \alpha_{ij}^r \quad \forall (i,j)\in A \tag{3.69}$$

$$\sum_{p\in P^k} x_p^k = d^k \quad \forall k\in K \tag{3.70}$$

$$\sum_{r\in R^f} \beta_i^r y_r^f = 0 \quad \forall i\in N, f\in F \tag{3.71}$$

$$x_p^k \geq 0 \quad \forall k\in K, p\in P^k \tag{3.72}$$

$$y_p^f \geq 0 \quad and \quad integer \quad \forall r\in R^f, f\in F \tag{3.73}$$

The objective (3.68) of the model is to finish all commodity transportation demand at minimal total cost including the fixed and variable flow cost. Constraint (3.69) restricts commodity flows. Constraint (3.70) imposes that the demand of each commodity is satisfied. Constraint (3.71) ensures the balance of demands for each fleet type. This is one of the generic models of SNDP, for others please refer to [45] and [205].

3.3.8 PVRP

Periodic vehicle routing problems (PVRPs) work on a planning horizon that usually cover several days where each customer is required to be visited at least once (or during a specified time window, then the problem becomes PVRP with time windows) within a planning horizon. A PVRP is different from a VRP, as the planning horizon is more than a single period. PVRPs arise naturally from real-world applications, such as waste collection. The problem can be formally defined on a multigraph $G = (V, A)$, where V is the node set ($V = v_0, v_1, ..., v_n$) and A is the arc set $(A = v_i, v_j)^{kl}$. Each customer i is also associated with a visit combination set C_i. k and l are the index of vehicle and day of visit respectively. c_{ijkl} denotes the cost of vehicle $k \in K$ travel from node i to node j on day $l \in L$. x_{ijkl} is 1 if vehicle k visits customer j immediately after visiting customer i during day l. a_{rl} is set to 1 if day l belongs to visit combination $r \in C_i$, and y_{ir} is set to 1 if visit combination $r \in C_i$ is assigned to customer i. The problem can be formally defined as follows [40]:

$$\min \sum_{i,j \in V} \sum_{k \in K} \sum_{l \in L} c_{ijkl} x_{ijkl} \tag{3.74}$$

subject to

$$\sum_{r \in C_i} y_{ir} = 1 \quad \forall i \in V \tag{3.75}$$

$$\sum_{i \in V} \sum_{k \in K} x_{ijkl} - \sum_{r \in C_i} a_{rl} y_{ir} = 0 \quad \forall i \in V, \forall l \in L \tag{3.76}$$

$$\sum_{i \in V} x_{ihkl} - \sum_{j \in V} x_{hjkl} = 0 \quad \forall i, h \in V, \forall l \in L \tag{3.77}$$

$$\sum_{j \in V} x_{0jkl} \leq 1 \quad \forall k \in K, \forall l \in L \tag{3.78}$$

$$\sum_{i,j \in V} q_i x_{ijkl} \leq Q_k \quad \forall k \in K, \forall l \in L \tag{3.79}$$

$$\sum_{i,j \in V} (c_{ijkl} + d_i) x_{ijkl} \leq D_k \quad \forall k \in K, \forall l \in L \tag{3.80}$$

$$\sum_{v_i, v_j \in S} x_{ijkl} \leq |S| - 1 \quad \forall k \in K, \forall l \in L, S \subseteq V \setminus \{0\}, |S| \geq 2 \tag{3.81}$$

$$x_{ijkl} \in \{0, 1\} \quad \forall i, j \in V, \forall k \in K, \forall l \in L \tag{3.82}$$

$$y_{ij} \in \{0, 1\} \quad \forall i \in V, \forall r \in C_i \tag{3.83}$$

The objective of this model is to satisfy all customer demands with minimal cost. Constraint (3.75) restricts that every feasible combination must be assigned to each customer. Constraint (3.76) imposes that customers are visited only during corresponding days. Constraint (3.77) indicates when vehicle arrives at a customer on a given day, it also leaves at customer on the same day. Constraint (3.78) ensures that one and only one vehicle is used every day. Constraint (3.79) and (3.80) restrict vehicle capacity and route duration. (3.81) is a sub-tour elimination constraint.

3.3.9 Multi-shift container transshipment formulation

More recently, Chen (2016) [30] proposed a multi-shift container transshipment formulation for the study concerned in this thesis and solved it by hyperheuristic method. The formulation is task based and inspired by the formulations given by Wang et al. (2002) [208] who proposed a m-TSPTW based model for full truckload transportation. The model is described below.

In this model, a task is defined as a standard transportation volume that represents the process of transporting full truck load from their source to their destination. All container transportation tasks are converted into a node set N with node 0 and $n+1$ denoting the depot of vehicle k. The fleet size is denoted by K. Each route starts from 0 and ends with $n+1$ (same for all vehicles). $S = \{1, 2, \dots, s, \dots, |S|\}$ is the consecutive shifts considered in the problem and Y_s and Z_s is the time window of shift s. Let d_{ij} denote the cost of deadheading (i.e. the movement of commercial vehicles in non-revenue mode for logistical reasons) from node i to node j and t_{ij} denote the travelling time from i to j. T_i is the truck arrival time at node i, and l_i is the time needed to complete the task represented by i, including the loading time, transportation time and unloading time. Binary variable X_{ijs} indicates whether a task is performed on the arc (i, j) at the shift s. Each node i has a time window (a_i, b_i) that constrains the service time for node i.

Using the above notations, this problem is formulated as a cost minimising problem:

$$\sum_{i\in N}\sum_{j\in N\setminus\{i\}}\sum_{s\in S} d_{ij}X_{ijs} \tag{3.84}$$

subject to

$$\sum_{s\in S}\sum_{i\in N\setminus\{j\}} X_{ijs} = 1, \quad \forall j \in N \setminus \{0, n+1\}. \tag{3.85}$$

$$\sum_{i\in N\setminus\{j\}} X_{ijs} = \sum_{m\in N\setminus\{j\}} X_{jms}, \quad \forall j \in N \setminus \{0, n+1\}, s \in S. \tag{3.86}$$

$$X_{ijs}(T_i + l_i + t_{ij} - T_j) \leq 0, \quad \forall (i, j) \in A, s \in S. \tag{3.87}$$

$$X_{i0s} = 0, \quad \forall s \in S, i \in N \tag{3.88}$$

$$X_{(n+1)js} = 0, \quad \forall s \in S, j \in N \tag{3.89}$$

$$\sum_{j\in N\setminus\{0\}} X_{0js} = K, \quad \forall s \in S. \tag{3.90}$$

$$\sum_{i\in N\setminus\{n+1\}} X_{i(n+1)s} = K, \quad \forall s \in S. \tag{3.91}$$

$$a_i \leq T_i \leq b_i - l_i, \quad \forall i \in N \setminus \{0, n+1\}. \tag{3.92}$$

$$X_{ijs}(Y_s + t_{0j}) \leq X_{ijs}T_i \leq X_{ijs}(Z_s - t_{i(n+1)} - l_i),$$
$$\forall i, j \in N \setminus \{0, n+1\}, s \in S. \tag{3.93}$$

$$X_{ijs} \in \{1, 0\}, \quad \forall i \in N, j \in N, s \in S, i \neq j. \tag{3.94}$$

The objective is to minimise the total cost due to empty truck movement. Constraint (3.85) states that each task is served exactly once. Constraint (3.86) specifies that a task may only be serviced after the previous task is finished. Constraints (3.85) and (3.86) together ensure arcs in different shifts are not connected with each other. Tasks' time constraint (3.87) enforces the correct temporal relationship between consecutive tasks. Constraints (3.88) and (3.89) prohibit movements back to 0 after a truck starts or movements out from $n+1$ so that the correct route structure is generated. Constraints (3.90) and (3.91) are the fleet size constraints that ensure the correct number of trucks are used in the solutions. Constraint (3.92) limits the visiting time of tasks to their time windows. Constraint (3.93) ensures all start time of tasks are within the shift time window. Finally, constraint (3.94) defines the domain of the decision variables X_{ijs}.

3.4 Related Works of The MFTLVRP

The study of freight transportation concerns planning and modelling transportation of commodities under a number of customer requirements. The planning of a transport system can be classified into three levels: strategic, tactical and operational planning [205]. Strategic planning is also referred to as long-term planning, where decisions are made on the design and construction of a physical network, such as infrastructure and locations of terminals. Tactical planning is also referred to as medium-term planning that concerns the design of transportation networks. In contrast to strategic planning, the study of tactical planning concentrates on optimal utilisation of resources to provide transportation services to customers, firms, and organisations. This problem can be usually modelled as a service network design problem (S-NDP, see Section 3.3.7). Operational planning is also known as short-term planning performed by local management, yard masters and dispatchers [45]. The study of operational planning includes the scheduling of crews, shipments and vehicles. Please see Crainic et al. (e.g. [44], [47], [46]) for more about freight transportation problems.

Freight transportation can be also classified into full truckload transport (FTL), less-than truckload transport (LTL) and express deliveries [205]. Despite numerous research studies on freight transportation, most of them have focused on consolidation based transportation (LTL and express deliveries) and research on the FTL problem is somewhat limited. Among all the FTL problems, container transport is a special case of the full truckload transport problem since containers are both shipment commodities and transport resources [20]. The container transportation industry is under fierce competition and pressure to improve its efficiency and reduce energy use and increasingly more studies have been devoted to the optimisation of operations at container terminals (see [186], [196], [164], [209] and [217] for recent examples). Intermodal container transportation experienced rapid development since the 1980s [131]. In intermodal container transportation, the connections between ports, docks or terminals mainly rely on truck drayage, which is also known as drayage problem. In this thesis, we study a multi-shift inter-dock container forwarding problem, using data from a real-life problem faced by the Port of Ningbo. The problem can be formally defined as MFTLVRP and it is common for any drayage problem with multiple docks being operated simultaneously.

As mentioned, the study of MFTLVRP is relatively limited despite numerous publications on other types of freight transport problems. To help understand the core features of this problem, it is possible to broadly classify freight transportation into **consolidated** transport and **non-consolidated** transport problems. In consolidated transportation, freights can be split and shipped via multiple paths of a service network and freights may get transferred and consolidated along some of the paths. That is, during the process, some nodes in the service network act as hubs or consolidation centres and a package may be transported by multiple vehicles before arriving at its destination. In non-consolidated freight transportation, freight is delivered to its destination directly, in its entirety, by a single vehicle. For example, in CVRP, a vehicle contains many freights and deliver each freight to its destination. The role of the vehicle is merely to deliver freights even though it contains (not consolidated defined here) many freight within the limitation of vehicle capacity. As each freight is delivered to its destination directly and in its entirety by a single vehicle, CVRP falls into non-consolidated category.

For both the consolidated and non-consolidated transports, the shipment of freight can be **single-directional** or **bi-directional**. In single-directional transport, each node in the transportation network is either a **supply** node or a **demand** node but not both while the nodes in a bi-directional transport can both be supplies as well as demands. Table 3.4 gives typical examples for each type of transportation problems.

Single-directional consolidated transportation includes the classical production logistics (in which necessary raw materials, parts and sub-parts of products are consolidated and transported to the production factories) and deliveries for fresh produce and hazardous materials that require special vehicles.

Single-directional, non-consolidated transport problems are studied intensively in the forms of several vehicle routing problem variants [189], including capacitated vehicle routing problem (CVRP, see Section 3.3.1), vehicle routing with time windows (VRPTW, see Section 3.3.2), multi-depot vehicle routing problem (MDVRP), periodic vehicle routing problem (PVRP, see Section 3.3.8), etc. With regards to bidirectional, consolidated transportation, service network design research [8] for less-than-truckload transport, express delivery and postal mail delivery are typical examples. In terms of the search space and complexity, this category probably represents the most challenging freight transportation problem due to the huge size of the search space.

The final category is the bidirectional, non-consolidated transportation. Typical examples include vehicle routing with pickup and delivery [136], full truckload transport [126], and container transport which is a special case of the full truckload transport problem. The problem that is considered in this study falls into this final category of Table 3.4.

In real-life freight transport, the time window of deliveries can fall in a wide planning horizon, which could result in very large problem size. In some problems, the deliveries can be partitioned into several classes depending on the urgency of deliveries (e.g. half-day, day, two-day parcel delivery). The planning horizon is thus structured and can be partitioned into multi-

Table 3.4: A possible classification of freight transportation problems.

	Consolidated	Non-consolidated
Single-directional	Production supply chain, Fresh produce delivery, Hazards transport, etc.	CVRP, VRPTW, TSP, Multi-depot VRP, PVRP
Bidirectional	Service network design, Postal mail delivery, Less-than truckload transport (LTL), Express delivery	Vehicle routing with pickup and delivery, Full truckload transport (FTL), Container transport (Drayage operations), Dial-a-ride problem, etc.

periods. This type of problem is referred to as a periodic vehicle routing problem (PVRP).

In MFTLVRP, each commodity has an operation time window defining its availability time and the delivery deadline. Time constraints require that both the pickup and delivery operations occur within this time window for a commodity. In many other vehicle routing problems such as pickup and delivery problem with time windows problem (PDPTW), two separate time windows are used, one for pickup and the other for delivery. Note that for non-time critical full truckload transportation, having one time window is reasonable since all the terminals (nodes) operate all the time, and having short time windows for both pickup and delivery does not make sense, although it is very different for express deliveries which are mostly for household customers.

To sum up, MFTLVRP belongs to the operational planning level of freight transportation. It falls into bidirectional and non-consolidated category in Table 3.4. Most studies are focusing on single-shift planning horizon, while this problem shares some similarity with periodic vehicle routing problem which considers a multi-shift planning horizon.

The MFTLVRP should has the following characteristics:

- The unit size of the commodity (container) is equivalent of that of the trucks. Thus this is full truck load problem.
- Schedules are shift based. Thus this is a multi-shift vehicle routing problem.
- All docks are within a short distance of each other and a unit of any commodity could be completed within a shift.
- Each commodity has an operation time window defining its availability time and the delivery deadline.

3.5 Existing Approaches and Experimental Studies

The remainder of this section provides a review of existing research work for the final bidirectional, non-consolidated transportation research with a special focus on the container transportation problems.

3.5.1 Vehicle routing problem with pickup and delivery

The vehicle routing with pickups and deliveries (VRPPD) differs from classic VRP problems (see Section 3.3.1) in that some of the nodes are both demand and supply nodes and the flow of the freight at these nodes is, therefore, bidirectional (both incoming and outgoing). The inherent mixed loading capacity constraints in VRPPD often lead to increased computational complexity. A comprehensive review for VRPPD can be found in Berbeglia et al. (2007) [14]. Research by Min et al. (1989) [136] represents one of the first scientific studies on VRPPD. The problem was abstracted from a public library distribution system in Ohio, US and a simple three-phased procedure that resembles the well-known "cluster-first, routing-second" heuristics was developed and compared against the real-world manual solution. Pisinger et al. (2007) [156] proposed a generic adaptive large neighbourhood search (ALNS) metaheuristic for 4 variants of the VRP problems with competitive results reported for all variants. The proposed ALNS shares many common features to the simulated annealing hyperheuristics (Bai et al. (2012) [8]) that was shown successful for the coursework timetabling and the well-known bin packing problem. Gabriel et al. (2010) [88] studied a variant of VRPPD in which the pickups are selective while deliveries are compulsory. A branch-and-price algorithm was developed which could solve instances containing up to 50 customers optimally. Derigs et al. (2012) [53] studied a real-life full truckload routing problem arising in timber transportation and used a multilevel neighbourhood search method to solve the problem. Liu et al. (2013) [127] studied a vehicle scheduling problem encountered in home health care logistics. A genetic algorithm and a tabu search method were proposed for this problem. The method was tested on the benchmarks for the VRP with mixed backhauls and time windows (VRPMBTW) against existing best solutions and obtained solutions that are better than the best-known solutions in the literature. Pandelis et al. (2013) [153] studied capacitated VRPPD in which finite and infinite-horizon single VRP with a predefined customer sequence and pickup and delivery is considered. A special-purpose dynamic programming algorithm that determines the optimal policy was developed. Zhang et al. (2014) [220] studied time dependent vehicle routing problems with simultaneous pickup and delivery by formulating this problem as a mixed integer programming model. A hybrid algorithm that integrates an ant colony algorithm and a tabu search method was developed and the computational results suggest that the hybrid algorithm outperforms stand-alone ant colony algorithm and tabu search. Chen et al. (2014) [32] studied the routing problem with unpaired pickup and delivery with split loads for fashion retailer chains. However, the common time window constraints are missing. Both a simple heuristic and a variable neighbourhood search method were proposed. More recently, Avci et al. [] studied a variant of the classical vehicle routing problem, Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). A simple adaptive local search algorithm (HLS) combines a parameter-free Simulated Annealing (SA) based approach with Tabu Search (TS) have been proposed. It can be seen that due to the NP-Hard nature of the problem, almost all studies adopted metaheuristics to solve large scale problem instances.

Another type of vehicle routing with pickup and delivery problem that has been studied specifically is the dial-a-ride (DAR) problem. Kirchler et al. (2013) [109] proposed a fast algorithm for solving the static Dial-a-Ride Problem (DARP). A granular tabu Search method has been applied for the first time to solve this kind of problem. Paquette et al. (2013) [155] developed a multi-criteria heuristic embedding a tabu search process in order to solve DARPs combining cost and quality of service criteria. This is the first study that handles more than two criteria for this type of problem. Ferrucci et al. (2014) [68] introduced dynamic pickup and delivery problem with real-time control (DPDPRC) in order to map urgent real-world transportation services. A tabu search algorithm was proposed and computational result showed that newly arriving requests, traffic congestion, and vehicle disturbances can be efficiently handled by this approach. Braekers et al. (2014) [21] considered a multi-depot heterogeneous dial-a-ride Problem (MD-H-DARP) in real life. A exact branch-and-cut algorithm and a deterministic annealing metaheuristic were developed for solving small and large problems respectively.

3.5.2 Bidirectional full truckload transport

In bidirectional full truckload transport, commodities are shipped to destinations in their entirety without intermediate stops or transhipment. Therefore it is different from VRPPD since some of the commodities in VRPPD go through intermediate nodes before reaching their destinations. Truck container transport (in intermodal transport, the truck container transport within short distance is also referred as drayage operations) is a typical example of such a problem since the size of a container is usually equivalent to the capacity of the truck (hence full truckload) and flow of containers (either loaded or empty) can be bidirectional. Therefore, the solution methods for VRPPD cannot directly be used for the truck container transport problem or even if some variants are applicable, their performance will not be as good since important features (e.g. no intermediate stops) are not exploited fully in the algorithms designed for VRPPD. Even for the truck container transport problems, different characteristics will lead to different problems. From the no-free-lunch theorem of [206], we know that it is very hard to develop a generic algorithm, performing best for all possible instances. In Section 3.3.6, we also reviewed full truckload pickup and delivery problem with time windows (FTPDPTW), which is an extension of VRPPD with time windows that a vehicle carries full truckload. FTPDPTW is usually applied in the context of intermodal container transportation.

Zhang et al. (2010) [219] proposed a nonlinear model (see Section 3.3.3) based on a preparative graph for container transportation between shippers, receivers, depots and terminals. A solution method was designed by improving the time window partitioning scheme used in Wang et al. (2002) [208] for a multiple travelling salesman problem with time windows (m-TSPTW). The empirical results for a set of randomly generated instances indicate that improved performance can be achieved compared with a reactive tabu method in Zhang et al. (2009) [171]. The method is effective for small instances but may suffer for large scale problems since the size of the graph can explode with increase in the number of shipments and nodes. Similar issues exist for instances with very wide time windows (e.g. time windows that spans over a few days) due to the time partitioning scheme adopted in the method.

Nossack et al. (2013) [149] presented a new formulation for the truck scheduling problem based on a Full-Truckload Pickup and Delivery Problem with Time Windows and propose a 2-stage heuristic solution approach. The results of computational experiments indicate that their 2-stage heuristic outperforms the time window partitioning method applied by Zhang et al. (2010) [219] in terms of computational efficiency.

Braekers et al. (2013) [22] investigated a 2-stage deterministic annealing algorithm for a full truckload transport problem with simultaneous pickup and delivery nodes. The problem was formulated as an asymmetric m-TSPTW. Similarly, the problem was tested for a set of randomly generated instances with commodity time windows ranging between 60 to 240 min, which is much smaller than those in our problems. Better results were obtained using the algorithm than those given by the method of Zhang et al. (2010) [219]. Most research studies assumed a constant travel time among the transportation network which is not always realistic. Therefore, Braekers et al. (2012) [19] studied how time-dependent travel times will affect the full truckload transport planning and scheduling, in which the optimal departure times become decision variables in addition to the routing variables. In real-life of drayage operation, shippers may request empty containers to be delivered while consignees may have empty containers available to be picked up.

By considering this, Braekers et al. (2014) [21] studied vehicle routes performing all loaded and empty container transports in the service area of one or several container terminals during a single day. A bi-objective approach (minimising the number of vehicles and minimising total distance travelled) is considered and it is shown that this method obtained considerably better results than those reported in Braekers et al. (2013) [22]. Sterzik et al. (2013) [181] proposed a single-shift general model for transporting both full and empty containers among multiple nodes (depots, terminals and customers). A tabu search heuristic is developed and tested on instances that contain up to 5 depots, 3 terminals and 75 loads with an one-day planning horizon.

The drayage operations problem is a typical case of bidirectional full truckload vehicle routing problems. Here, we summaries the relevant research on the drayage operation problems which we broadly classify into three fields: drayage operations with and without relocation requirements of empty containers, and drayage operations with dynamic situations (e.g. either the origin or the destination of the container transport is partially unknown in advance; thus resulting in a dynamic travel time of trucks).

Drayage problem without relocation of empty containers

Wang et al. (2002) [208] model a full truckload pickup and delivery problem with time windows (FT-PDPTW) as an asymmetric multiple travelling salesman problem with time windows (am-TSPTW) and propose a time-window discretisation scheme. Similarly, Gronalt et al. (2003) [85] treat FT-PDPTW as am-TSPTW and develop four different savings based heuristics. Jula et al. (2005) [96] extend the am-TSPTW model with social constraints and propose an exact algorithm based on dynamic programming. Moreover, a hybrid method combining dynamic programming and genetic algorithms (GAs) is also investigated, as well as an insertion heuristic method. Chung et al. (2007) [38] design several types of formulations for practical container road transportation problems. The basic problem is formulated as a multiple travelling salesman problem (MTSP), which is solved by an insertion heuristic. Namboothiri et al. (2008) [144] apply a root column generation heuristic to solve a local container drayage problem. Cheung et al. (2008) [34] analyse a cross-border drayage container transportation problem which manages individual and composites of multiple resources simultaneously. An attribute-decision model is designed and solutions are found using an adaptive labelling algorithm. Lai et al. (2013) [117] propose a new routing problem that can be viewed as a vehicle routing problem with clustered backhauls (VRPCB). Solutions are obtained with the famous Clarke-and-Wright algorithm and improved further by a neighbourhood based metaheuristic. Xue et al. (2014) [210] investigate a new drayage operation mode in which tractors and trailers can be separated using the processing time of customers. The problem is formulated as a vehicle routing and scheduling problem with temporal constraints and solved by a tabu search metaheuristic. More recently, Chen (2016) [30] proposed a multi-shift container transshipment formulation (see Section 3.3.9) for this research and solved it by a reactive shaking variable neighbourhood search (VNS) and a simulated annealing hyperheuristic method (SAHH).

Drayage problems with relocation of empty containers

Efforts to combine the planning of loaded and empty container transports are made by several authors. Coslovich et al. (2006) [42] analyse a fleet management problem for a container transportation company by decomposing the problem into three subproblems, which are then solved using a Lagrangian relaxation. Lleri et al. (2006) [99] present a column generation based approach for solving a daily drayage problem. Smilowitz et al. (2006) [176] model a drayage operation with empty repositioning choices as a multi-resource routing problem (MRRP) with flexible tasks. The solution approach is a column generation algorithm embedded in a branch-and-bound framework. Imai et al. (2007) [100] formulate a container transportation problem as vehicle routing problem with full container loads (VRPFC) and solve it with a subgradient heuristic based on Lagrangian relaxation. Caris et al. (2009) [28] extend this work and model the problem as a FT-PDPTW (see Section 3.3.6). A local search heuristic is proposed. The work is further extended by using a deterministic annealing algorithm suggested in Caris et al. (2010) [29]. Zhang et al. (2007) [219] improve the time window partitioning scheme used in Wang et al. (2002) [208] for container transportation in a local area with multiple depots and multiple terminals. The results indicate that good performance can be achieved compared with a reactive tabu search (RTS) method demonstrated in Zhang et al. (2009) [171]. After that, Zhang et al. (2011) [218] also investigate the single depot and terminal problem. Again, an RTS is proposed. Vidovic et al. (2011) [197] extend the problem analysed by Zhang et al. (2010) [219] and Imai et al. (2007) [100] to the multi-commodity case and formulate it as a multiple matching problem. Solutions are obtained via a heuristic approach based on calculating utilities of matching nodes. Nossack et al. (2013) [149] present a new formulation for the truck scheduling problem based on a FT-PDPTW and propose a two-stage heuristic solution approach. More recently, Braeker et al. (2013) [22] investigate a sequential and an integrated approach to plan loaded and empty container drayage operations. A single- and a two-phase deterministic annealing algorithm are presented. This solution approach is further adapted in Braekers et al. (2011) [18] to take a bi-objective optimisation function into account. The algorithms are further improved in Braekers et al. (2014) [21].

Drayage problems with dynamic inputs

Some researchers examine drayage operations problem in dynamical situations. Tjokroamidjojo et al. (2006) [188] investigate dynamic load assignment problems where loads arrive in a dynamic fashion and load information becomes available over time. The dynamic problem is repeatedly solved with the aid of a load assignment optimisation model. Wen et al. (2007) [204] study a local container vehicle routing problem with variable travelling time using a GA. Mes et al. (2007) [134] demonstrate real-time scheduling of full truckload transportation orders with time windows that arrive during schedule execution using an agent-based approach. The performance of the solution method is enhanced in Mes et al. (2010) [133].

Most of the aforementioned research work has been trying to formulate the drayage problem as some forms of classical vehicle routing problems in order to exploit the time constraint structures to prune the search space. However, this type of formulation does not work well for problems where time related constraints are not very tight and node-based solution representations generally lead to unnecessarily large search space, resulting to inefficient solution methods.

3.5.3 Multi-period vehicle routing problem

In real-life vehicle routing problems, the due dates of deliveries can fall in a wide planning horizon, which could result in very large problem size. In some problems, the deliveries can be partitioned into several classes depending on the urgency of deliveries (e.g. half-day, day, two-day parcel delivery). The planning horizon is thus structured and can be partitioned into multi-periods. The adaption of conventional commodity-flow VRPTW formulation is unable to efficiently solve multi-period VRP owing to the sparsity of the generated graph [124]. To circumvent this difficulty, the original large problem can be split into many subproblems according to the priority of deliveries and each subproblem is treated as an independent VRP. Related work can be found at [62], [36] and [185]. An alternative strategy is proposed by Letchford et al. (1998) [124] who investigated the rural postman problem with deadline classes (RPPDC, a variant of TSP where the underlying network may not form a connected graph) and the flow variables have been avoided and new valid inequalities are introduced in order to exploit graph sparsity. Frizzell et al. (1995) [71] studied the split delivery vehicle routing problem with time windows (SDVRPTW), where the length of time window may be too short to meet all customer demands. Therefore, some demands have to be delivered in the subsequent time slots. Three heuristics for the problem were developed and implemented. Local search was first introduced to SDVRPTW by employing the following operators: moving a customer to a new route and swapping customers between routes. Trudeau et al. (1992) [190] analysed a stochastic inventory routing problem for heating oil distribution for a long-time planning horizon covering up to 12 consecutive weeks. Christiansen et al. (2002) [35] studied a ship scheduling problem concerned with the pickup and delivery of bulk cargoes within given time windows. Wide time windows are regarded as multiple time windows. N-guyen et al. [147] considered the Time-dependent Multi-zone Multi-trip Vehicle Routing Problem with Time Windows (TMZT-VRPTW), which is an extension of the VRPTW involving both designing and assigning routes to vehicles within time synchronization restrictions.

In the area of waste collection and grocery distribution problems, each customer has to be served with a given periodicity, such as certain times a week. In practice, the vehicles have to return back to the depot when the working shift is over or the capacity reaches its limitation. This type of problem is usually formulated as PVRP model and the aim of the solution is twofold: to select a visiting schedule for each customer and to find vehicle routes in each working shift. A review of typical solution methods of PVRP can be found at [82] and [40] who also proposed a mathematical model and tabu search heuristic for PVRP. More recent studies of PVRP and its variants can be found at [98], [137], [194] and [129].

3.6 Summary

This chapter places the work in context and presents related literature about MFTLVRPs. Several works for vehicle routing problems, both on optimisation model studies and experimental studies, are reviewed. We introduce and classify the studies and models that share similarities with our research with a special focus on both bidirectional full truckload and multi-period vehicle routing problems.

For truck drayage problems, most of the models employ VRP-style (e.g. m-TSPTW) formulations that usually produce huge graphs if the number of tasks is large. Representing each task/customer/load as a node in a graph is a common practice for VRP-based formulations.

Most of the research work presented has been tried to formulate the drayage problem as some form of classical vehicle routing problem, trying to exploit the time constraint structures to prune the search space. However, this type of formulation does not work well for problems where time related constraints are not very tight and node-based solution representations generally lead to many-to-one mapping (will be discussed in next chapter) from solution to objective values. To address this issue, we present a novel set covering integer linear programming model for this research problem in the next chapter.

Chapter 4

A Set Covering Model and Lower Bound for the Multi-shift Full Truckload VRP

4.1 Introduction

This chapter studies a multi-shift full truckload vehicle routing problem (MFTLVRP) using data from a real-life problem faced by the Port of Ningbo. A set covering model is developed based on a novel route representation. A lower bound of the problem is also obtained by relaxing the time window constraints to the nearest shifts and transforming the problem into a service network design problem. Finally, features and merits of the set covering model are discussed.

4.2 Problem Description

Because of the distinct features illustrated in above, the problem cannot be solved by off-the-shelf approaches, such as, vehicle routing problems with

Table 4.1: The list of notations.

Input Parameters	
V	The set of nodes in the transportation network.
S	A list of time-continuous shifts in the planning horizon.
s	The s^{th} shift in S.
et^s	The beginning time for shift s.
lt^s	The end time for shift s.
d_{ij}	The distance between nodes $i \in V$ and $j \in V$.
μ_{ij}	The travel time between nodes $i \in V$ and $j \in V$.
t_j	The service time at node j.
t_j^l	The loading time at node j.
t_j^u	The unloading time at node j.
n	The number of trucks available for use.
R	Set of feasible truck routes within a shift. Each truck starts from depot v_0 and returns to depot before shift ends.
d_r	The distance of the route $r \in R$.
K	A set of commodities to be delivered. Each commodity is delivered by exactly one truck.
$Q(k)$	The quantity of the standard commodity k.
$o(k)$	The origin of the commodity $k \in K$.
$d(k)$	The destination of the commodity $k \in K$.
$\sigma(k)$	The available time of the commodity $k \in K$.
$\tau(k)$	The deadline or completion time by which the commodity $k \in K$ has to be serviced.
δ_{ri}^k	A binary constant indicating whether k can be serviced at the ith node in $r \in R$.
e_{ri}^s	Earliest time that truck route r finishes a service (either a drop-off or a pickup) at its ith node in shift s.
l_{ri}^s	The latest time that truck route r may depart from its ith node in shift s.
M	A sufficiently large positive number.
Decision Variables	
y_r^s	The number of times a given route $r \in R$ is used during shift s and $y_r^s \in \mathbf{N}^+$.
x_{ri}^{ks}	Commodity flow of the ith node of r in s for servicing commodity k.

pickups and deliveries. Similarly, the problem is different from the classic service network design problem which is primarily consolidation oriented. Although our problem shares similar constraints to the inland container transportation problem studied in Zhang et al. (2010) [219] (see Section 3.3.3), the nature of the constraints are very different. For example, the time window of a load in the problem considered in this research spans from a few hours to up to 3 days (or 6 shifts), compared to a load time window of between 1 to 4 hours in Zhang et al. (2010) [219]. Therefore, the time-window partition heuristics will not work for our problems due to the potentially huge number of sub-loads generated, causing prohibitive computational time.

When the time window of a load spans several working shifts, determination of the shift in which this load is serviced forms part of the decisions to optimise. In addition, the currently studied FTPDPTW such as the one illustrated in Section 3.3.6, that developed by Caris et al. (2009) [28], is only able to handle single-shift problem. However, the problem concerned in this research requires a multi-shift model that is much larger than the single-shift model adopted in Zhang et al. (2010) [219] and Caris et al. (2009) [28].

Although there are a number of research studies on full truckload/container transport problems with several models and algorithms being proposed, none of them can be effectively used to solve the problem described above. The reasons are summarised as following: 1) the planning horizon of our problem is much longer than those in the previous studies. This is because the time window of shipments in our problem spans from 1 hour to up to 3 days. The time-window partitioning approach will lead to a huge graph that is prohibitively large to solve. 2) The number of shipments is significantly larger than the instances tested in the previous studies while the number of physical nodes is relatively small (i.e. 9). The existing approach mentioned above could not exploit these structures explicitly. 3) Finally the operation in our problem is shift based (each shift is 12 hours). Although a shift can be interpreted as a time window for truck, its actual width is not necessarily bigger than the time windows of shipments, and inconveniently most VRP solution methods assumed a bigger truck time window than shipment time window. These issues lead us to consider a different approach which can fully exploit the structures of the problem, and hopefully can be more efficient than the existing approach.

Without losing the generality of the problem, we define the following shift-based full truckload shipment problem with operation dependent service time. A full list of the notations used in our model is given in Table 4.1. Denote $G = (V, A)$ a directed graph with a set of nodes V (representing origins and destinations of different commodities) and a set of arcs A between these nodes. Note that node 0 is the depot from which all vehicles depart at the beginning of the shift. Denote K be the set of all the commodities to be delivered. Each commodity $k \in K$ is defined by a tuple $(Q(k), o(k), d(k), \sigma(k), \tau(k))$, standing for the quantity, origin, destination, available time and deadline respectively. Denote S be a list of time-continuous shifts in the planning horizon and s be the s-th shift in S. All trucks have an identical capacity of 1 unit. Therefore, commodities are shipped directly to their destinations without transfers or consolidation.

Denote t_j be the service time at node j. Note that t_j is dependent on both the node a vehicle visits and the types of operation (either loading or unloading) done at this node. Denote t_j^l and t_j^u respectively be the loading and unloading time at node j, then we have

$$t_j = \begin{cases} t_j^l & \text{if loading operation only at node } j \\ t_j^u & \text{if unloading operation only at node } j, \\ t_j^l + t_j^u & \text{if both loading and unloading at node } j. \end{cases} \tag{4.1}$$

The problem is to find a set of vehicle routes with minimum total costs to deliver all the commodities within their time windows. Each vehicle route should depart from the depot at the start of a shift and return to the depot before the shift ends.

4.3 Feasible Route Generation

For a given directed graph $G = (V, A)$ where V is the set of nodes, representing different freight forward terminals and A is the set of arcs between nodes. Let node 0 be the depot. A feasible route is defined as a sequence of nodes that a truck can cover in a shift. Since no transshipment is permitted in the operation, for any feasible route, we ensure that each node will have at least one operation (i.e. either loading or unloading) with some nodes involving both operations simultaneously. Since time taken for loading/unloading operations is substantial and is comparable to the travel time between nodes, the service time at each node in a truck route will depend on actual commodity shipments along the route. The service time for nodes involving both of the operations will be much bigger than the service time if only one operation is scheduled at this node. This creates a very challenging issue for modelling since the service time is no longer a constant and depends on the actual solution. To circumvent this problem, for any node that involves both of the operations, we insert a copy of the node immediately after it in the route, setting the distance between them to 0 but an unloading service time for the first copy and a loading time for the second. In this way, all the routes now have exactly one operation per node except the depot. Because each unit of commodity shipment involves exactly two operations (i.e. loading at the source node and then unloading at the destination node) and a truck would never visit a node without a service, each of the feasible routes should contain an even number of nodes (including nodes copies). The following is an example of a feasible route.

```
  0 ---> 2 ---> 3 ---> 4 ---> 5 ---> 5 ---> 6 ---> 0

         |      |      |      |      |      |

depot  load   unload  load  unload  load  unload
```

In this particular route, a truck departs from the depot and picks up a commodity of unit quantity from node 2, and unloads the commodity at node 3. Then the truck picks up another commodity at node 4, drops it off at node 5. The final commodity delivered by this truck is from node 5 to node 6 before the truck returns to the depot. Therefore, in this route, in addition to truck movements to and from the depot, the truck movement from node 3 to node 4 is also empty. At node 5, the truck does both unloading and loading since it has two copies in the route. Excluding depot, odd numbered nodes are loading nodes and even numbered nodes have unloading operations only. Therefore, the service time of each node will be determined by the index of the node in the route. For an 0-indexed route, the service time of an odd-numbered node equals the loading time and even-numbered node has service time equalling its unloading time. Denote r^i be the i-th node in a feasible route r and t_{r^i} be the service time at node r^i:

$$t_{r^i} = \begin{cases} 0 & \text{if } r^i \text{ is depot,} \\ t^l_{r^i} & \text{if } r^i \text{ is an odd-numbered node in } r, \\ t^u_{r^i} & \text{if } r^i \text{ is an even-numbered node in } r. \end{cases}$$

where $t^l_{r^i}$ and $t^u_{r^i}$ are loading and unloading times at node r^i. With the route representation introduced above, we can now develop an integer model as follows. We will discuss later the algorithm to generate all feasible routes.

Denote R the set of all possible feasible routes within a shift and K the set of commodities and S the set of shifts within the planning horizon. Here each commodity $k \in K$ represents a number of containers with same properties defined by tuple $\{s(k), d(k), \sigma(k), \tau(k), Q(k)\}$, standing for its source, destination, time of arrival at port, deadline for shipment, and its quantity respectively. Note that in this application, we consolidate the quantity of commodity so that each truck carries one unit of a commodity exactly. For the real-life problem under consideration in this thesis, one unit of a commodity means two small containers (20 foot) or 1 large container (40 foot).

The solution can be encoded into two decision variables x_{ri}^{ks} and y_r^s. The first variable defines the commodity flow of the ith node of r in s for servicing commodity k while the latter defines the frequency of route r being used in a given shift s. Therefore, a given commodity k could potentially be serviced by several arcs of a route in different shifts, subject to constraints of time windows $(\sigma(k), \tau(k))$ and source-destination pairs being matched up between the arc and the commodity.

In order to speed up the processing time, one could pre-process all the possible arcs in each of the feasible routes for a given commodity. For each of the feasible route $r \in R$ and a given shift $s \in S$, a binary variable δ_{ri}^{ks} is introduced to indicate whether the ith node in route r of shift s can be used as the starting service node for commodity k. Therefore, $\delta_{ri}^{ks} = 1$ means that the following conditions should be satisfied, otherwise it is set to 0.

$$i \texttt{ mod } 2 = 1 \tag{4.2}$$

$$r^i = o(k) \tag{4.3}$$

$$r^{i+1} = d(k) \tag{4.4}$$

$$l_{r^i}^s \geq \sigma(k) + t_{r^i} \tag{4.5}$$

$$e_{r^{i+1}}^s \leq \tau(k) \tag{4.6}$$

Condition (4.2) indicates that the starting service node must be the node with loading action. Conditions (4.3) and (4.4) define source and destination of commodity for starting service node i. In constraints (4.5) and (4.6), $l_{r^i}^s$ is the latest departure time from the i-th node of route r in shift s to ensure all the subsequent services can be delivered on time. Similarly $e^s_{r^{i+1}}$ is the earliest time that a truck can possibly arrive at the $(i+1)$th node of r during shift s. $l^s_{r^i}$ and $e^s_{r^i}$ can be pre-calculated as follows:

$$e^s_{r^0} = et^s \tag{4.7}$$

$$e^s_{r^i} = e^s_{r^{i-1}} + \mu_{r^{i-1}r^i} + t_{r^{i-1}} \tag{4.8}$$

$$l_{r^{\bar{0}}} = lt^s \tag{4.9}$$

$$l^s_{r^i} = l^s_{r^{i+1}} - \mu_{r^i r^{i+1}} - t_{r^{i+1}} \tag{4.10}$$

where et^s and lt^s are the beginning and ending time for shift s respectively and $r^{\bar{0}}$ denotes the final node in route r (i.e. the depot). Eqs. (4.7) and (4.9) provide initial values for recursive equations (4.8) and (4.10).

4.4 Model Formulation

We now describe our proposed formulation for this problem. Our formulation is similar to the classic set-covering model with additional side constraints. The underlining idea is to find a subset of truck routes (from all possible feasible routes) that sufficiently covers all the transportation demands with a minimum total cost (i.e. distance). Because of the fact that all shifts are of identical periods and all the trucks must depart from the depot at the beginning of every shift and return to the depot before the shift ends, the feasible route set is the same for all shifts, assuming the travel times and service times are the same at different shifts.

Our problem can be formulated as finding a subset of all feasible route R for each of the shifts such that all the tasks are `covered` (or serviced) on time and the total routing cost is minimised.

Denote x_{ri}^{ks} and y_r^s the two decision variables. y_r^s is the frequency of route r used in the s^{th} shift in a solution. Variable x_{ri}^{ks} denotes, in a given solution, the commodity flow of the ith node of r in s for servicing commodity k. The problem can be formally defined as follows:

$$\min \sum_s \sum_r d_r y_r^s \tag{4.11}$$

subject to

$$\begin{aligned}
\sum_r y_r^s &\leq n \quad \forall s \in S && (4.12)\\
\sum_s \sum_r \sum_i x_{ri}^{ks} &= Q(k) \quad \forall k \in K && (4.13)\\
\sum_k x_{ri}^{ks} &\leq y_r^s \quad \forall i \in r, \forall r \in R, \forall s \in S && (4.14)\\
x_{ri}^{ks} &\in \mathbb{Z}^+ \quad \forall i \in r, \forall r \in R, \forall k \in K, \forall s \in S && (4.15)\\
y_r^s &\in \mathbb{Z}^+ \quad \forall r \in R, \forall s \in S && (4.16)
\end{aligned}$$

The objective is to minimise the total distance of all routes used in a solution. Constraint (4.12) ensures the availability of trucks the company actually possesses. Constraint (4.13) ensures all the tasks are serviced. Constraint (4.14) makes sure that the total flow of all commodities does not exceed the arc capacity.

Different from other VRP-style formulations (e.g. m-TSPTW or FT-PDPTW) that represent each load as a node in a network (same as Zhang et al. (2010) [219]), the above model treats commodities as `flows` to be covered by routes. This makes the proposed model advantageous compared to the other alternative methods. In real-life instances, it is common that large number of containers arrives with a same S/D pair and a same time window. For the above model, the complexity of solving a problem instance with $Q(k) = 10$ would be similar to the instance with $Q(k) = 100$. However, the latter instance would be multitude times harder to solve for Zhang et al. (2010)'s method [219]. The reason is that our model treat commodities as flows covered by route, therefore, a route is able to carry as much commodity flows as possible with the limitation of route capacity constraint. However, in Zhang's method, each container movement is considered as a node, thus, the graph size is significantly increased.

Nossack et al. (2013) [149] proposed a nonlinear integer programming model, in which time window constraints are handled explicitly. However, due to its nonlinear property, the formulation cannot be solved exactly. In our proposed set covering model, these time window constraints are implicitly handled offline during the feasible route generation stage. In this way, we can handle any forms (linear, nonlinear) of route related constraints, including nonlinear time window constraints and shift constraints. In fact, more and tighter constraints are advantageous to this formulation as it can reduce the size of the feasible route set.

More recently, Jianjun Chen et al. (2016) [30] proposed a multi-shift container transshipment formulation for this research and solved it by hyperheuristic method. The formulation is task based and inspired by the formulations given by Wang et al. (2002) [208] who proposed a m-TSPTW based model for full truckload transportation. As the number of tasks is equal to (exclude the depot node) the number of nodes in the graph, the model leads to a huge graph with the increasing number of tasks. For a modest-sized, 8-shift problem with 1000 tasks, the network would contain more than 8 million discrete decision variables with the number of constraints in the similar order [30]. Representing each task/customer/load as a node in graph is a common practice for VRP based formulations, indeed, in most routing and scheduling problems, the customers are located in different physical locations and can be visited only once. However, in terms of the drayage operation problem, the docks normally can be visited many times. There thus exist multiple optimal solutions as many tasks share an identical source, destination and time window but they are defined as different nodes in the graph. Therefore, different from other VRP-style formulations that represent each load as a node in a network, the set covering model presented in this study treats physical docks as **nodes** and commodities as **flows** to be covered by routes. In other words, the node representation applied to this problem enables an $arc(i, j)$ in a route to represent all possible collection of commodities that: 1. Sharing the same source of node i and destination of node j. 2. Satisfying time window constraint of delivery.

One of the most helpful benefits of this encoding is the transformation of a previous m-TSP based non-linear model (e.g. the model proposed by Chen et al. (2016) [30]) into a linear integer model, so it can be solved using various integer programming techniques. This was done through hiding nonlinear time related constraints into the generation of the feasible truck

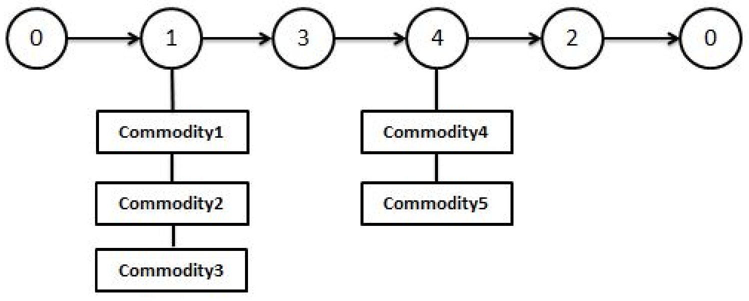

Figure 4.1: Example of a routing sharing among 5 commodities.

route set. Figure 4.1 presents a simple example of a feasible truck route where 0 denotes the depot. For a 0-indexed route node list, odd numbered nodes are commodity loading nodes (i.e. nodes 1 and 3 in Figure 4.1) while even numbered nodes are unloading nodes. If a node on a route is involved with both loading and unloading, a copy of it is created so that the above rules are maintained.

A second benefit of this solution representation is its capability to handle nonlinear cost functions. For example, the costs of routes could be a nonlinear, complex function of the distance. It also permits to include various other constraints related to drivers (e.g. maximum driving distance, time or preferred terminals).

4.4.1 One-to-one mapping vs. many-to-one mapping

Figure 4.1 illustrates a simple example of this flow assignment scheme (defined by x_{ri}^{ks}) on a feasible route being used six times (i.e. $y_r^s = 6$).

In this example, commodities 1, 2 and 3 share 6 units of truck capacity on arc (1, 3) while commodities 4 and 5 share 6 units of capacity on arc (4, 2). For arc (1,3), there are total of $4+3+2+1=10$ (i.e. 4 commodity1, 1 commodity2, 1 commodity3 or 3 commodity1, 2 commodity2, 1 commodity3...) possible flow allocations between the three commodities, while for arc (4,2) the total number of possible flow allocations between commodities 4 and 5 is 5. Therefore, in total, there are $10 \times 5 = 50$ different flow allocations and all of them result in the same objective value in terms of the total distance.

The advantages of this solution encoding is clear in this example, because all these solutions are encoded as one representation in the proposed encoding scheme. Hence the mapping from the search space to the objective function is one-to-one and the size of the search space is reduced significantly. In the case of node-based traditional VRP types of formulations (e.g. those adopted in Zhang et al. (2010) [219], Chen et al. (2016) [30]), each of these 50 solutions will be represented uniquely, despite all resulting in the same objective value (i.e. many-to-one mapping). This leads to a significantly larger search space with a plateau.

4.5 A Lower Bound

The VRP belongs to the class of NP-hard problems and the exact model presented above is only able to solve small problems (e.g. number of commodity unit is less than 400) optimally. The lower bound model is thus developed to provide a guide of optimal solutions and analyse the performance of our approaches. Therefore, the function of the lower bound is twofold: on one hand, it can be used to determine the nature of instance data. For example, the unbalanced demand distribution in space and time nodes would result in a large number of dead-heading of empty trucks. On the other hand, it evaluates the ability of finding near optimal solutions by our approaches.

To do this we solve a simplified problem in which the time window requirements of each shipment (i.e. arrival time and delivery deadline) are relaxed to the corresponding shift in which the time window lies. For ease of modelling, we also neglect the empty truck movements from/to the depot in computing the lower bound. We define $\upsilon(k)$ and $\omega(k)$ respectively be the shift that commodity k becomes available and the shift that the delivery deadline of k lies in. Denote u_{ij}^{ks} be the flow of commodity k on arc (i, j) during shift s and v_{ij}^{s} be the number of vehicles covering arc (i, j) during shift s. In addition, the constraint of all trucks returning to the depot is discarded to exclude the factor of inappropriate depot location. The relaxed problem can be formulated as the following service network design problem.

$$\min \sum_{s} \sum_{(i,j)} d_{ij} v_{ij}^{s} \tag{4.17}$$

subject to

$$\sum_{s=\upsilon(k)}^{\omega(k)} \sum_{j} u_{ij}^{ks} - \sum_{s=\upsilon(k)}^{\omega(k)} \sum_{j} u_{ji}^{ks} = b_i^k \quad \forall k, \forall i \tag{4.18}$$

$$\sum_{j} v_{ij}^{s} - \sum_{j} v_{ji}^{s} = 0 \quad \forall s, \forall i \neq 0 \tag{4.19}$$

$$\sum_{k} u_{ij}^{ks} \leq v_{ij}^{s} \quad \forall (i, j), \forall s \tag{4.20}$$

where constraints (4.18) are the flow conservation constraints, (4.19) are the truck balance constraints, and (4.20) are the capacity constraints.

4.6 Summary

This chapter presents a set covering integer linear programming model for a real-life MFTLVRP. The model would significantly reduce the problem size compared with the VRP-style formulations that are widely adopted for representing drayage operations problems.

The problem belongs to the class of NP-hard problems and the set covering model is only able to solve small problems optimally. The lower bound model is created in order to provide approximately optimal solutions in a short time and evaluate the performance of solution procedures. In order to evaluate the feasibility and performance of our model, we applied it to a number of real-life and artificial instances. The result and solution procedure will be presented in Chapter 6.

4.6.1 The non-deterministic drayage operation problem

In most existing models for VRP, the travel time of each trip is assumed to be fixed, preventing the resulting schedule from operating as planned in practice owing to the variability of traffic conditions. To tackle this situation, many studies have been conducted: 1) to try to set more accurate travel times; 2) to reschedule during real-time operations when the situation occurs; 3) to reinforce the robustness of a schedule (e.g. leave sufficient buffer time).

As mentioned, the travel times between docks are necessary parameters for the model formulated in Section 4.3. Indeed, travel time is one of the most essential parameters for the compilation of a vehicle routing problem. Its accuracy may also affect the on-time probability of a vehicle schedule [174]. There are many methods that can be adopted to set travel times. The most commonly used method may be based on the experience and common sense of human schedulers [73]. Thanks to the advancements in automatic data collection system technologies (e.g. Automated Vehicle Location systems (AVL), Global Positioning Systems (GPS)), which are increasingly being installed in transport systems, large amounts of collected data are available for us to determine more accurate travel times. Examples include mean time, 85th percentile travel time [72], and mean value plus the standard deviation [143].

By analysing real-life GPS data obtained from a container truck fleet at the Port of Ningbo, we observed an increase in travel time patterns during peak times. This motivated us to investigate further to estimate travel times more accurately and efficiently. We ultimately developed a short-haul travel time prediction model and algorithm using real-life GPS data. Instead of using fixed travel times when generating feasible routes for the set covering model in practice, this prediction model is suggested to estimate travel times due to the variability of traffic and driving conditions. Hence, even though the set covering model studied in this chapter is deterministic, its travel time parameters can be non-deterministic. The idea of this approach is an alternative method to other non-deterministic approaches (e.g. stochastic programming approach [174], robust scheduling [198]). As mentioned, the feasible route set of the model can be handled offline, which eases its implementation in practice.

The non-deterministic drayage operation problem can be solved by 3 stages: 1) travel time prediction; 2) generation of feasible routes; 3) solve the set covering model. In real-life, stage 1 and stage 2 need not to be executed whenever stage 3 is conducted as the travel time parameters obtained from stage 1 will not significantly change within a short period of time (e.g. 1-2 weeks). Stage 1 and 2 can be executed only when there is a dramatic change (e.g. port location, seasonal change) in travel times parameters that may affect the generation of feasible routes. Reoptimisation of stage 3 may take place when there are lots of changes that occur to the tasks. For instance, more than 10% tasks that have been planned cannot be picked up or delivered as planned. In that case, we remove the tasks that have been successfully delivered and add new tasks when necessary, then solve stage 3 one more time. If only a few tasks (e.g. less than 10% of total tasks) cannot be picked up or delivered as planned, we suggest to heuristically insert (similar to the method applied in Section 6.4) these tasks to other unfinished routes in order to reduce the burden to the server caused by the reoptimisation.

In the next chapter, a short-haul travel time prediction model and algorithm using real-life GPS data (the stage 1 that has been discussed in the previous paragraph) are presented.

Chapter 5

Truck Travel Time Prediction in Port Drayage Network

5.1 Introduction

Unreliable travel time is regarded as among the most problematic issues in freight operations [83]. Truck drivers experience unnecessary wait times if they arrive early and truck queuing causes high diesel engine emissions. Delays can incur more problems [157]. The model presented in Chapter 4 relies on travel times between docks to generate a feasible route set. Therefore, accurate travel time prediction is necessary and important for this model.

In this chapter, a short-haul travel time prediction model and algorithm using real-life GPS data is presented. Fleet GPS data used in this work were obtained from Ningbo Port Co., Ltd. The processes of data preparation, variable/model selection and data generation are also illustrated.

5.2 Related Literature

Travel time information plays a role of great importance in the field of transportation and logistics planning, as accurate travel time prediction is not only crucial for developing intelligent transportation systems (ITS) that help traffic managers to make decisions and it is also useful for logistics company managing freight transportation [202].

Much effort has been contributed to short-term traffic forecasting which has been a crucial part of traffic management since the 1980s. The ability of rapidly processing traffic data has brought significant development of ITS (Intelligent Transportation System) technologies. Short term traffic forecasting is concerned with predictions made from a few seconds to possibly a few hours into the future based on the current and past traffic information [199]. The advances in new technologies in traffic data collection such as automatic vehicle identification system, global positioning system and smart phones have made data for short-term traffic time prediction more rapidly available. The most commonly used variables for traffic predicting in literature are traffic flow, occupancy, speed, and travel time [107]. In most literature the traffic flow parameters dominate the field of traffic forecasting, but they also exhibit conflicting results when deciding which parameter is more suitable for traffic predicting. Since the past few decades, travel time prediction has drawn increasingly more interest, as it is not only an important measurement of traffic performance but also a straightforward variable to inform drivers of the current and future traffic conditions.

Much research effort has been invested in developing accurate and robust traffic prediction models and the problems have been treated from various angles including time-series, pattern recognition, clustering, and regression. Generally, those models can be classified into parametric, non-parametric and traffic simulation methods. Parametric methods for traffic prediction primarily rely on statistical techniques such as autoregressive moving average models, linear and nonlinear regressions [173]. These techniques try to detect a function between the past information and the predicted state. However, these methods are typically sensitive to errors and data quality. Non-parametric approaches adopt computational intelligent methods like fuzzy systems, machine learning, and evolutionary computation. Such techniques can generally handle imprecise data, dealing with the nondeterministic, complex and nonlinear systems but they tend to have efficiency problems that means there exist a trade-off between accuracy and efficiency. Micro-simulation imitates real world process of traffic conditions, however it is easy to draw wrong conclusions if one does not fully understand the processes of traffic operation in detail [24].

Parametric methods are applied extensively in estimation, prediction and modelling fields. In the work conducted by Rice et al. (2004) [166], a linear regression model was developed to predict travel times on freeways using loop-detector data on flow and occupancy at selected locations in California. Okutani et al. (1984) [151] employed Kalman filtering theory to predict short-term traffic volume based on data collected from a street network in Nagoya city. Average prediction error of their method is found to be less than 9% and the maximum error is less than 30%. Jula et al.

(2008) [102] also applied Kalman filtering to predict the travel time between Los Angeles and West Covina, in California. Ahmed et al. (1979) [4] first introduced ARIMA (autoregressive integrated moving average) in traffic forecasting. After that, many extensions of ARIMA have been created for traffic predicting. Yu et al. (2004) [214] introduced a switching ARIMA model for traffic flow forecasting in Yuetanbei street of Beijing. The author applied ascending, decreasing and bottom patterns switching ARIMA models to tackle their problems and it was demonstrated that switching ARIMA performed better than conventional switching models.

With the development of computational intelligence in transportation research, a large number of computational intelligent applications have appeared. Karlaftis et al. (2011) [105] discussed differences and similarities between the statistical method and the neural network method. Lint et al. (2008) [191] built an online machine learning approach based on extended Kalman filter model for traffic time prediction on a 7-km 3-lane southbound freeway in Netherlands by using dual inductive loops data. Coufal et al. (2003) [43] introduced two new prediction models between Lahti to Heinola, Finland (28km) using traffic data obtained from traffic camera and inductive loop detectors. The first one is a result of GUHA style data mining analysis, which is a method of computerised generation of hypotheses based on given data, and a so called Total Fuzzy Similarity method. The second one is a hierarchical model based on neuro-fuzzy modelling. Sun et al. (2014) [183] combined Multidimensional Scaling with nonlinear regression Support Vector Machines (SVM) to forecast traffic flow.

Simulation is a valuable support tooling for evaluating traffic conditions. Microscopic and macroscopic are two concepts that are widely adopted by traffic engineers in traffic forecasting. Mbiydzenyuy et al. (2013) [132] proposed a travel time prediction method which makes use of a micro-level simulation, and a set of input GPS data in order to simulate the movement of vehicle. By simulating several journeys along the same route and according to the principles of Monte Carlo sampling, the method generates a distribution over the predicted travel time. Hu et al. (2009) [97] applied two algorithms, the flow based and the vehicle based models, for travel time prediction based on the concept of simulation assignment models.

Most of the models and techniques discussed above are mainly applied for passenger cars or similar types of vehicles which are differ from container trucks as: firstly, the problem concerned in this work is regarding with short-haul transportation and the travel distance for a truck may not comparable with the road distance for a passenger car studied before. Secondly, road and traffic condition of ports is different from urban main road or freeway studied in previous research. In this research, we specifically study computational intelligent models and techniques for the prediction of container truck travel time. The outcomes of this research can be used to produce high quality robust container truck transportation schedules.

Most literature has been focused on developing forecasting techniques for urban main road or freeway, however, literature on prediction for real-world container truck (drayage operations) travel time has been somewhat limited. In this chapter, a real world short-haul travel time prediction model is presented. Fleet GPS data used in this work were obtained from Ningbo Port Co., Ltd. The processes of data preparation, variable/model selection and data generation are illustrated. The impact of festival, time of day and rainfall variables for the predictive model is evaluated. The forecast performances by an autoregressive integrated moving average (ARIMA) method, a neural network (NN) approach and a support vector machine (SVM) model were also compared. The results indicate that for the traffic data under drayage operation scenario, ARIMA model consistently performs best.

5.3 Data Preparation

The fleet GPS data contain information of trucks' license plate, time information, location, speed and direction. The ports' GPS location data record maximum and minimum of longitude and latitude (shown as the shaded squares in Figure 5.1) of each port. The travel time is calculated by the difference between the last timestamp when a truck leaves the source port and the first timestamp when the truck enters into the destination port. Figure 5.1 shows truck movement trajectories from port BLCT2 to BLCTMS. Timestamps at the points in 'truck figure' in Figure 5.1 were recorded.

Two problems were encountered when processing the data: Firstly, in some rare occasions, the positions reported by trucks are not accurate and their GPS location contains errors more than 30 to 50 meters. It is not a significant problem if the errors occur during transportation but it would lead to wrong calculation of travel time if the errors occur at ports' area.

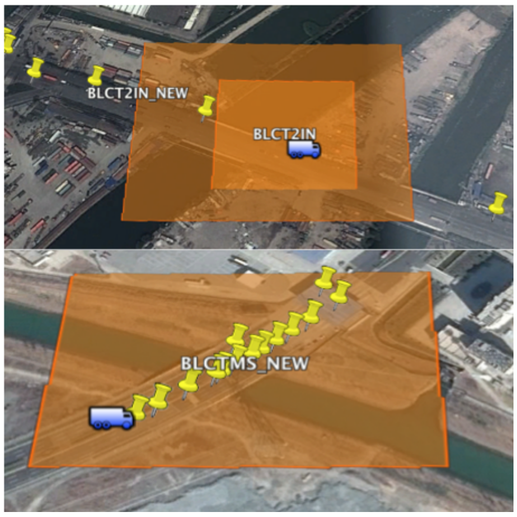

Figure 5.1: Truck transport from port BLCT2 to BLCTMS.

The second problem is regarding with ports' location area which is a rectangle with 200-250 meters length and 150-200 meters width. This area is comparatively small when a truck travel in high speed passing through the area, as the GPS device installed on truck reports in every 30 seconds so that it may move out the area without leaving any footprint there when a truck goes through this area at relatively high speed. The solutions we adopted for dealing with those problems are to enlarge ports' area and replace original ports' area to its' most frequently visited area such as main road or conjunctions near by them.

After obtaining travel time between different ports based on the method described above, large and unacceptable standard deviations for the travel times between ports were observed. Therefore, the reasons are analysed by selecting and plotting representative trucks' trajectories, which indicate travel time abnormally long or short. By analysing trucks' trajectories, the following possible reasons were found: Firstly, official or personal business disturbs drivers and prevents them going directly from one port directly to another port. Secondly, drivers occasionally run into traffic problems such as traffic congestion. Thirdly, drivers choose different routes which could lead to very different travel times.

To filter out noise, drivers' experience, civilian navigation system and experience gained from analysing representative trucks' trajectories were adopted in this study. The estimation by the drivers experience provided a rough time to complete a task. This value is denoted as D_t. An upper bound $E_t = 1.3{*}D_t$ is set based on our experience and discussions with the staff and drivers from Ningbo Port Co., Ltd. Travel time smaller than E_t is considered as guidance value. The civilian navigation system adopted in this work is AutoNavi, which provides digital map content and navigation solutions in China. AutoNavi gives us another reference of cost in travel time between ports. By comparing the reference values with the travel time obtained by our method, a 10-20% smaller reference value is always observed. This may be because civilian navigation system is designed for family cars that usually travel faster than container truck. For example, Figure 5.2 and 5.3 show the distribution of travel time from BLCT to BLCT2 and BLCT3 respectively, X axis in the figure indicates travel time in minutes while Y axis indicates its probability density function. The travel time suggested by civilian navigation system between BLCT to BLCT2 and BLCT to BLCT3 are 6 and 39 minutes respectively but the travel time obtained by us shown in figures are around 8 and 43 minutes respectively.

The container transshipment between the nine nodes are unbalanced. For example, thousands of transportation records from port BLCT to BLCT2 were obtained but from ZHCT to BLCT2 there are only few records (e.g. 5-20). In order to maintain the quality of data, transportation between ports with less than 200 records is not considered in this study. Consequently, the transportation between ports in pairs considered by us for prediction are given in Table 5.1.

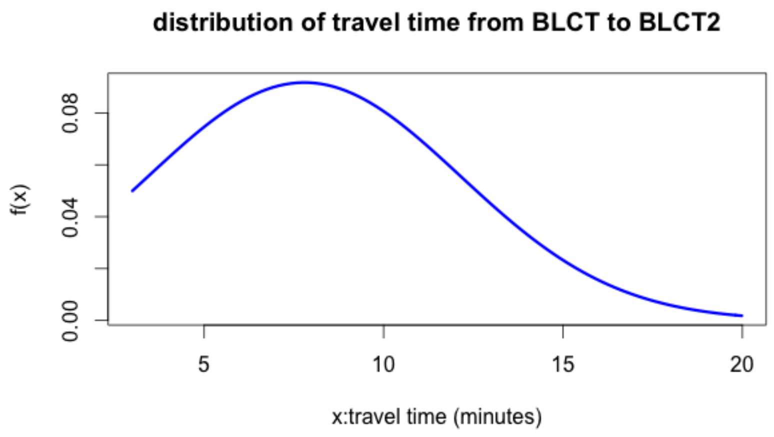

Figure 5.2: Distribution of travel time from BLCT to BLCT2.

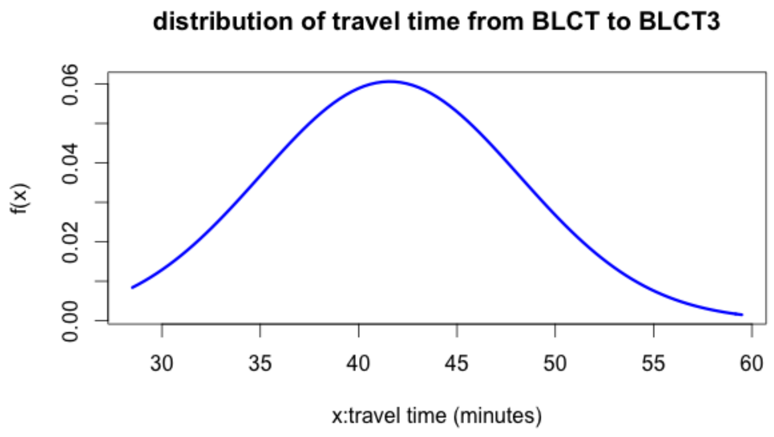

Figure 5.3: Distribution of travel time from BLCT to BLCT3.

Table 5.1: Names of journey for prediction.

Instance	Source	Destination
1	BLCT	BLCT2
2	BLCT	BLCT3
3	BLCT	BLCTMS
4	BLCT	BLCTZS
5	BLCT2	BLCT3
6	BLCT2	BLCTMS
7	BLCT2	BLCTZS
8	BLCT3	BLCTMS
9	BLCT3	BLCTZS
10	BLCTZS	BLCTMS

5.4 Data Analysis

According to previous travel time prediction studies [213] and [125], precipitation, time of day and day of week are important variables for formulating predictive model. Besides, we also would like to investigate how festivals affect travel times between port transportation. Because the size of data (3 weeks) is limited, the day of week variable is not considered in this work. Therefore, in this section, the impact of festival, time of day and rainfall variable are investigated and the qualified variables were also chosen for this travel time prediction work.

5.4.1 Impact of festivals

Data in this work was obtained from 25th April to 15th May, during this period, a traditional holiday on May 1st- the International Labour Day was considered to be a factor for affecting the travel time between ports. Although most employees were having holiday from 1st to 3rd of May, workers in ports are supposed to stay at work due to the particularity of their work, thus the holiday will not affect the operation of the truck fleet. In order to evaluate impact of this holiday on travel time, the travel time in holiday against its normal time counterpart is presented.

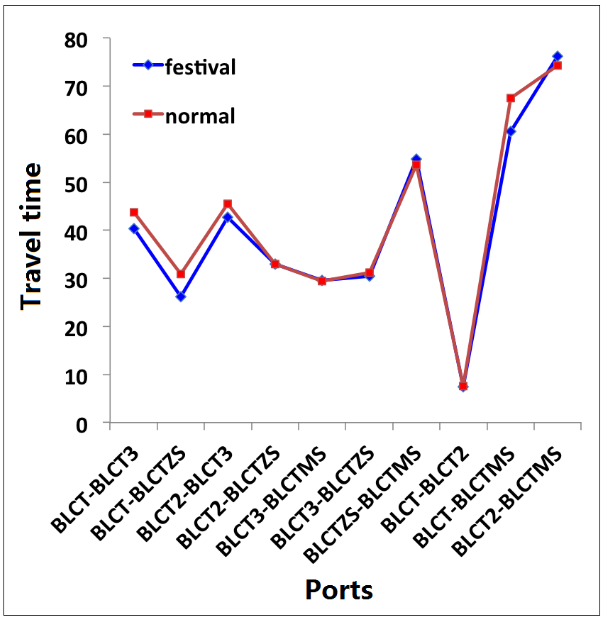

Figure 5.4: Festival vs. normal-mean travel time.

Figure 5.4 shows the comparison of travel time in festival and normal time of each port. *X* axis represents names of ports in pairs and *Y* axis shows mean value of corresponding travel time in minutes. In this figure, part of records show more travel time was taken during the festival (e.g. BLCT to BLCTMS and BLCTZS to BLCTMS) while others took less travel time during the festival (e.g. BLCT to BLCT3, BLCT to BLCTZS and BLCT2 to BLCT3). Therefore, it is not conclusive whether the festival of the International Labour Day has significant effects on travel time. A two-tailed t-test was conducted between the two groups of data. P-value of this test is 0.861 (>0.05), for this reason, the impact of festival is not considered in the final prediction model. Further studies will be conducted once more data is obtained.

5.4.2 Impact of time of day

By considering commuters' travelling behaviour in Ningbo, a time period of 6am to 8am, 11am to 1pm and 4pm to 6pm were chosen as traffic peak time. Figure 5.5 shows mean value of travel time between ports in peak and off-peak period. Port names are plotted in horizontal axis as pairs while mean travel time in minutes are presented in vertical axis. It can be seen from Figure 5.5 that the mean travel time in peak time of all selected ports outweighs that of its off-peak counterpart. A two-tailed t-test was conducted between the two groups of data. P-value of this test is 0.004 (<0.05), therefore a conclusion can be drawn that the peak time would slow down travel time. After this, the impact of different times of each day on travel time is also investigated and it shows the travel times fluctuate over 24 hours each day. For example, Figure 5.6 and Figure 5.7 show how travel time between BLCT to BLCT3 and BLCT to BLCTZS varies in different times of day. For each figure, the X axis indicates time of day in 24 hours and Y axis shows mean value of corresponding travel time in minutes. It can be seen clearly that increasing travel time patterns appear in peak time from these two figures. For the fact that different times of day do have effects on travel time, time of day is considered as a variable for the prediction model.

5.4.3 Impact of rainfall

In this study, rainfall data was obtained from ACL (air resource laboratory) which is a research laboratory of NOAA's Office of Oceanic and Atmospheric

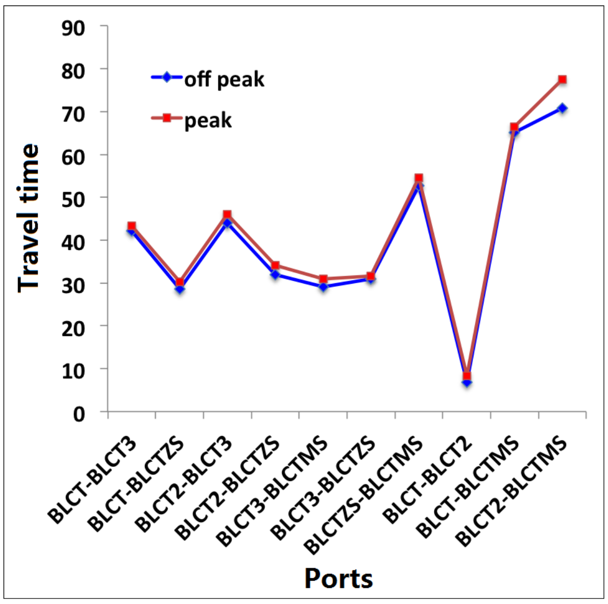

Figure 5.5: Peak vs. off peak-mean travel time.

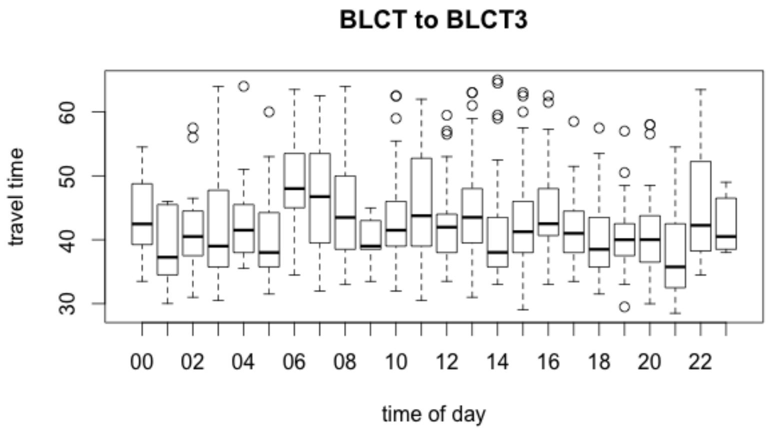

Figure 5.6: Travel time from BLCT to BLCT3.

Research with interval of three hours. The rainfall data and travel time data were firstly combined into one dataset and we then compared travel time between ports with rainfall against no rainfall situations. Figure 5.8 shows the comparison of travel time with and without rainfall between selected ports. X axis represents port names as pairs and Y axis shows mean corresponding travel time in minutes. In this figure, travel times between some ports are higher in the rainfall (e.g. BLCT to BLCT3 and BLCTZS

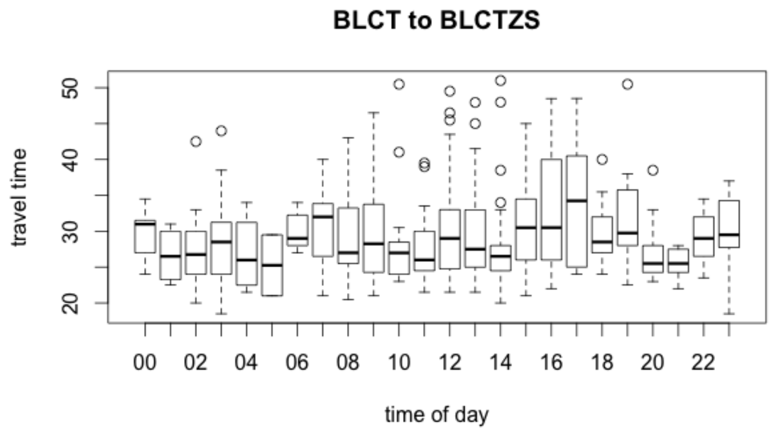

Figure 5.7: Travel time from BLCT to BLCTZS.

to BLCTMS) while travel times of others are lower in rainfall (e.g. BLCT2 to BLCT3, BLCT to BLCTZS). A two-tailed t-test was conducted between the two groups of data. P-value of this test is 0.975 (>0.05), thus, rainfall is not considered in the prediction model since it doesn't have significant effects on travel time.

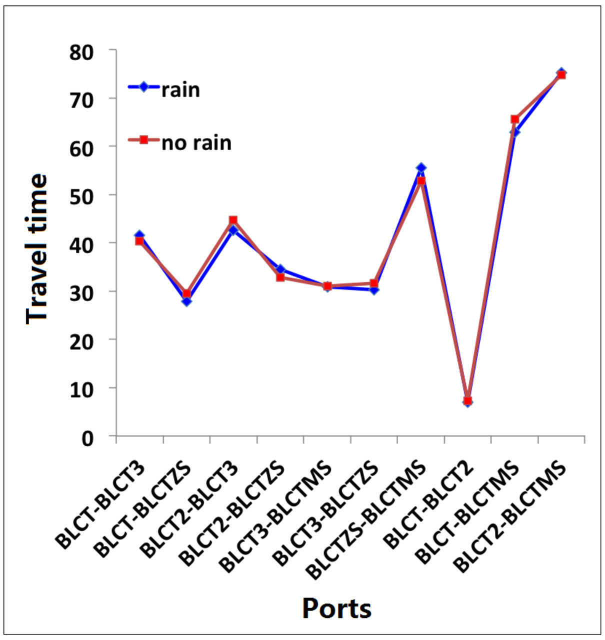

Figure 5.8: Rainfall vs. no rainfall-mean travel time.

After the evaluation of the impact of festival, time of day, and rainfall variables on travel time, we draw an initial conclusion (which is limited by the size of the data available) that festival and rainfall do not have significant impact on the investigated travel time and they are not considered in our prediction model. Time of day variable is adopted as prediction variable due to its impact on travel time. This conclusion differs from previous research [125] [65] and [215], and the reasons may lie in: Firstly, the problem concerned in this work is regarding short-haul transportation, and the travel distance for a truck may not be comparable with the road distance studied before. Secondly, road and traffic condition of ports are different from urban main road or freeway studied in previous research.

5.5 Travel Time Forecasting

In this section, a time series data transformation approach is introduced and a method for travel time prediction is developed. In particular, we focus on the (autoregressive integrated moving-average) ARIMA. We take travel time estimation from BLCT to BLCT2 as an example for illustrating the solution procedures.

5.5.1 Data transformation

As mentioned in Section 5.4.2 the different times of day do have effects on travel time. The data we obtained are recorded over time and can be considered time series data. Also, the demand of container transportation between different docks vary in time. For example, the records of travel time from port BLCT to BLCT2 have 50 observations from 12pm to 1pm, 40 observations from 1am to 2am, and 220 observations from 3pm to 4pm. This means the data are not continuous with equal periods, therefore, further transformation of the raw data has to be conducted. The original travel time data are transferred into time series data by copying the neighbouring value and the previous period's value of the missing data. Missing values' neighbours or mean value A_i are used to fill the data when there is no observation in time i and A_i is calculated in Eq. (5.1). Please note that A_{i-1} indicates the mean value of travel time one hour ahead of time i and A_{i+1} indicates the mean value of travel time one hour after time i. A_i is dependent on its neighbours's value if there is no observation at time i, or its value in previous periods if there exist observations at time i. The notations are listed in Table 5.2.

Table 5.2: Notations used in the method description.

d	Total number of days, $d \in \mathbf{N}^+$
i	Time of Day, $i \in \{1, 2, 3..., 24\}$
j	The index of day, $j \leq d$
T_{ij}	Travel time in time i of day j
A_i	Average travel time of time i.

$$A_i = \begin{cases} \frac{\sum_{j=1}^{d} T_{ij}}{d}, & \sum_{j=1}^{d} T_{ij} > 0 \\ \frac{A_{i-1}+A_{i+1}}{2}, & \sum_{j=1}^{d} T_{ij} = 0 \end{cases} \tag{5.1}$$

Transformation of data would inevitably bring biases. The proposed data transformation method is able to keep the original shape of data and avoid introducing significant biases. Take the travel time from BLCT to BLCT2 as an example, the statistics of original and transformed data is given in Table

5.3 which indicate that the minimum and maximum value stay unchanged and no significant difference between 1st quartile, median, mean of original and transformed data. The transformed data are more smooth than original data as the standard deviation reduced from 1.25 to 0.83.

Table 5.3: Original VS. transformed data (BLCT to BLCT2).

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	S.d.
Original	3.00	4.50	5.5	5.54	6.00	19.02	1.25
Transformed	3.00	5.05	5.48	5.54	5.94	19.02	0.83

5.5.2 Travel time prediction by ARIMA

Time series models come in three kinds: moving average (MA) models, autoregressive (AR) models and autoregressive moving average (ARMA) models. The method studied in this thesis is built upon (autoregressive integrated moving-average) ARIMA due to the non-stationary (e.g. BLCT2 to BLCTZS time series, see the next paragraph) and stationary property (e.g. BLCT to BLCT2 time series) of the data obtained. As its name interpreted, ARIMA (p,d,q) model is a combination of three terms, an autoregressive term (p), a moving-average term (d) and an integrating term (q) [4]. ARIMA(p,d,q) can be transformed into an ARMA(p,q) by differencing its (d). ARIMA was originally developed for applications in the fields of business, industry, and economics and have been later applied to time series applications. There have been numerous attempts to use ARIMA model for short-term traffic flow forecasting [214].

The time series data has to be stationary before it can be fitted into ARIMA model directly. Otherwise, we have to adjust it by differencing its

(d) which is able to convert non-stationary time series to stationary time series as long as (d) is rightly identified. Time series plot is the most basic approach to help us determine whether the data are stationary. Another strategy is by calculating the autocorrelation function (ACF) as well as partial autocorrelation function (PACF) and checking if the lags die out quickly. ACF and PACF provide a useful measure of the dependence degree of a time series at different times. The ACF and PACF of BLCT to BLCT2 are presented in Figure 5.9 which shows the correlation at lag 0 is close to 1 and the lags die out quickly, so we can conclude this time series is stationary. To confirm with this result, the augmented Dickey-Fuller (ADF) test is also conducted: the resulted P-value equals to 0.01, which suggests that the data is stationary. In this study, the time series between some ports are non-stationary, in which case, stationary series normally can be obtained by differencing its (d) 1-2 times.

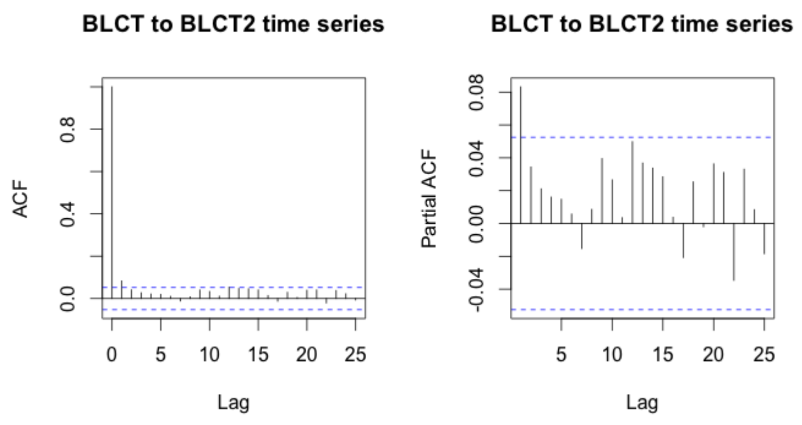

Figure 5.9: ACF and PACF of BLCT to BLCT2 time series.

5.5.3 Modeling and model validation

Generally, there are two groups of algorithms that can be used for estimating model parameters: the preliminary estimation (e.g. Yule-Walker Estimation and Burg's algorithms) and maximum likelihood estimation. Maximum likelihood estimation libraries provided by R toolkit was used to obtain estimates of the model. After desired time series data are generated by the method described in Section 5.5.2, the time series data are split into a training set which accounts for 2/3 of total records and test set which accounts for the rest 1/3 of total records. Again, we take travel time estimation from BLCT to BLCT2 for illustration and found that ARIMA(2,0,2) model with AR coefficients of (0.5306,-0.9851) and MA coefficients of (-0.5314, 0.9994) seems to generate the best results for road section from BLCT to BLCT2. 21 days of predicted traffic time vs. real traffic time from port BLCT to BLCT2 is shown in Figure 5.11. This figure is an example of how predicted values fit the real value using ARIMA (2,0,2). The residuals analysis was conducted to verify the ARIMA(2,0,2) model for road section from BLCT to BLCT2. Figure 5.10 shows the residuals are random and close to zero and roughly lie within the bounds from -3 to 3, indicating that overall the residual time series approximate a zero mean white noise behaviour. Thus we believe the proposed data transformation and ARIMA model is capable of forecasting travel time scenarios studied in this thesis. Model accuracy analysis was conducted and measured by MARE (mean absolute relative error) and all the results measured by MARE are listed in Table 5.6.

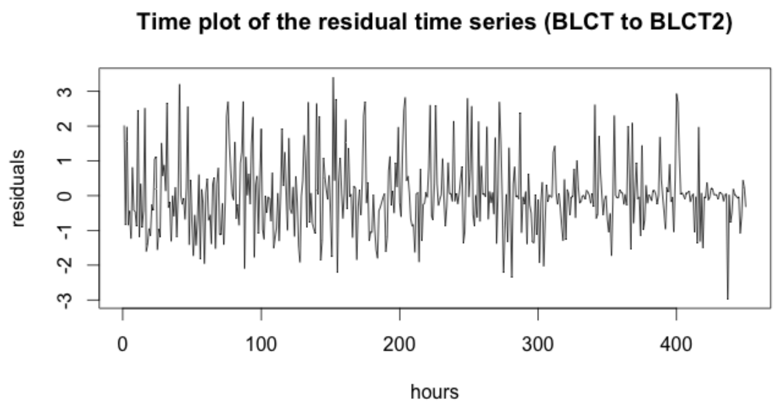

Figure 5.10: BLCT to BLCT2 model residuals.

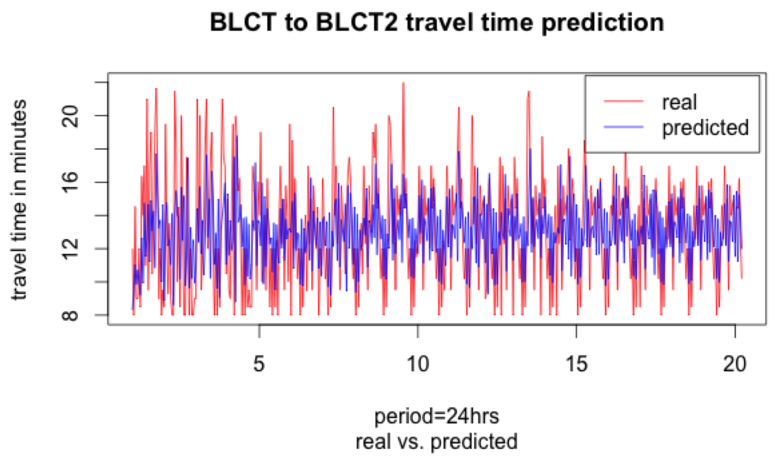

Figure 5.11: Predicted vs. real travel time from BLCT to BLCT2.

5.6 Comparisons with Other Forecasting Methods

There are different techniques explored for predicting traffic variables such as time series analysis, artificial neural networks (ANN) and more recently support vector machines (SVM). However, the studies reported the performance of these predicting techniques are varying due to using different traffic data. Indeed, the traffic conditions of different areas vary from one to another. Hence, there is a need to explore the use of ANN and SVM for predicting traffic time under drayage operation scenario studied in this thesis.

5.6.1 Back-propagation NN

In the field of data analysis in transportation research there are two 'schools of thoughts': the first employ statistics as the tool of choice while the second insists on neural network methods. The two have kept each other at arm's length [105]. As ARIMA model can be treated as a statistical method, we are also interested in the comparison of its performance on travel time predicting with a neural network (NN) method. Back-propagation NN is one of the most widely used model of neural network, it was selected for this application based on its well developed theory, and its ability to model relationships between continuously nonlinear valued variables [177]. Therefore, we compared NN with ARIMA by implementing a back-propagation NN. The time series data were scaled (unscaled after the predicted values are obtained) and split into training set which account for 2/3 of total records and test set which account for the rest 1/3 of total records. The model contained less than 5 units (optimised with tuning) on one hidden layer. The maximum number of iterations is set as 100. Root mean square error (RMSE) was used to select the optimal model using the smallest value. The tuning parameters of optimised result are given in Table 5.4.

Table 5.4: The tunning parameters of optimized result (NN)

Ports	Size	Decay
BLCT to BLCT2	1	0
BLCT to BLCT3	3	0.0001
BLCT to BLCTMS	1	0.1
BLCT to BLCTZS	1	0
BLCT2 to BLCT3	5	0.0001
BLCT2 to BLCTMS	1	0
BLCT2 to BLCTZS	1	0
BLCT3 to BLCTMS	3	0.0001
BLCT3 to BLCTZS	5	0.1
BLCTZS to BLCTMS	5	0.0001

5.6.2 SVM

Recent literature [211] [221] and [187] show support vector machines (SVM) obtained remarkable result in travel time forecasting. It is essential to choose appropriate kernel function for SVM. Gaussian kernel is an example of radial basis function kernel which is commonly used and can obtain high efficiency and accuracy. SVM with Gaussian kernel model is also implemented in this work in order to compare predicting performance with our ARIMA method. Again, the time series data were scaled and split into training set which account for 2/3 of total records and test set which account for the rest 1/3 of total records. The value of kernel parameter sigma and cost are adjustable and plays a major role in the performance. In this study the value of sigma and cost are tuned and the model is evaluated by RMSE. The tuning parameters of optimised result are given in Table 5.5.

5.6.3 Comparison of three models

The forecasting performance which is measured by MARE (mean absolute relative error) of the three models are compared. The results of the com-

Table 5.5: The tunning parameters of optimized result (SVM)

Ports	Sigma	Cost
BLCT to BLCT2	4.3	1
BLCT to BLCT3	3.88	0.25
BLCT to BLCTMS	15.9	0.25
BLCT to BLCTZS	4.2	0.25
BLCT2 to BLCT3	5.47	1
BLCT2 to BLCTMS	11.2	0.25
BLCT2 to BLCTZS	18.7	0.25
BLCT3 to BLCTMS	6.32	0.25
BLCT3 to BLCTZS	6.81	1
BLCTZS to BLCTMS	27	0.25

parison are given in Table 5.6. Because NN and SVM models do not require time series data to perform the test therefore we apply original data set with equal period as data for ARIMA model to perform the experimental tests on NN and SVM models. Results in this study show that ARIMA model has the minimal MARE in each comparison which means it performed best, while single-hidden-layer neural network model performed worst and support vector machines are in between. Therefore we believe that ARIMA model together with data preparation method mentioned in Section 5.5.2 to estimate travel time parameters would be more suitable for this logistics planning system.

5.7 Summary

We have shown a study of automated container truck travel time prediction based on real-life GPS data using ARIMA. We also implemented a back-propagation NN model and a SVM with Gaussian kernel model and compared their forecasting performance with ARIMA. The results indicate that for the traffic data under drayage operation scenario, ARIMA model appears to perform best, indicating that ARIMA model together with the data preparation method discussed in this chapter is a more reliable approach to estimate travel time parameters for the container based logistics planning system.

The travel time prediction model and algorithm are suggested for generating feasible routes (see Section 4.3). When planning truck schedules, in the first step, the feasible route set is generated based on the recently obtained fleet GPS data (e.g. 3 months). In practice, this step does not have to be conducted frequently (e.g. we can run it once a week) and can be performed when the server is not fully occupied. For example, when planning truck schedules starting from Sep 2nd, using GPS data recorded from Jun 1st to Sep 1st, we can generate the feasible route set at the middle night of Sep 1st. In the next step, we solve (the solution method is given in Chapter 6) the container transportation problem for the next a few days using the set cover model given in Chapter 4. Of course, something unex-

Table 5.6: Comparison of MARE of three models.

Ports	ARIMA	SVM	NN
BLCT to BLCT2	1.29	2.67	3.07
BLCT to BLCT3	2.81	5.35	5.85
BLCT to BLCTMS	7.98	7.58	8.36
BLCT to BLCTZS	1.76	4.25	4.64
BLCT2 to BLCT3	3.13	5.56	5.98
BLCT2 to BLCTMS	12.31	11.64	13.78
BLCT2 to BLCTZS	2.51	4.70	5.01
BLCT3 to BLCTMS	1.29	3.43	3.72
BLCT3 to BLCTZS	2.08	4.45	5.02
BLCTZS to BLCTMS	4.04	5.72	6.51
AVG.	**3.92**	5.53	6.19

pected may happen any time in the real-life situation. For example, task might be changed and driver may not able to follow schedule strictly. In that case, one can either resolve the set covering model or apply heuristic method to reschedule the solution.

Unfortunately, as mentioned, the travel time prediction model is not integrated with the set covering model for further analysis, as only limited size of data obtained so that we are not able to obtain a connected graph with travel time between every pair of docks. For the travel time parameters used in experiments in Chapter 6 and 7, only part of travel times were estimated by analysing the GPS data, the rest were estimated by experience (i.e. travel time suggested by experienced manager in Port). The travel times are given in Table 5.7 and 5.8. In the next chapter, we propose a 3-stage hybrid method the solve the set covering model developed in Chapter 4.

Table 5.7: The travelling time between ports (part1)

	BLCT	BLCT2	BLCT3	BLCTYD
BLCT	0	15	50	50
BLCT2	15	0	50	5
BLCT3	50	50	0	0
BLCTYD	50	50	0	0
BLCTZS	40	40	40	40
DXCTE	40	40	40	40
BLCTMS	90	90	40	40
ZHCT	70	70	120	120
B2SCT	10	10	50	50

Table 5.8: The travelling time between ports (part2)

	BLCTZS	DXCTE	BLCTMS	ZHCT	B2SCT
BLCT	40	40	90	70	10
BLCT2	40	40	90	70	10
BLCT3	40	40	40	120	50
BLCTYD	40	40	40	120	50
BLCTZS	0	5	60	90	10
DXCTE	5	0	60	90	10
BLCTMS	60	60	0	180	90
ZHCT	90	90	180	0	60
B2SCT	10	10	90	60	0

Chapter 6

A 3-stage Hybrid Method for the Multi-shift Full Truckload VRP

6.1 Introduction

In general, this study concerns a multi-shift inter-dock container forwarding problem and the model formulated in Chapter 4 can be utilized to any drayage problem with multiple docks being operated simultaneously. However, the model has some problems in practice. The most critical one is the size of the feasible route set R which can increase exponentially with the number of nodes (or terminals). Additionally, small travel times (compared with a shift length) between nodes can also result in large R which causes long computational time. In the case of real-world problem (see Section 3.2) studied in this thesis, the size of R can be very large to be handled by a normal PC when the number of nodes exceeds 8 and the travel times between nodes are small (e.g. Less than 1 hour when shift length is 12 hours).

However, the real-world problem have certain special features to permit some nodes being merged. This chapter presents a 3-stage hybrid solution method for the set covering model presented in Chapter 4. The 3-stage hybrid method is: 1) **Pre-processing**: the original problem is reduced to a problem with a smaller number of nodes and commodities. 2) **Solving the reduced problem**: the reduced problem is solved in Gurobi solver. 3) **Post-processing**: finally the commodities that were temporarily excluded from the reduced problem were heuristically inserted into the existing routes whenever possible. A new truck route is opened if there are commodities that cannot be assigned to any existing routes.

In order to evaluate the feasibility and performance of our model and the 3-stage solution method, we applied them to solve real-life instances at Port of Ningbo. It was demonstrated that the model can be applied to solve real-life, medium sized instances of the container transport problem at a large international port. In addition, test instances with certain features were created to fully assess the approach and to gain knowledge that may not be discovered from real-life instances. The results are also compared with a reactive shaking variable neighbourhood search (VNS) and a simulated annealing hyperheuristic method (SAHH).

The 3-stage algorithm is found to be computationally expensive for large instances. Moreover, the algorithm becomes invalid for problems that do not possess these features to permit node merging. To address this issue, a more efficient and powerful hybrid branch-and-price approach is studied in Chapter 7. The idea is to use the pricing information to guide the generation of promising feasible routes dynamically.

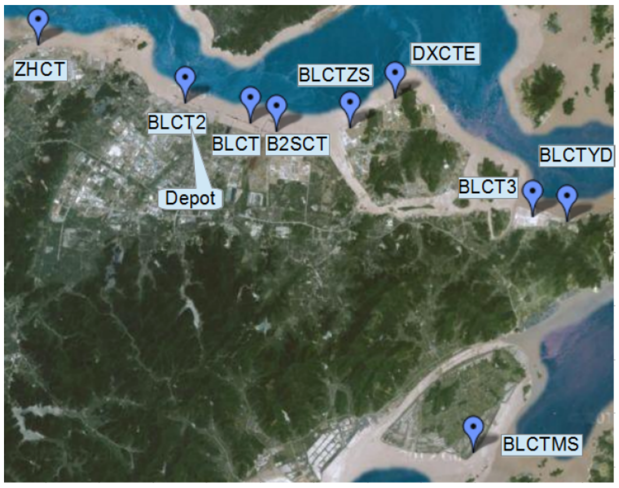

Figure 6.1: Positions of the container terminals in the port of Ningbo.

6.2 Pre-processing

The original problem contains a total of 9 docks/nodes (see Figure 6.1). The depot is at node BLCT2. We applied a recursive algorithm to generate all feasible routes that satisfy all the constraints in Section 4.3.

Note that precomputing all feasible routes is possible since the time related constraints in our problem are slightly different from those present in the traditional pickup and delivery problem with time windows (PDPTW). In our multi-shift FTL problem, each commodity k has an operation time window $(\sigma(k), \tau(k))$ defining its available time and the delivery deadline. Time constraints require that both the pickup and delivery operations occur within this time window for commodity k. While in PDPTW problems, two separate time windows are used, one for the pickup and the other for delivery. Note that for non-time critical full truckload transportation having one time window is reasonable since all the terminals (nodes) operate all the time and having short time windows for both pick and delivery do not make sense, although we acknowledge it is very different for the express deliveries who are mostly household customers.

Since the travelling time between some nodes are very short (e.g. 5 to 15 minutes), the algorithm generated millions of feasible routes on an 12-hour shift. This huge size of feasible routes will lead to prohibitively long solution time for our model in Section 4.4. Therefore, the original data was preprocessed before being populated into the model. All the nodes that are within 15 min travel times are merged into super nodes and its servicing time is set as the mean service time of the two original nodes. In the end, our reduced problem has a total of 6 nodes, including three super nodes {BLCT, B2SCT}, {BLCTZS, DXCTE} and {BLCT3, BLCTYD} and the depot. We did not merge the depot with BLCT2 as we need the depot in our route representation. Based on these 6 nodes, the feasible route generator returns a total of 43081 feasible routes. We then exclude all the commodities within the super nodes from the reduced problem. These commodities will be inserted into the final solution during the final stage of the proposed method.

6.3 Solving the reduced problem

Gurobi 5.6 was used to solve the reduced problem directly with the default algorithm setting in conjunction with Java 7.0. The experiments were run on a PC with Intel i7 3.40GHZ processor (8 cores) and 16GB RAM. Although the total distance is chosen as the objective for the mathematical model (see Section 4.4), the final solution is evaluated in terms of the heavy load distance rate (HLDR), which is the preferred efficiency indicator in practice. HLDR is defined as follows:

$$HLDR = \frac{\text{loaded distance}}{\text{total travel distance}} \tag{6.1}$$

Since containers are shipped to their destinations directly, the total amount of the loaded distance is fixed for a given commodity set K. Therefore, HLDR is equivalent to the objective function (4.11) as far as optimisation is concerned since minimising the total travel distance will also improve HLDR.

6.4 Post-processing

In the reduced problem, we excluded the nodes that have very few shipments and combined some nodes into super-nodes (and the shipments within the super nodes are excluded accordingly). Therefore, these commodities need to be inserted into the solution obtained from the reduced problem. For each route in the current solution, a total of 4 conditions have to be satisfied before a commodity is inserted into this route. Firstly, the route must have enough remaining time for additional commodities. Secondly, the insertion of a commodity must not affect the feasibility of the solution. Thirdly, the deadline of commodities within the super-nodes should be satisfied. Since the nodes in the super nodes are very close, most of intra-node shipments can successfully be inserted into the current solution. In a few cases where there is no feasible insertion, the procedure opens a new route. Finally, when multiple insertion points are available, the procedure favours the one that leads to the least empty load distance.

6.5 Benchmark Instances

In order to evaluate the feasibility and performance of our model, we applied it to solve real-life instances at a large international port (Port of Ningbo) in China. In addition, test instances with certain features were created to fully assess the approach and to gain knowledge that may not be discovered from real-life instances. These instances can be downloaded from http://www.cs.nott.ac.uk/˜rzb/research/transport/data/nbport.zip.

6.5.1 Real-life instances

A total of 15 real-life instances were extracted from the real-life problem data provided by the port. The original data contains three month demands from February to May 2012. Since the time window of most shipments ranges from 2 to 8 shifts, these instances have three planning horizons of 4, 6 and 8 shifts (the instance name, planning horizon and commodity size are given in Table 6.1).

6.5.2 Artificial instances

In addition to the real-life instances, we have also created a total of 17 artificial instances with controlled demand parameters in terms of commodity quantity, load (im-)balance and time windows. The number of available trucks is set to $n = 100$. The other parameters (nodes, distance matrix,

Table 6.1: Some details of the 15 real-life instances.

instance	**no. of shifts**	**total commodity units**
NP4-1	4	465
NP4-2	4	405
NP4-3	4	526
NP4-4	4	565
NP4-5	4	765
NP6-1	6	1073
NP6-2	6	920
NP6-3	6	384
NP6-4	6	746
NP6-5	6	557
NP8-1	8	913
NP8-2	8	827
NP8-3	8	786
NP8-4	8	1008
NP8-5	8	798

time matrix, operation time, etc.) remain the same. More specifically, we distinguish emergent tasks and non-emergent tasks. A transportation task is defined as **emergent** if the difference of its available time and deadline is less than 10 hours [*]. We also measure whether the transportation demand of a problem is balanced or not in both space and time through the following index:

$$B = \frac{1}{|V|} \sum_{i=1}^{|V|} \sum_{s \in S} |I_i^s - O_i^s|$$

where $|V|$ is the total number of physical nodes (i.e. docks). I_i^s and O_i^s are respectively the total incoming and outgoing commodities at node i during shift s. The following 4 types of features are used in creating the test instances.

- **Tight** instance: an instance is considered having "tight" time-windows if 70%-80% of its commodities are emergent.

[*]The 10-hour threshold is based on consultations with the port operators.

- **Loose** instance: an instance is defined as "loose" if up to 30% of its tasks are emergent.
- **Balanced** instances are defined by a balance index B. An instance is "balanced" if B is no more than 30.
- **Unbalanced** instance has a balance index B greater than 30.

With 2 different planning horizons (4, and 8 shifts), we have a total of 8 combinations. For each combination, 2 instances are created, resulting in a total of 16 instances. In addition, we also created a very large instance with 8 shifts and 2000 commodities. Details of these instances are given in Table 6.2.

Table 6.2: The list of artificial instances

instance	**configuration**	**no. of shifts**	**total commodity units**
LB4-1	Loose,Balanced	4	484
LB4-2	Loose,Balanced	4	396
TB4-3	Tight, Balanced	4	282
TB4-4	Tight, Balanced	4	368
LU4-5	Loose,Unbalanced	4	448
LU4-6	Loose,Unbalanced	4	479
TU4-7	Tight, Unbalanced	4	217
TU4-8	Tight, Unbalanced	4	354
LB8-1	Loose,Balanced	8	592
LB8-2	Loose,Balanced	8	657
TB8-3	Tight, Balanced	8	497
TB8-4	Tight, Balanced	8	621
LU8-5	Loose,Unbalanced	8	551
LU8-6	Loose,Unbalanced	8	559
TU8-7	Tight, Unbalanced	8	607
TU8-8	Tight, Unbalanced	8	525
Large	Mixed, Unbalanced	8	2614

6.6 Computational Results

For brevity, we denote our three-stage method as *hybrid* method. We compare its results against a reactive shaking variable neighbourhood search (VNS) (Chen et al. (2013) [31]) and a simulated annealing hyperheuristic method (SAHH) (Chen (2016) [30]). VNS and SAHH were run on a PC with 2.8GHz Xeon processor and 5GM memory. The computational time limit is 20 minutes per shift for VNS and 15 minutes per shift for SAH-H. Therefore, for a 4-shift instance, VNS requires 80 minutes and SAHH requires 60 minutes.

6.6.1 Computational results for real-life instances

Table 6.3: The HLDR results of our hybrid method for 4-shift instances.

	NP4-1	NP4-2	NP4-3	NP4-4	NP4-5
total distance	13508.5	16635.5	16878.5	21886	26731
time(s)	33301	15742	11178	18537	20647
hybrid	**89.2%**	69.2%	**78.6%**	**70.0%**	79.3%
VNS	83.2%	69.2%	77.1%	68.5%	80.7%
SAHH	83.2%	**69.3%**	76.2%	69.0%	**80.8%**

hybrid:the 3-stage hybrid method;
VNS:variable neighbourhood search metaheuristic;
SAHH:simulated annealing hyperheuristic;
time(s):runtime of the hybrid method in seconds;
Results are presented as **HLDR** (see Section 6.3).

The computational results for the real-life instances are given in Tables 6.3, 6.4 and 6.5. Values in **bold** represent the best results. It can be seen from the table that for most of the instances, the hybrid 3-stage method outperformed both VNS and SAHH. Taking NP4-1 as an example, the improvement is as much as 6.0%, which translates into nearly 1000km saving

Table 6.4: The HLDR results of our hybrid method for 6-shift instances.

	NP6-1	NP6-2	NP6-3	NP6-4	NP6-5
total distance	34054.5	33316	16191.5	26260	16880.5
time(s)	160079	138486	3978	58898	104446
hybrid	**82.7%**	**75.0%**	**66.1%**	**80.5%**	**83.2%**
VNS	79.3%	72.9%	64.2%	80.3%	77.7%
SAHH	80.5%	74.1%	65.8%	80.2%	78.8%

hybrid:the 3-stage hybrid method;
VNS:variable neighbourhood search metaheuristic;
SAHH:simulated annealing hyperheuristic;
time(s):runtime of the hybrid method in seconds;
Results are presented as **HLDR** (see Section 6.3).

Table 6.5: The HLDR results of our hybrid method for 8-shift instances.

	NP8-1	NP8-2	NP8-3	NP8-4	NP8-5
total distance	35685	30633	28314	44224	25451.5
time(s)	148067	147241	121074	66438	131369
hybrid	72.5%	**76.8%**	**76.7%**	61.7%	**75.8%**
VNS	74.6%	74.1%	76.1%	**62.1%**	74.6%
SAHH	**74.7%**	74.7%	76.0%	**62.1%**	73.3%

hybrid:the 3-stage hybrid method;
VNS:variable neighbourhood search metaheuristic;
SAHH:simulated annealing hyperheuristic;
time(s):runtime of the hybrid method in seconds;
Results are presented as **HLDR** (see Section 6.3).

in distance in 2 days. We can be fairly confident in saying that, overall, the proposed hybrid method could produce better solutions to these real-life instances when compared to the recent multi-neighbourhood metaheuristic approaches.

For 4 instances (NP4-2, NP4-5, NP8-1, and NP8-4), the hybrid method is outperformed by either SAHH or VNS. However, the margin is relatively small. Note that our hybrid method does not guarantee the optimal solution because of the approximations made in the first and last stage of the approach. More specifically, in the first stage of the hybrid method (see

Section 6.2), nodes BLCTZS and DXCTE were merged as a super node and its service time was set to the mean value of the service times of BLCTZS and DXCTE. However, because the service times for BLCTZS and DXCTE are very different (in our setting, they are 60 minutes and 5 minutes respectively), using the mean service time (33 minutes) could lead to either infeasibility (which needs to be handled in stage 3) or inferior solutions. In order to mitigate this problem, we replaced the simple mean service time for the super node to the weighted average service time, with the weights being proportional to the number of operations at the corresponding physical nodes. Therefore, all the remaining experiments in Section 6.6.2 were conducted with this new setting.

6.6.2 Computational results for artificial instances

The computational results for the artificial instances by different algorithms are given in Table 6.6. The best results are highlighted in **bold**. It can be seen that the 3-stage hybrid method was able to obtain best overall objective results for all the instances except two, whose performance seems to be improved further by the weighted average service time. Again the better performance was achieved at the expense of more computational time. Generally, the proposed method can solve most "tight" instances fairly quickly but struggled for some of "loose" instances. This is not surprising since "tight" time-window constraints actually help speedup the search by producing better bounds for an integer programming solver. This is in direct contrast to the metaheuristic methods which often struggle for highly con-

Table 6.6: The computational results by the hybrid method for artificial instances in comparison with VNS and SAHH [31].

instance	**hybrid method**			**VNS**	**SAHH**
	HLDR(%)	distance	time(s)	HLDR(%)	HLDR(%)
LB4-1	**77.6**	15763.0	13438	77.1	76.4
LB4-2	**83.4**	14319.0	3812	79.3	78.1
TB4-3	**68.9**	10866.5	1415	67.5	67.9
TB4-4	**72.4**	12507.5	186	67.1	66.7
LU4-5	**63.2**	18499.5	1590	59.3	58.8
LU4-6	65.3	20315.5	1783	66.8	**67.2**
TU4-7	**49.2**	13032.5	79	46.6	46.6
TU4-8	**54.3**	17024.5	138	51.8	51.8
LB8-1	**95.1**	18132.5	138988	94.1	93.0
LB8-2	**88.7**	22834.0	157354	88.1	87.8
TB8-3	**67.8**	21337.5	148	66.7	66.8
TB8-4	**61.6**	28167.0	561	61.3	61.1
LU8-5	**71.7**	21226	4380	61.4	61.9
LU8-6	**67.9**	23261.0	13202	65.1	64.7
TU8-7	**60.9**	31094.0	140	53.2	53.2
TU8-8	**49.9**	27406.0	66	48.5	48.5
Large	n.a.*	n.a.*	48h	56.1	**56.5**

*:The algorithm fails to solve the problem after 48 hours.
hybrid:the 3-stage hybrid method;
VNS:variable neighbourhood search metaheuristic;
SAHH:simulated annealing hyperheuristic;
time(s):runtime of the hybrid method in seconds;
Results are presented as **HLDR** (see Section 6.3).

strained instances. When the problem size, in terms of commodity size, increased to over 2000, the proposed hybrid method failed to solve the problem while both VNS and SAHH can still produce feasible solutions with significantly less computational time. In this sense, the proposed hybrid method is not a complete replacement for, but rather a complementation to the existing metaheuristic methods. The proposed algorithm would be the preferred solution method for small or tightly constrained instances while large and less constrained instances should be solved by the existing metaheuristics.

In terms of the transportation efficiency (i.e. HLDR), it can be observed that generally a balanced demand can contribute to a higher HLDR, which is consistent with the observation made by Chen et al. (2013) [31]. Also it can be seen that "tightness" in the time window of transportation tasks affected the HLDR negatively. The reasons are twofold: firstly, similar to the vehicle routing problem with time windows, there are less flexibilities to coordinate different transportation loads to improve HLDR when a task is highly constrained by time. Secondly, although overall transportation demand in each shift may be balanced, tight time windows of tasks will cause unbalanced demand during that particular time window, which leads to a low HLDR.

6.6.3 Compared with lower bound

Table 6.7: The results of our hybrid algorithm for the real-life instances when compared with the lower bound (see Section 4.5).

Instance	Lower bound	hybrid algorithm	gap(%)
NP4-1	13322.0	13508.5	1.4
NP4-2	16386.0	16635.5	1.5
NP4-3	16663.5	16878.5	1.3
NP4-4	20754.0	21886.0	5.2
NP4-5	26121.0	26731.0	2.3
NP6-1	33566.0	34054.5	1.4
NP6-2	32550.0	33316.0	2.3
NP6-3	16000.5	16191.5	1.2
NP6-4	26096.5	26260.0	0.6
NP6-5	16639.0	16880.5	1.4
NP8-1	33568.0	35685.0	5.9
NP8-2	30333.0	30633.0	1.0
NP8-3	27420.5	28314.0	3.2
NP8-4	43617.0	44224.0	1.4
NP8-5	25350.0	25451.5	0.4

*gap(%)=(ObjectiveValue-LowerBound)/ObjectiveValue * 100%

For the real-life instances it can be seen from the Table 6.7 that the

Table 6.8: The results of our hybrid algorithm for the random artificial instances when compared with the lower bound (see Section 4.5).

Instance	**Lower bound**	**hybrid algorithm**	**gap(%)**
LB4-1	15383.5	15763.0	2.4
LB4-2	13837.0	14319.0	3.4
TB4-3	8914.0	10866.5	18.0
TB4-4	10198.5	12507.5	18.5
LU4-5	15770.5	18499.5	14.8
LU4-6	17817.0	20315.5	12.3
TU4-7	9998.0	13032.5	23.3
TU4-8	14565.5	17024.5	14.4
LB8-1	17601.5	18132.5	2.9
LB8-2	20656.0	22834.0	9.5
TB8-3	16620.5	21337.5	22.1
TB8-4	18772.0	28167.0	33.4
LU8-5	20507.5	21226.0	3.4
LU8-6	21999.0	23261.0	5.4
TU8-7	26922.5	31094.0	13.4
TU8-8	24227.0	27406.0	11.6

*gap(%)=(ObjectiveValue-LowerBound)/ObjectiveValue * 100%

results of the proposed method are very close to the lower bounds. For some instances, the difference (gap%) is smaller than 2%. This is particularly true for the instances with relatively low HLDR (e.g. NP4-2, NP6-3 and NP8-4). It is indeed the demand imbalance that caused low transport efficiency. For some instances, the gaps to the lower bound are bigger (e.g. NP8-1). However, this observation cannot be repeated for the random artificial instances (see Table 6.8), for which the gap can be as large as 33.4%. This suggests that many artificial instances have very different characteristics to those shown by the real-life instance. Although small gap to the lower bound can prove the high performance of the algorithm, one cannot conclude that a big gap to the lower bound implies poor solutions. This is because the bound for these instances may be poor. Generally the gap is much bigger for tight instances than for loose instances. Indeed, the time window relaxation made in lower bound model leads to very different problems for the original tight instances. In addition, the lower bound model also excluded the constraint of using the depot as the sole departure node at the start of each shift, whose inclusion may have caused long-distance empty truck returning to the depot. For real-life instances, the depot is in fact very close to busier ports and in most cases trucks return to the depot fully loaded. For these cases, tighter bounds are needed. This would be out of the scope of this study but could be an interesting research direction in the future.

It should be noted that one would only need to use the 3-stage hybrid method when the problem is too large to be handled directly. For small and moderate instances (i.e. less than 7 nodes), only the second stage is required and the exact solutions can be obtained.

6.6.4 Computational time

Table 6.9: A comparison of HLDR results by the hybrid method against metaheuristics with longer computational time (slow version). The results of the fast version of VNS and SAHH are from Section 6.6.

			Fast version		**Slow version**	
instance	**time (s)**	**hybird**	**VNS**	**SAHH**	**VNS**	**SAHH**
NP4-1	33301	**89.2**	83.2	83.2	82.9	83.2
NP6-3	3978	**66.1**	64.2	65.8	63.3	65.3
NP8-3	121074	**76.7**	76.1	76.0	76.0	76.0
LB4-1	13438	**77.6**	77.1	76.4	76.6	76.8
LB4-2	3812	**83.4**	79.3	78.1	79.4	79.4
LB8-1	138988	**95.1**	94.1	93.0	92.9	91.6
LB8-2	157354	88.7	88.1	87.8	89.9	**90.8**
LU8-5	4380	**71.7**	61.4	61.9	67.6	67.7
LU8-6	13202	**67.9**	65.1	64.7	64.2	66.0

hybrid:the 3-stage hybrid method;
VNS:variable neighbourhood search metaheuristic;
SAHH:simulated annealing hyperheuristic;
time(s):runtime of the hybrid method in seconds;
Results are presented as **HLDR** (see Section 6.3).

As indicated in the previous section, the hybrid method requires more than 10 hours computation for many instances (except some of tightly time-constrained instances). However, the computational time by both VNS and SAHH is less than 20 minutes per shift. For a fairer comparison, additional experiments were also carried out with same amount of computational time permitted for VNS and SAHH on some instances. Due to the long experiments time, a subset of 9 instances were selected for this experiment. 3 of them were selected from real-life instances for which the proposed hybrid method outperformed metaheuristics with shorter running time. The rest were chosen from artificial instances for which the hybrid algorithm took longer time in solving than the metaheuristics did. In this particular experiment, both VNS and SAHH were permitted the same amount of running time by the hybrid algorithm (see Table 6.9) and their results are also given in the same table, along with the previous results with a short computational time (i.e. 20 minutes per shift).

The results showed that, with the additional computational time, both VNS and SAHH are able to improve the results for some instances (e.g. LB8-2, LB8-5). However, for many other instances, they fail to make noticeable improvement. In fact, to our surprise, some of the results are even marginally worse than before (e.g. NP4-1, NP6-3, LB8-1, LU8-6). We believe that this was caused by the fact that the parameters by both VNS and SAHH were finely tuned for the previous setting only and do not perform well for the new setting. This sensitivity in parameters, again, is a common criticism for many metaheuristic approaches. It's interesting to observe that SAHH slightly outperformed VNS both for the fast version and the slow version. We believe this was probably contributed by the learning mechanism within the simulated annealing hyperheuristic that provides better adaptation than VNS does by dynamically choosing between different neighbourhood functions for different instances and experiment conditions. A more profound analysis of neighbourhood selection and adaptation will be out of the scope of this study. Readers are encouraged to refer to the latest hyperheuristic research which has gradually gained more and more research attention recently.

In terms of practical applications, although the hybrid method may be too long for direct utilisation, the problem can be resolved through multi-core parallel computing facilities which have recently become available at acceptable costs either through rented cloud services or through building a moderate low-cost cluster.

6.7 Summary

This chapter presented a 3-stage hybrid method for the set covering model proposed in Section 4.4. It was found that for real-life instances, the solution obtained from the set covering model is very close to the lower bound, suggesting that the time window may not be the driving factor for the low transport efficiency but the demand imbalance between different ports is. For the artificial instances, the model can be solved efficiently for most "tight" instances but is found to be computationally expensive for instances with "loose" time windows.

The results were also compared with a reactive shaking variable neighbourhood search (VNS) (Chen et al. (2013) [31]) and a simulated annealing hyperheuristic method (SAHH) (Chen (2016) [30]). Both VNS and SAHH were able to obtain feasible but inferior solutions with less computational time compared with the 3-stage hybrid method.

The computational time for optimally solving large sized problems was prohibitive. To circumvent this problem, the original data was pre-processed to reduce the problem size before being populated into the model. The solution cannot be globally optimum as some intra-node shipments are not processed by the model. The real-life problem has some special features to permit some of nodes being merged. However, in addition to the excessive computational time by the hybrid algorithm, it may become even invalid for problems that do not possess these features to permit for nodes merging. To address this issue, in the next chapter, a more efficient hybrid branch-and-price approach is studied as it is an effective integer programming method for problems with larger number of columns. It is a potentially very good method for the problem formulation stated in this chapter when the feasible route set is very large.

Chapter 7

A hybrid Branch-and-price Method for the multi-shift full truckload VRP

7.1 Introduction

The real-life problem has some special features to permit the hybrid solution method proposed in previous chapter being used. However, in addition to the excessive computational time by the hybrid algorithm, it may become even invalid for problems that do not possess the features possessed by this problem. To address this issue, in this chapter, a more efficient hybrid branch-and-price approach is studied.

In this chapter we propose a hybrid branch-and-price method to address the multi-shift full truckload vehicle routing problem (MFTLVRP) using the set covering model presented in Chapter 4. The underlining idea is to use the pricing information to guide the generation of the promising feasible routes dynamically. We propose to use heuristics for this column generation subproblem. The reasons are twofold. Firstly, the reduced cost of the routes cannot be directly computed from the dual problem of the set covering model because of the solution encoding scheme. Secondly, even if the accurate cost information can be obtained, the subproblem to compute the optimal column to add in the restricted master problem is also NP-Hard (see 2.2.5).

Two rounds of experiments were conducted. For the first round of experiments, we consider instances with relatively small route set R. As such, all instances in the first round of experiments have seven nodes, resulting in around 60000 feasible routes which is close to the limit to which our model can be solved directly. For larger instances, the feasible route set R can become very big and therefore it becomes impossible to enumerate them all as we did in the first round of experiment. In the second round of experiments, we investigate the effectiveness and performance of two meta-heuristic approaches for column generation: variable neighbourhood search (VNS) and genetic algorithms (GA).

7.1.1 The proposed branch-and-price framework

Branch-and-price (B&P) is an effective integer programming method for problems with large number of variables, most of which are non-basic in the optimal solutions. It is a potentially very good method for the relaxed problem formulation stated in Section 4.4, where the feasible route set R is very large, leading to a model with a huge number of columns while the optimal solution is a very small subset of it. We propose to use the branch-and-price approach for this problem in which the sub-problem (pricing problem) is solved to identify the variables that should enter the basis.

In this application, the constraint (4.12, ensures the availability of truck-s) is not used in our pricing subproblem. It is not possible to obtain a feasible solution if constraint (4.12) is violated, therefore, the number of trucks is always large enough in order to yield feasible solutions. In that case, the duals (prices) of constraint (4.12) are always equal to 0, meaning such resource is not completely utilised and always available. The price of constraint (4.14, ensures commodity flow on a node does not exceed the arc capacity) varies in different routes but is the same as the price of each commodity (constraint (4.13)) that can be serviced by that node (see Section 4.4.1). That means the price of constraint (4.14) can be derived from those of constraint (4.13). Therefore, both sets of constraints (4.12) and (4.13) are not used in our pricing subproblem. As a result, only price information related to constraint (4.13) is used.

A route is constructed by arcs that forwarding commodities, the reduced cost of a route thus depends on the price of the arcs included within the route. When calculating the reduce cost of a route, we do not have to determine which shift that a route belongs to, as the feasible route set is the same for all shifts. The price of an arc can be estimated by the price of all possible commodities (for all shifts) that can be serviced by that arc. Therefore, some approximation is needed to estimate the price of an arc (using price of constraint (4.13)) so it is not a branch-and-price method in the traditional sense. Despite reword features that were discussed in the previous sections, this is a slight drawback of our model, leading to an inexact B&P method. The branching process is required as the column generation does not automatically guarantee integer solutions. The overall branch-and-price framework is outlined in here, followed by detailed steps of the procedure.

The integer programming formulation presented in section 4.4 is also referred to as the master problem. The Restricted Master Problem (RMP) is the master problem that considers only of a subset of truck routes R that are generated by the pricing problem (subproblem) using the dual information obtained from the Linear Programming Relaxation (LPR) of the RMP. The pricing problem and the LRP will be discussed in section 7.1.5 and section 7.1.3 respectively. Before the RMP is solved for the first time, no dual information is available and an initial truck routes set (see section 7.1.2) is thus required to start the process. Then the LPR is solved to optimality and the dual information is obtained for calculating the reduced costs of routes during the pricing subproblem.

However, the pricing subproblem in this formulation is not quite s-traightforward. As mentioned earlier, because the price information is only available for commodity assignment variables (x_{ri}^{ks}), prices of the columns (routes) can only be estimated based on the possible commodity flows each of them can carry, resulting in an inexact solution method. Because of this, the standard branch-and-price procedure is also adapted, in which multiple routes with negative reduced costs are added to the RMP at each pricing subproblem iteration. It is hoped that by doing this the total number of RMP calls can be reduced. This process is repeated until the stopping criteria are met. The column generation approach does not automatically guarantee integer solutions so some of the decision variables are fractional, therefore, the branching process (see 7.1.4) is required.

Because the pricing problem is solved repeatedly in the branch-and-price framework, it is crucial that the solution algorithm for the pricing subproblem is as efficient as possible. Therefore, we propose two different strategies, one for problems with small-sized R and one for problems with a large R. For the former case, we propose to adapt an explicit enumerative generation of R priori and then try to solve the pricing subproblem when no column with negative reduced cost can be found. We apply a recursive algorithm to generate all feasible routes (as described in Section 4.3) before the start of the branch-and-price algorithm. In the case of a large R, researchers tend to use heuristic approaches (see Section 7.1.6) to solve the pricing problem. In this chapter, both methods are investigated. The whole branch-and-price framework as described above is outlined in Figure 7.1.

7.1.2 Initial set of routes

Before the RMP is solved for the first time, no dual information is available and an initial set of columns is required to start the process. We apply two methods described in detail in the next two subsections to generate an initial set of columns (routes).

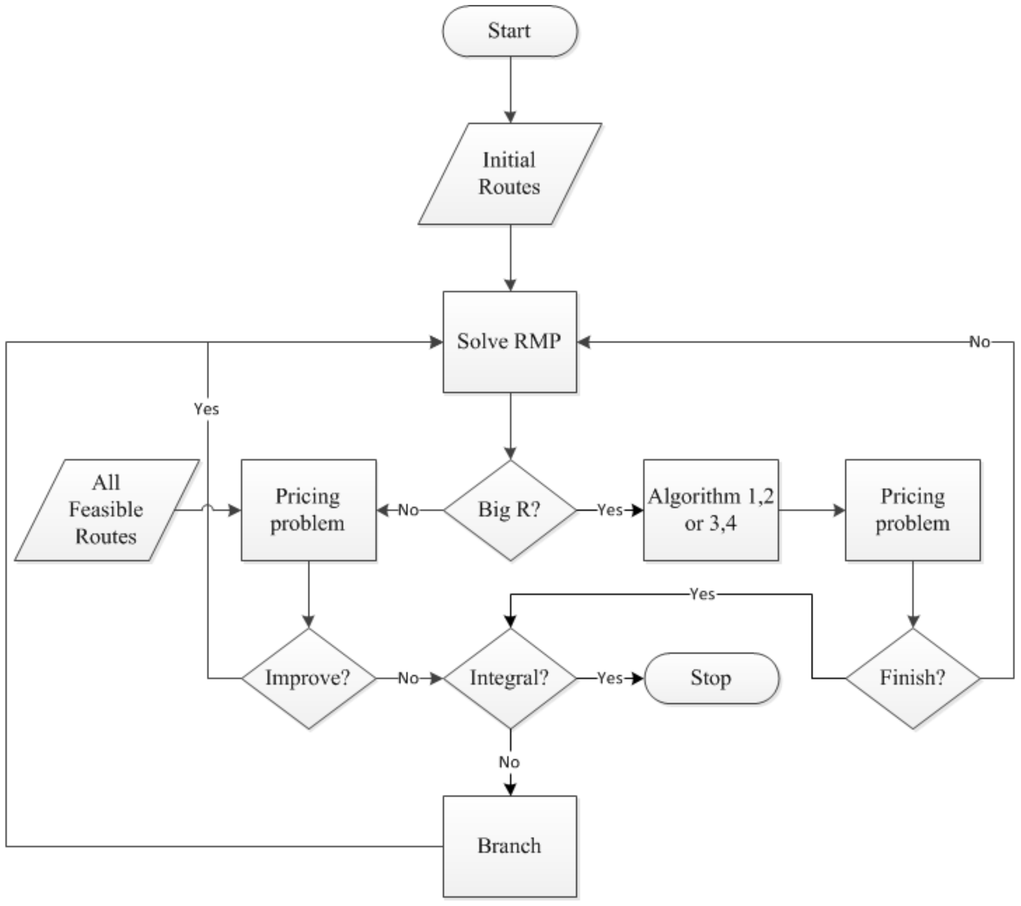

Figure 7.1: Framework of the branch-and-price process

Simple route initialisation

A prerequisite of constructing a basic route set is to ensure that each commodity has at least one route to service it. Thus the simplest solution is to generate a dedicated route for each commodity, in which an empty truck leaves the depot and travels to the source of a commodity, loads the commodity and delivers it to its destination. After that, the truck returns to the depot. This method works fine but may of course lead to an infeasible solution in terms of maximum number of vehicles constraint (4.12).

Insertion heuristic method

The advantage of the basic route initialisation in the previous section is its simplicity. However, very rarely will these routes be used in the optimal solutions, neither do they resemble any of the routes that are present in the optimal solutions. In this study, we propose to use constructive heuristic methods to generate these initial routes for our branch-and-price method.

The insertion heuristic has been demonstrated to be a good starting solution method heuristic for the VRP. In particular, we used the same insertion heuristics described in Chen et al. (2013) [31]. To construct routes, the task that cause minimum empty load distance is inserted by following two initialisation criteria: First, the most urgent tasks that have deadlines closer to the shift start time are inserted into the initial empty routes. The second criterion considers tasks that have earlier availability time.

More specifically, to generate routes for a shift s, the insertion heuristic first tries to create initial routes by assigning one task to each trucks route based on initialization criteria. Then, it inserts mandatory tasks of the current shift to the initial routes. After all mandatory tasks of the current shifts are inserted, the algorithm will try to assign tasks from the optional task set of s. Instead of picking a task from the whole optional task set for the current shift, the algorithm only picks the mandatory tasks from the next shift. It should be noted that these mandatory tasks in the next shift still belong to the optional task set for the current s. This insertion process is carried out until all trucks have an initial task in s or there is no more tasks can be assigned in s.

As shown by the following experiments (see Section 7.3.2), there are two benefits here. First, because the constructive heuristic produces a feasible solution for the original problem, the vehicle routes extracted from the solution shall also produce a feasible solution in our branch-and-price method, satisfying the maximum number of vehicles constraint. Second, because the pricing subproblem is solved heuristically, starting from a good set of initial vehicle routes will enable the branch-and-price method to generate high quality solutions more quickly compared to the simple route initialisation method. The proposed method will converge to a high quality solution much faster.

7.1.3 Linear programming relaxation (LPR)

The linear programming relaxation (LPR) problem of our branch-and-price method is as follows, in which discrete variables are relaxed to continuous ones.

$$\min \sum_{s} \sum_{r} d_r y_r^s \tag{7.1}$$

subject to

$$\sum_r y_r^s \leq n \quad \forall s \in S \tag{7.2}$$

$$\sum_s \sum_r \sum_i x_{ri}^{ks} = Q(k) \quad \forall k \in K \tag{7.3}$$

$$\sum_k x_{ri}^{ks} \leq y_r^s \quad \forall i \in r, \forall r \in R', \forall s \in S \tag{7.4}$$

$$x_{ri}^{ks} \geq 0 \quad \forall i \in r, \forall r \in R', \forall k \in K, \forall s \in S \tag{7.5}$$

$$y_r^s \geq 0 \quad \forall r \in R', \forall s \in S \tag{7.6}$$

7.1.4 Branch strategies

Branching is an important part of the branch-and-price method as the column generation approach does not guarantee integer solutions and often the solutions that are obtained are fractional [103]. For classical vehicle routing problems with time windows (VRPTW), a typical branching strategy is branching on a fractional decision variable θ_r (indicating whether an arc r is used in the solution or not). In our problem, the decision variable route y_r^s is not binary, therefore, the branching strategy of VRPTW cannot be directly adopted.

Branching on arc flow is another popular strategy in the context of VRPTW. For a graph $G = (N, A)$ there is an arc set $A = \{(i, j) : i, j \in N\}$ where N is the node set standing for customers. One can branch an arc (i,j) where the flow is fractional: one branch where the arc cannot be included in the solution and another branch where the arc is enforced in the solution. Again, this method of branching on arc flow is not suitable for the problem studied in this thesis, because the node set N in the problem under study is physical terminals instead of customers (i.e. delivery tasks). As an arc in this problem services all commodities sharing the same source and destination terminals, branching an arc is equivalent to branching all commodities that are possible to be serviced by that arc.

Another commonly used strategy is to branch on the number of vehicles used in the solution. If the number of vehicles used is fractional, two branches are generated: one concerns the upper limit which is rounded down from the fractional value, and another concerns the lower limit which is rounded up of the fractional value. However, this may result in an infeasible solution [66]. One may refer to [103] for more branching strategies.

The problem in this thesis has two sets of decision variables x_{ri}^{ks} and y_r^s. Initial investigations show that branching on y_r^s gives stronger bounds than branching on x_{ri}^{ks}, as the number of possible values for y_r^s is much less than that of x_{ri}^{ks}. In this study, we first branch on y_r^s with fractional values. Two branches are created: one sets the upper limit of the rounded-up fractional values, and another for the lower limit which is the rounded-down value of the fractions. If no fractional y_r^s is found we then consider to branch on fractional x_{ri}^{ks} using the same strategy as branching y_r^s. Note that branching on y_r^s might lead to infeasible solutions. If this happens, we abandon it and branch on x_{ri}^{ks} only. If all x_{ri}^{ks} values are integer, an integral solution of the problem is obtained as the integral of arc flow is satisfied and the frequency of route be used in a shift y_r^s is also integral. The development of search tree of branch-and-bound (e.g. fathoming) is left to Gurobi solver.

7.1.5 Pricing methods

As explained earlier, the pricing information gathered from the LPR model is for arc assignment variables x_{ri}^{ks}, based on which the cost of a route is estimated. In this research, we investigated three different route price estimation methods: The first method enumerates and examines all possible task assignments for each vehicle route and the best route (i.e. the route with the most negative reduced cost) is selected. This method is referred to as `pricing by enumeration`. For efficiency, two other pricing estimation methods are also investigated. They are `average pricing` and `demand weighted average`.

Pricing problem by enumeration. Let π_k^* denote the optimal dual value (i.e. price) of commodity k (constraint 7.3) and d_r be the cost (in this case the distance) of route r. Let $W = \{w_1, w_2, ...,\}$ be the set of all possible commodity combinations, each of which can be delivered by one instance of route r. In the example of the route in Figure 7.2, all possible commodity combinations are $W = \{[1,4], [1,5], [2,4], [2,5], [3,4], [3,5], ...,\}$.

The `reduced cost` (often negative, standing for the potential reduction in the objective value in RMP) of a route r is estimated by the smallest reduced cost among all commodity combinations:

$$\min_i c_r = d_r - \sum_{k \in w_i} \pi_k^* \tag{7.7}$$

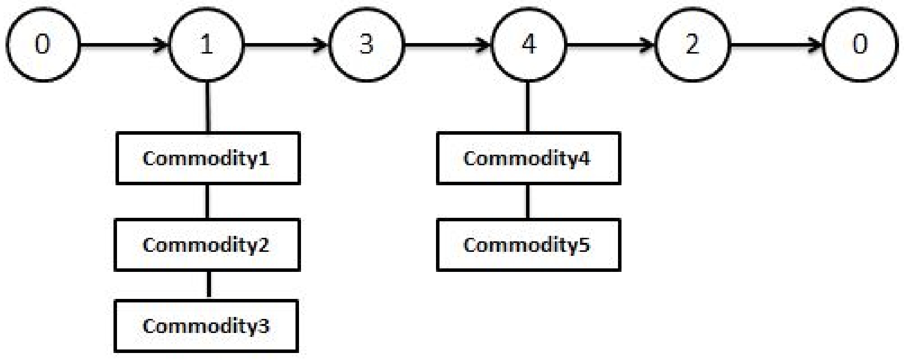

Figure 7.2: Example of a routing sharing among five commodities.

In practice to speed up the process, as soon as c_r becomes negative for a given w_i, route r is added to the RMP. This process guarantees that the reduced cost of each route is examined but its efficiency is low as some W may contain thousands of commodity combinations and the procedure is computationally too expensive. This motivates us to investigate the following two other methods.

P1: average pricing Instead of enumerating all the commodity combinations of a route and then checking the c_r for each of w_i, a more efficient approach is to use the average prices to estimate c_r approximately. More specifically, let J be the set of all service nodes in r (e.g. nodes $\{1, 4\}$ in Figure 7.2). Denote V_j be the set of all commodities that can be serviced by a node j in r. The reduced cost c_r for route r is calculated by the following equation:

$$c_r = d_r - \sum_{j \in J} \left(\frac{1}{|V_j|} \sum_{k \in V_j} \pi_k^* \right) \tag{7.8}$$

P2: demand weighted average Though the commodities processed by a service node in a route share the same source and destination node, the quantity of the commodities varies from one to another. The simple average pricing method P1 fails to take into account the quantity of the commodities, so that large quantity commodities may be left unpaired to improve the efficiency. Therefore, the demand weighted average method tries to give priority to large commodities at the early stage. A weight ω_k that is proportional to the commodity quantity $Q(k)$ is used. The weighted average pricing method uses the following equation to estimate the reduced cost.

$$c_r = d_r - \sum_{j \in J} \sum_{k \in V_j} \omega_k \pi_k^* \tag{7.9}$$

7.1.6 Heuristic column generator for large set R

As can be seen from Figure 7.1, a heuristic column generator is used within the branch-and-price framework. As optimally solving the pricing problem involves an expensive recursive tree search, we propose to use a variable neighbourhood search (VNS) and a genetic algorithm (GA) to tackle the pricing subproblem. The goal of the metaheuristics is to identify new columns with negative reduced costs. The underlining idea is that, instead of generating a new column (i.e. route) from scratch, it is probably more efficient to search from the existing routes through either neighbourhood moves or route combinations (i.e. crossovers). VNS and GA are widely adopted frameworks to implement these ideas. The main difference here is that the metaheuristics are guided by an objective function that heavily relies on the pricing information obtained from the linear program relaxations.

VNS

The pseudo-code of our VNS algorithm is given in Algorithm 1 and the parameters of the algorithms are listed in Table 7.1. In our VNS method, the neighbourhood functions include *swap*, *2-opt*, and *relocate*. These operators are very similar to those used in solving the classical VRP problems. For example, the *swap* operator swaps two arcs of two different routes. The *2-opt* exchanges two nodes on the same route. The *relocate* operator relocates an arc from its current route to a different one. By exploring different neighbourhood structures, the method has an increased probability to detect more diversified routes than a single neighbourhood. The neighbourhood functions are called one by one in turn. Once a neighbourhood function can no longer find a better set of routes, the next neighbourhood is called. If, however, a better solution (e.g. a more negative reduced cost) is found, the algorithm will restart from the first neighbourhood.

Table 7.1: Abbreviations of VNS

$\mathbf{z}$	starting solution
R_z	a set of routes present in $\mathbf{z}$
i	index of neighbourhood
i_{max}	index of the last neighbourhood function
c_{min}	minimum reduced cost of route set
c'_{min}	modified minimum reduced cost of route set
$maxIteration$	max number of column generation iterations
$maxColumns$	max number of routes

Algorithm 1 Pseudo-Code of VNS column generator

```
Require: $\mathbf{z}$, $maxIteration$
  $j \leftarrow 0$;
  while $j < maxIteration$ do
      $columnPool \leftarrow VNS(\mathbf{z})$;//Algorithm2
      $\mathbf{z} \leftarrow RMP(columnPool)$, $j \leftarrow j + 1$;
  end while
  return $\mathbf{z}$
```

Algorithm 2 Pseudo-Code of VNS()

```
Require: $\mathbf{z}$, $i_{max}$, $maxColumns$
  $i \leftarrow 1$, $R_z \leftarrow z$, $c_{min} \leftarrow 0$;
  while $i \leq i_{max}$ do
      $R' \leftarrow neighbourhood(R_z, i, maxColumns)$;
      $c'_{min} \leftarrow minReducedCost(R')$;
      if $c'_{min} < c_{min}$ then
          $i \leftarrow 1$, $c_{min} \leftarrow c'_{min}$;
          $columnPool \leftarrow sortByReducedCost(R', maxColumns)$;
          $columnPool \leftarrow columnPool \cup \mathbf{z}$;
      else
          $i \leftarrow i + 1$;
      end if
  end while
  return $columnPool$
```

Our VNS aims to find out a set of feasible routes with the most negative reduced costs to be solved by the RMP. The VNS based column generator is not conventionally implemented as a single point search method. The search is guided by the pricing methods described in Section 7.1.5. The *neighbourhood(R,i,maxColumns)* function applies the i-th neighbourhood function on all routes in R_z to search for new feasible routes. The constraints related with feasible route pattern (see Section 4.3) are imposed. It returns the maximum of $maxColumns$ distinct routes with negative reduced cost. Function $minReducedCost(R')$ returns the minimum reduced cost of route set R'. Function $sortByReducedCost(R_z \cup R', maxColumns)$ sorts routes in $R_z \cup R'$ by their corresponding reduced costs in an ascending order and returns the top $maxColumns$ distinct routes. The $RMP(R_z \cup \mathbf{z})$ is the restricted master problem (see Section 7.1.3) based on the route set R_z and previous solution $\mathbf{z}$, the solution is stored in vector $\mathbf{z}$.

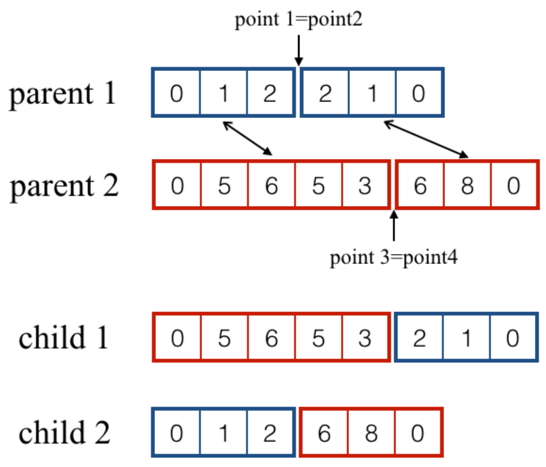

Figure 7.3: Example of two-point crossover 1

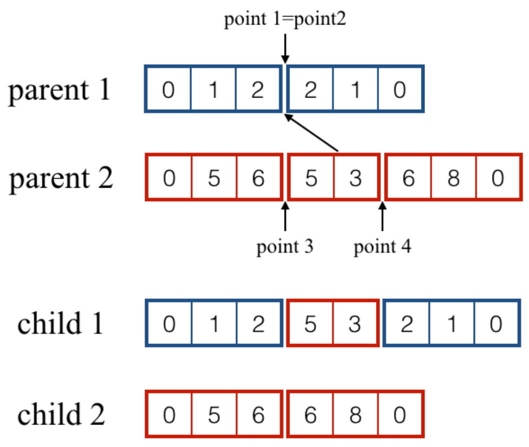

Figure 7.4: Example of two-point crossover 2

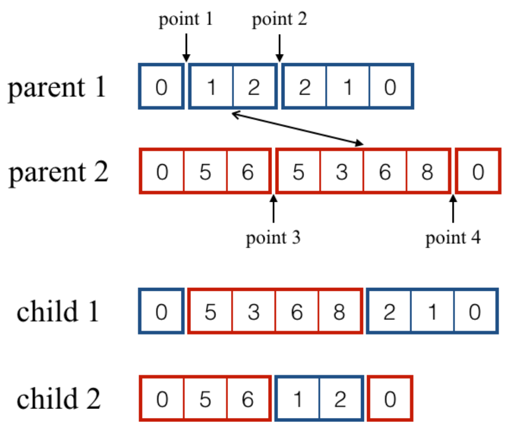

Figure 7.5: Example of two-point crossover 3

GA

We also investigate a Genetic Algorithm (GA) approach to tackle the pricing subproblem. The motivations are two-fold: first, at each branch-and-price iteration, we need to obtain a set of routes with the most negative reduced costs, which the VNS may struggle to achieve as a single point search method. The GA is potentially more powerful as it can find a population of routes through evolution. Secondly, we believe that high quality routes (i.e. most reduced costs) may share some common structures which could be evolved more efficiently through crossover operations in the genetic algorithm. Therefore, each chromosome in our generic algorithm stands for a vehicle route, leading to a variable length chromosome.

The pseudo-code for the GA search is given in Algorithm 3. The initial population is generated by using the insertion heuristic by Chen et al. (2013) [31]. The size of the initial population for each RMP iteration is equal to the number of distinct vehicle routes used in the solution $\mathbf{z}$ but increased to a pre-defined value *populationSize* in the following generations. Other implementation details of our GA are as follows. Two-point crossover operators were adopted. The length between two crossover points is randomly generated from 0 to 2 arcs, as larger crossover length would increase the possibility of generating infeasible routes due to the violation of routes' travel time constraint. Figures 7.3, 7.4 and 7.5 illustrate examples of the two-point crossover. A standard mutation operator is used in which each chromosome is subject to an uniform *2-opt* mutation with probability *mutationRate*. The 2-opt mutation operator is the same as the 2-opt neighbourhood moves in our VNS method. A local search stage is incorporated into our GA to ensure that local optima are reached in each generation. The local search is performed every time when new individuals have been generated. More specifically, the local search phase swaps two nodes between two different routes and returns two new routes that are local optimal with regard to the neighbourhood.

We use the tournament selection method. As the first population is obtained by the insertion heuristic, it usually has smaller population size than the predetermined constant value ($populationSize$). The tournament size is set to $populationSize \times tournamentRate$ so that it is population dependent. The fitnesses of individuals are calculated according to the functions in Section 7.1.5. Note that only feasible routes that satisfy the time constraints are considered and evaluated. If their fitnesses are better than any of the routes in the $columnPool$ which stores the set of best routes so far, they replace the inferior routes in the $columnPool$, to allow a maximum of $maxColumns$ columns to be stored. Finally, the algorithm terminates when the number of RMP iterations reaches a predefined parameter, $maxIterations$. The pseudo-code of the proposed GA is given in Algorithm 3.

Note that although the main framework of our GA is the same as many other GA implementations, the goal is very different. Our GA here does not solve the overall problem, but rather evolves a set of vehicle routes (columns) with the most negative reduced costs. These set of routes will then be used in solving the updated RMP problems. In general, GA search will be more robust if the population contains more various individuals, as population diversity will encourage the exploration phase of the search and prevent the population from converging prematurely to local optima. Premature convergence occurs when the population of a GA reaches a suboptimal state that the genetic operators can no longer produce offspring that outperform their parents. Therefore, improving diversity of the population is always an implicit goal of almost any basic evolutionary computation algorithms. However, maintaining a high population diversity usually decreases convergence speed. In this particular application, a fast convergence of GA is preferred because the best solution for GA will not necessarily lead to the best overall performance in the branch-and-price solution procedure. Of course, it would be an interesting future research to have a more advanced GA.

Algorithm 3 Pseudo-Code of GA column generator

```
Require: $maxIterations$, $\mathbf{z}$, $generations$, $populationSize$, $columnPool$, $maxColumns$
  while $i < maxIterations$ do
      $R \leftarrow \mathbf{z}$, Clear $columnPool$;
      while $j < generations$ do
          $R \leftarrow generateNewPolulation(\mathbf{z}, R, populationSize)$;//Algorithm 4
          $j \leftarrow j + 1$;
      end while
      $\mathbf{z} \leftarrow RMP(columnPool)$;
      $i \leftarrow i + 1$;
  end while
  return $\mathbf{z}$
```

7.2 Experiments on small set R

For the first round of experiments, we consider instances with relatively small R. As such, all instances in the first round of experiments have seven

Algorithm 4 Pseudo-Code of $generateNewPolulation()$

```
Require: current solution $\mathbf{z}$, current population $R$, empty new population
  $R'$, $populationSize$, $tournamentRate$, $mutationRate$

  // Ensure $R$ include $\mathbf{z}$
  for each $r$ in $\mathbf{z}$ do
      if $r$ not in $R$ then
          $R.add(r)$
      end if
  end for

  while $R'.size < populationSize$ do
      // Selection
      $r_1 \leftarrow$ tournamentSelection($populationSize, tournamentRate, R$);
      $r_2 \leftarrow$ tournamentSelection($populationSize, tournamentRate, R$);

      // Crossover
      $R'' \leftarrow$ crossover($r_1$,$r_2$);

      // Mutation
      $R'' \leftarrow$ mutation($R''$,$mutationRate$);

      // Local Search
      $R'' \leftarrow$ localSearch($R''$);

      // Evaluation
      for each $r$ in $R''$ do
          if fitness($r$)<0 then
              $R'$.add($r$);
              updateColumnPool($r$);
          end if
      end for

  end while
  return $R'$
```

nodes, resulting in a total of 61365 feasible routes which is close to the limit to which our model can be solved directly. Therefore, we can compare how our methods perform in comparison with exact methods.

A set of randomly generated instances are used in the experiments. These instances are generated based on characteristics of real-life instances which are obtained from historically scheduled container operation data of a truck company. All artificially generated instances have three planning horizons of 4, 6, 8, reflecting the different problem scenarios in practice. These instances were grouped into three sets. All the instances are generated by the same parameters except the size of the planning horizon. Five instances are generated for each problem set, referred to as I4, I6 and I8, standing for shift length of 4, 6 and 8, respectively. The information and configuration of these problem sets is illustrated in Tables 7.2 and 7.3.

Table 7.2: Configuration of artificial instance

Node Number:	7 (including the depot).
Commodity Time Window:	1-2 hours up to the length of planning horizon.
Commodity Available Time:	nearly 30% commodities' available time are set before the planning horizon.
Emergency:	10% to 30% emergent commodities (the time window is less than 10 hours).

In order to test the efficiency of the column generation process in the first round of experiments, the initial route set is constructed by the simple method detailed in Section 7.1.2. Since the RMP solving will take the majority of computational time, at each iteration, we add multiple columns in the RMP model (capped by $maxColumns$). If $maxColumns$ is set too small, more RMP solving calls are required which are computationally very

Table 7.3: Detail of artificial instances

Instance	no. of shift	no. of commodity	no. of unit
I4-1	4	51	360
I4-2	4	56	340
I4-3	4	50	266
I4-4	4	87	624
I4-5	4	71	305
I6-1	6	77	489
I6-2	6	79	564
I6-3	6	94	581
I6-4	6	105	783
I6-5	6	99	818
I8-1	8	106	888
I8-2	8	120	831
I8-3	8	106	939
I8-4	8	124	1067
I8-5	8	127	971

expensive. However, if the $maxColumns$ is set too large, time to solve each RMP would also increase (the extreme case is that all feasible columns are included in RMP and it is equivalent to the original problem). Some initial experiments suggest that $maxColumns = 1000$ provides a good trade-off. We use this value on the understanding that it may not be the best parameter for every instance. Note that in our method, in the early search stage, we permit our method to use more trucks than the limit (n), but this constraint will later be restored at the end of the column generation procedure (before the branching stage). Gurobi 5.6 linear programming libraries were used in conjunction with Java 7.0. These experiments were run on a PC with an Intel i7 3.40GHZ processor and 16GB RAM.

The experimental results are given in Table 7.4. Since the pricing by enumeration method (see Section 7.1.5) takes an unrealistically long time even for the smallest instances (e.g. 3-4 hours for a 4-shift instance), it is not used for further experiments. Column **T** is the total running time of

Table 7.4: Comparison of two pricing methods (artificial data)

	BP1			BP2			Gurobi			Random
Instance	T	Obj.	Col.	T	Obj.	Col.	T	Obj.	Obj.*	Obj.
I4-1	23	15516	4691	21	**14154**	3005	1215	13746	n.a.	22530
I4-2	93	16480	9448	151	**15988**	5976	1208	15823	n.a.	20743
I4-3	3	12793	2281	6	**11067**	4319	155	11037	n.a.	15885
I4-4	271	27557	4674	711	**25642**	8811	1736	25307	28819	33583
I4-5	187	13407	4021	343	**11435**	6430	1193	11429	14624	25798
I6-1	131	27566	7742	193	**25540**	13985	1153	24713	29542	34589
I6-2	175	26719	3046	507	**23374**	4861	2772	21665	n.a.	29294
I6-3	87	32009	3142	513	**30124**	5321	2604	30029	n.a.	31889
I6-4	218	41301	2170	290	**35935**	3321	9462	33898	n.a.	44497
I6-5	172	33799	2040	420	**30207**	3216	5406	29223	n.a.	n.a.
I8-1	276	53871	2863	694	**50178**	2724	14890	49797	n.a.	58269
I8-2	323	38589	2199	958	**33532**	3361	17202	32668	n.a.	41667
I8-3	213	44856	3539	970	**39643**	2701	10006	38108	n.a.	n.a.
I8-4	479	35850	2022	919	**32307**	2286	36132	31979	n.a.	39778
I8-5	213	45066	2414	704	**39911**	3455	19170	37979	n.a.	51476
Avg.	191	31025	3753	493	**27936**	4918	8287	27160		

Bold values: represent the best results;
BP1:branch-and-price method 1; **BP2**:branch-and-price method 2;**T**:Total running time(s);
Gurobi: Result obtained by Gurobi solver using all feasible routes;
Random: Result obtained by Gurobi solver using randomly selected columns;
Col.:Total columns generated; **Obj.**:Objective Value(km);
n.a.:Failed to find feasible solution in given time;
Obj.*:Objective value obtained by limited run time;
Comment: Overall, the table shows the importance of the pricing subproblem.
BP2 generates more columns than BP1, resulting in longer running times. Compared with Gurobi solver, BP1 and BP2 use significantly less time with competitive solutions.

the entire process, from data parsing, solving, to the solution output. **Col.** shows the total number of columns being generated during the process. **Obj.** gives the objective value which is the total travel distance. Hereafter, **BP1** and **BP2** are short abbreviations for branch-and-price solution methods adopting **P1 average pricing** and **P2 demand weighted average** respectively (see Section 7.1.5).

Overall, the results (based on the average of 5 runs.) in Table 7.4 show that most instances are solved in 1000s or less. In most cases, BP2 generated a larger number of columns than BP1 during the branch-and-price process. On average, BP2 generates 1165 (or 23.7%) more columns than BP1, resulting in longer running times, but BP2 uses 3089km (or 10%) less distance than BP1. Seemingly, this fact is due to BP2 generating more columns that enlarge the search space used by the model. However, we notice that for the result of instances I4-1, I4-2 , I8-1 and I8-3, BP1 obtained a larger number of columns which did not result in a smaller objective value.

The performance of both algorithms is also compared with the results from the Gurobi IP solver with the default algorithm setting in two experiments. The first experiment allows the solver to solve the problem to optimality and its objective value is denoted as *Obj.*. In the second experiment, Gurobi was given a limited computational time (the same time taken by the slowest of BP1 and BP2) and the corresponding objective value is marked as *obj.**. All the results are given in Table 7.4.

It can be seen that although Gurobi can solve all instances to optimality, it takes more than 8000s on average and sometimes more than 10h. In contrast, the proposed branch-and-price methods (BP1 and BP2) use significantly less time with competitive solutions. This is particularly true for BP2 as, on average, it uses around 5.0% of the time used by Gurobi but produces solutions that are only 776km (or 2.8%) away from optimality. On the other hand, if we reduce computational time, for many instances Gurobi fails to produce a feasible solution. Between BP1 and BP2, BP1 generates less columns and is faster, but produces inferior solutions for most instances.

In order to evaluate the usefulness of the pricing subproblem, we conducted another set of experiments that replaces it with 1000 randomly selected columns of the RMP at each iteration. All the other settings were kept the same as before. Column **Random** in Table 7.4 presents the objective values based on the average of 5 runs. The results are significantly inferior to those by BP1 or BP2, which shows the importance of the pricing subproblem. we believe 5 runs are enough to show the pricing subproblem performs much better than random setting in this round of experiments.

7.2.1 Compare with previous results

In order to further analyse the algorithms designed for small route set R, we also test them on both the real-life and artificial instances used in Chapter 6. Because the number of docks (nodes) is 9, which leads to a large number of feasible routes exceeding the limit our model can handle directly, the 3-stage hybrid method used in Chapter 4 is applied.

Experiments on real-life instances

The results from comparing two pricing methods using real-life data are outlined in Table 7.5. Overall, most instances are able to be solved within 20 minutes, except in instances with large numbers of commodities (e.g. NP6-1, NP6-2, NP8-3, NP8-4). In most cases, BP2 generated a larger number of columns than BP1, differing by a mean value of 346 (or 13.4%). As a result, a longer running time of 414s (or 36.6%) is required by BP2. On average, BP2 obtained a 423.5km (or 1.61%) shorter (better) objective

Table 7.5: Comparison of two pricing methods (real-life data)

	BP1			BP2		
Instance	T	Obj.	Col.	T	Obj.	Col.
NP4-1	221	**13401.0**	2045	340	13552.0	2032
NP4-2	397	**16599.0**	2398	535	16703.5	2441
NP4-3	159	17340.0	1697	176	**17248.0**	1850
NP4-4	249	22051.5	2405	498	**21743.5**	2704
NP4-5	832	26485.5	2106	807	**26284.0**	2195
NP6-1	1201	**34118.0**	2521	2951	34174.0	3187
NP6-2	1526	**33652.0**	2773	2808	33994.0	3278
NP6-3	59	**16984.0**	2566	145	17217.0	3194
NP6-4	227	33396.0	1013	788	**26442.0**	2002
NP6-5	497	**16814.0**	2060	585	16906.0	2052
NP8-1	654	34267.0	2195	1078	**33957.0**	2453
NP8-2	881	**30601.0**	2188	1189	30816.0	2944
NP8-3	1837	**28361.0**	3240	2147	28608.0	3309
NP8-4	1248	44340.5	2244	1717	**44297.5**	2908
NP8-5	767	**25460.5**	2002	1196	25576.0	2088
Avg.	717	26258.1	2230	1131	**25834.6**	2576

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Col.:Total columns generated; **Obj.**:Objective Value(km);
T:Total running time(s);
Bold values: represent the best results.

value than BP1. Note that for the result of instance NP4-2, BP2 obtained a larger number of columns that did not result in smaller objective values. In addition, the number of columns generated by BP1 in some instances (e.g. NP6-1, NP6-2, NP6-3, NP8-5) were less than BP2 but obtained better objective values. This confirms the previous analysis and indicates that solution quality is not very relative to the number of columns generated by the pricing problem but rather the quality of them.

The results also show that BP1 performed unstably, as for nearly half of the tests, BP1 obtained less objective value than BP2. However, the value is 33396km for NP6-4, which is nearly 7000km (or 21%) longer than the result obtained by BP2. These results suggest that BP2 is a more reliable pricing method, as BP2 takes transportation demand (i.e. quantity of the commodities) into account.

Table 7.6: Comparison with previous results (real-life data)

Instance	BP1	BP2	Hybrid	SAHH	VNS
NP4-1	**0.59%**	1.70%	1.38%	7.94%	7.83%
NP4-2	1.28%	1.90%	1.50%	1.26%	**1.25%**
NP4-3	3.90%	3.39%	**1.27%**	4.14%	2.77%
NP4-4	5.88%	**4.55%**	5.17%	6.27%	6.94%
NP4-5	1.38%	0.62%	2.28%	0.45%	**0.36%**
NP6-1	1.62%	1.78%	**1.43%**	4.43%	4.67%
NP6-2	3.27%	4.25%	**2.30%**	3.28%	3.72%
NP6-3	5.79%	7.07%	**1.18%**	1.51%	3.96%
NP6-4	21.86%	1.31%	**0.62%**	0.83%	0.67%
NP6-5	**1.04%**	1.58%	1.43%	6.52%	7.30%
NP8-1	2.04%	**1.15%**	5.93%	1.55%	1.79%
NP8-2	**0.88%**	1.57%	0.98%	3.12%	4.13%
NP8-3	3.32%	4.15%	**3.16%**	3.62%	3.62%
NP8-4	1.63%	1.54%	1.37%	**0.74%**	0.77%
NP8-5	0.43%	0.88%	**0.40%**	3.18%	1.52%
Avg.	3.93%	2.36%	**2.14%**	2.96%	3.12%

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Hybrid:the 3-stage hybrid method;
SAHH:simulated annealing hyperheuristic;
VNS:variable neighbourhood search metaheuristic;
Results are presented as **Gap** (see Section 7.10);
Bold values: represent the best results.

A comparison was also made with previous results obtained by the 3-stage hybrid method (hybird), metaheuristic methods and lower bound reported in Chapter 6. Table 7.6 presents the objectives obtained by the **Hybird** method, **BP1**, **BP2**, and hyperheuristic methods (Chen (2016) [30]) including simulated annealing hyperheuristic (**SAHH**) and variable neighbourhood search (**VNS**). In order to facilitate observation and comparison, all results are compared with the lower bound and transformed to

Gap which is defined as follows:

$$Gap = \frac{\text{objective value-lower bound}}{\text{objective value}} \tag{7.10}$$

Table 7.7: Running time (s) comparison with previous results (real-life instances)

Instance	BP1	BP2	Hybird	SAHH	VNS
NP4-1	**221**	340	33301	310	4800
NP4-2	397	535	15742	**386**	4800
NP4-3	**159**	176	11178	590	4800
NP4-4	**249**	498	18537	545	4800
NP4-5	832	**807**	20647	1022	4800
NP6-1	**1201**	2951	160079	1613	7200
NP6-2	1526	2808	138486	**955**	7200
NP6-3	**59**	145	3978	211	7200
NP6-4	**227**	788	58898	698	7200
NP6-5	497	585	104446	**492**	7200
NP8-1	**654**	1078	148067	822	9600
NP8-2	881	1189	147241	**869**	9600
NP8-3	1837	2147	121074	**878**	9600
NP8-4	**1248**	1717	66438	1631	9600
NP8-5	**767**	1196	131369	1128	9600
Avg.	**717**	1131	78632	810	7200

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Hybrid:the 3-stage hybrid method;
SAHH:simulated annealing hyperheuristic;
VNS:variable neighbourhood search metaheuristic;
Bold values: represent the best results.

In terms of gaps, the average results are exhibited in increasing order as follows: Hybrid < BP2 < SAHH < VNS < BP1. The best results are shown in bold. The solving time of the hybrid method ranges from 4 to 12 hours, which is much longer than BP2. The computational time for SAHH is between 4-10 minutes per shift and VNS is allowed to run up to 20 minutes for each shift. The running times in Table 7.7 reveal that the best objective value is obtained by the hybrid method, and the BP2 is able to find better objective value in a shorter amount of time compared to SAHH and VNS metaheurstic methods.

Table 7.8: Intermediate objective values (km) of real-life instances

Instance	BP1	BP2	Hybird
NP4-1	13810	13810	**13431**
NP4-2	17371	17383	**16789**
NP4-3	17517	17499	**16960**
NP4-4	22488	22466	**22228**
NP4-5	28361	28251	**27555**
NP6-1	38368	38368	**37019**
NP6-2	37460	37720	**35510**
NP6-3	17832	17918	**16726**
NP6-4	26504	26546	**26352**
NP6-5	17490	17425	**16829**
NP8-1	34184	33604	**33604**
NP8-2	33421	32979	**30657**
NP8-3	30533	30533	**29424**
NP8-4	44733	44671	**44470**
NP8-5	**25355**	25645	**25355**
Avg.	27028	26988	**26194**

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Hybrid:the 3-stage hybrid method;
Bold values: represent the best results.

Unexpectedly, the results also indicate that for some instances (e.g. NP4-1, NP4-2, NP6-5, NP8-1, NP8-2), the branch-and-price method obtained smaller travelling distance than the hybrid approach. As the branch-and-price method only processed a subset of total feasible routes applied by the hybrid approach, the possibility of the branch-and-price method producing a better objective value than the hybrid approach is small. These differences can be explained by the post-processing stage of the hybrid method modifying the original solution set by inserting excluded commodities, generating new routes and recalculating route distances. The comparison with intermediate results (without post-processing) of branch-and-price and hybrid methods are given in Table 7.8 which indicate that the hybrid method obtained better intermediate results for every instance. Again, in general, the intermediate result obtained by BP2 is better than BP1.

These tests on real-life instances highlight that even though the hybrid method is able to find the best solutions, BP2 offered comparable results in a much smaller computation time and obtained better solutions than the metaheuristic methods.

Table 7.9: Comparison of two pricing methods (artificial instances)

	BP1			BP2		
Instance	T	Obj.	Col.	T	Obj.	Col.
LB4-1	303	**15951.0**	2065	469	**15951.0**	2154
LB4-2	191	14490.0	2054	533	**14442.0**	2025
TB4-3	178	10976.0	2910	461	**10866.5**	3767
TB4-4	402	n.a.	3346	497	**13468.5**	3647
LU4-5	200	n.a.	2649	932	**18499.5**	3594
LU4-6	316	21706.0	2695	1173	**21263.5**	3696
TU4-7	24	14643.0	2058	306	**13701.5**	3025
TU4-8	250	n.a.	2824	483	n.a.	3752
LB8-1	11539	**18132.5**	6279	12792	**18132.5**	7481
LB8-2	5839	**22834.0**	4257	6092	**22834.0**	4605
TB8-3	1355	24567.0	2272	2155	**21508.5**	3129
TB8-4	439	31319.0	2673	566	**28167.0**	3689
LU8-5	730	n.a.	2419	1725	**24234.0**	3909
LU8-6	421	27235.0	2009	1739	**26341.5**	3601
TU8-7	1258	n.a.	3490	1839	n.a.	2928
TU8-8	404	n.a.	2740	1418	n.a.	2521
Large	69345	116599.0	5008	118392	**116396.5**	7677

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Col.:Total columns generated; **Obj.**:Objective Value(km);
T:Total running time(s);
Bold values: represent the best results;
n.a.:Fails to find feasible solution with truck number restriction.

7.2.2 Experiments on artificial instances

In addition to the real-life instances, we also test the proposed branch-and-price methods for small route set R on the artificial instances created in Chapter 6. Computational results are given in Table 7.9. Generally speaking, BP2 obtains better solutions than BP1. BP2 generates more columns than BP1, hence more computation time is required. This is consistent with our earlier findings. For the test on large instance, the algorithms perform better than the metaheuristic methods.

As mentioned in Section 7.1.2, for some experiments (denoted as n.a. in obj. column), the algorithms fail to obtain feasible solutions owing to the violation of truck number constraint. In order to tackle an infeasible solution problem, a constructive heuristic method can be used (see Section 7.1.2) to generate these initial routes. Another strategy to deal with infeasible solution is to modify pricing methods (to be discussed in Section 7.2.2).

Again, the performance of algorithms is also compared with those in the previous study solved by the 3-stage hybrid method (hybird), metaheuristic methods and lower bound. As shown in Table 7.10, with the exception of infeasible solutions, the results are corroborated with previous findings and gaps are shown in increasing order: Hybrid $<$ BP2 $<$ SAHH $<$ VNS $<$ BP1. The running times of those algorithms are summarised in Table 7.11.

These tests suggest that the hybrid method performs well in tight instances (e.g. TB4-4, TB8-3, TB8-4, TU8-7, TU8-8) but suffers in large and loose instances (e.g. LB8-1, LB8-2, Large). The reason is the hybrid method adopts an integer programming solver so that its solving time in-

Table 7.10: Comparison with previous results (artificial instances)

Instance	BP1	BP2	Hybrid	SAHH	VNS
LB4-1	3.56%	3.56%	**2.41%**	3.03%	3.92%
LB4-2	4.51%	4.19%	**3.37%**	8.11%	9.51%
TB4-3	18.79%	**17.97%**	**17.97%**	19.64%	19.16%
TB4-4	n.a.	24.28%	**18.46%**	24.43%	24.88%
LU4-5	n.a.	**14.75%**	**14.75%**	20.01%	20.69%
LU4-6	17.92%	16.21%	12.30%	10.28%	**9.75%**
TU4-7	31.72%	27.03%	**23.28%**	30.46%	27.34%
TU4-8	n.a.	n.a.	**14.44%**	18.24%	18.38%
LB8-1	**2.93%**	**2.93%**	**2.93%**	3.95%	5.07%
LB8-2	**9.54%**	**9.54%**	**9.54%**	10.15%	10.46%
TB8-3	32.35%	22.73%	**22.11%**	23.37%	23.26%
TB8-4	40.06%	**33.35%**	**33.35%**	33.68%	33.90%
LU8-5	n.a.	15.38%	**3.38%**	17.27%	16.59%
LU8-6	19.23%	16.49%	**5.43%**	9.32%	9.88%
TU8-7	n.a.	n.a.	**13.42%**	24.33%	24.36%
TU8-8	n.a.	n.a.	**11.60%**	13.97%	14.08%
Large	11.96%	**11.80%**	n.a.*	29.32%	28.81%

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Hybrid:the 3-stage hybrid method;
SAHH:simulated annealing hyperheuristic;
VNS:variable neighbourhood search metaheuristic;
Results are presented as **Gap** (see Section 7.10);
Bold values: represent the best results;
n.a.*:The algorithm fails to solve the problem within 48h.

creases exponentially with large problem size (i.e. loose and large instances). The proposed branch-and-price methods are able to reduce problem size, therefore, compared with the hybrid method, the solving time of branch-and-price is significantly decreased, especially for large problems. In terms of metaheuristics, SAHH can produce a competitive solution in less time.

Deal with infeasible solutions

As mentioned, for some tests (denoted as n.a. in obj. column in Table 7.9), the algorithms fail to obtain feasible solutions due to the violation of truck number constraint . In fact, the reduced cost of columns are approximately

Table 7.11: Running time (s) comparison with previous results (artificial instances)

Instance	BP1	BP2	Hybird	SAHH	VNS
LB4-1	303	469	13438	**292**	4800
LB4-2	**191**	533	3812	414	4800
TB4-3	**178**	461	1415	288	4800
TB4-4	n.a.	497	**186**	383	4800
LU4-5	n.a.	932	1590	**276**	4800
LU4-6	316	1173	1783	**233**	4800
TU4-7	**24**	306	79	232	4800
TU4-8	n.a.	n.a.	**138**	491	4800
LB8-1	11539	12792	138988	**1899**	9600
LB8-2	5839	6092	157354	**1266**	9600
TB8-3	1355	2155	**148**	2602	9600
TB8-4	**439**	566	561	2391	9600
LU8-5	n.a.	1725	4380	**915**	9600
LU8-6	**421**	1739	13202	1204	9600
TU8-7	n.a.	n.a.	**140**	484	9600
TU8-8	n.a.	n.a.	**66**	434	9600
Large	69345	118392	n.a.*	15848	**9600**

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Hybrid:the 3-stage hybrid method;
SAHH:simulated annealing hyperheuristic;
VNS:variable neighbourhood search metaheuristic;
Results are presented as **Gap** (see Section 7.10);
Bold values: represent the best results;
n.a.:Fails to find feasible solution due to the violation of truck number restriction;
n.a.*:Fails to solve the problem within 48h.

evaluated by the pricing problems which may neglect part of the columns that lead to feasible solutions. In order to circumvent this problem, we test two strategies on the instances that failed to be solved in previous experiments.

Strategy 1: Increase threshold of searching for negative reduced cost

The threshold of finding negative reduced cost can be increased to enlarge the search space of the pricing algorithms. For example, searching for columns with reduced cost that is less than a positive value (this value can be tuned optimally for each instance) instead of 0, so that more columns can be found by the pricing problems, enhances the chances of obtaining a feasible solution. Inevitably, this consumed more computation time.

Table 7.12: Effect of increasing threshold of evaluating reduced cost to 200

	Instance	T	Obj.	Gap	Col.
BP1	TB4-4	2682	12940.5	21.19%	18086
	LU4-5	2661	18531.5	14.90%	17140
	TU4-8	3118	18071.0	19.40%	24022
	LU8-5	3201	23416.0	12.42%	12844
	TU8-7	2771	32592.5	17.40%	12719
	TU8-8	3072	27586.0	12.18%	16948
BP2	TU4-8	3131	17795.0	18.15%	28414
	TU8-7	2850	32772.5	17.85%	14975
	TU8-8	3118	27541	12.03%	18495

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Col.:Total columns generated;
Obj.:Objective Value(km);
T:Total running time(s);
Results are presented as **Gap** (see Section 7.10).

For example, Table 7.12 gives results of increasing the threshold value from 0 to 200 for the instances that failed to be solved in previous experiments (see Table 7.9). It can be seen that the algorithms can obtain feasible solutions for all instances.

Strategy 2: Give priority to long columns

Increasing the threshold of finding negative reduced cost can randomly enlarge the search space of pricing algorithms. Another strategy is to guide the pricing problem to find the column that is capable of serving more commodities. As discussed, the reason behind the infeasible solution is the limited number of trucks. Therefore, a truck has to handle more commodities during a route in order to save truck usage. For this reason, pricing methods have to find the routes that are able to deliver more commodities. In order to do that, we guide the pricing problem to give priority to routes serving more commodities in the first few iterations of the branch-and-price framework.

Table 7.13: Effect of giving priority to long columns

	Instance	T	Obj.	Gap	Col.
BP1	TB4-4	513	13081.5	22.04%	5549
	LU4-5	411	18634.0	15.37%	5823
	TU4-8	858	18847.0	22.72%	7354
	LU8-5	637	24341.0	15.75%	5525
	TU8-7	760	36711.0	26.66%	6147
	TU8-8	534	27831.0	12.95%	6344
BP2	TU4-8	1530	18833.0	22.66%	12455
	TU8-7	1551	34495.0	21.95%	12438
	TU8-8	1682	27622.0	12.29%	14420

BP1:branch-and-price method 1;
BP2:branch-and-price method 2;
Col.:Total columns generated;
Obj.:Objective Value(km);
T:Total running time(s);
Results are presented as **Gap** (see Section 7.10).

Let o be the reward of serving one more commodity in the route if the number of commodities that the route currently serving is greater than a base number b. Recall the reduced cost equation illustrated in Section 7.1.5. Let m indicate the number of commodities that node i can service in route r and n denotes the number of service nodes in r. $\overline{\pi_i}$ is the dual value of a node i. For a route, its reduced cost c_r is calculated by the following equation:

$$c_r = d_r - \sum_{n} \overline{\pi_i} - o * m \qquad (m > b) \tag{7.11}$$

For example, we set the value of o to 20 and b to 3 for the first 5 iterations and test the algorithms on the instances that failed to be solved in previous experiment and the results are given in Table 7.13.

The results of Tables 7.12 and 7.13 show that the solving time of the first strategy is longer than the second one due to the larger number of columns generated by the first strategy. However, the first strategy also obtained better solution. Compared with the results given in Tables 7.10 and 7.11, we find that, with the exception of BP1, which required a longer computation time, the performance of the proposed two strategies are consistent with previous experiments.

In this section, we firstly examined two pricing methods BP1 and BP2 using a set of artificial instances that only involve 7 nodes (docks). In order to further analyse the algorithms, we have also tested them on both real-life and artificial instances that having 9 nodes. The 3-stage hybrid method proposed in Chapter 6 is used to reduce the problem of 9 nodes into 6 nodes so that the feasible route set is enumerable. From this test we found that the algorithms failed to obtain a feasible solution for some artificial instances owing to the violation of truck number constraint. For this reason, we proposed two strategies that are able to tackle the problems: 1) Increase threshold of searching for negative reduced cost. 2) Give priority to long columns. In fact, as mentioned in Section 7.1.2, an easier way is to apply constructive heuristic method to generate initial routes for our branch-and-price method. This method will be adopted in the next round of experiments in next section. As the feasible routes set R used by now is enumerable and within the limit that our PC can handle directly, we name it as small R. Next, we deal with large set R that the number of columns is too large to be enumerated.

7.3 Experiments on instances with a very large R

For larger instances, the feasible route set R can become very big and therefore it becomes impossible to enumerate them all as we did in the previous section. In this section, we investigate the effectiveness and performance of the two metaheuristic approaches presented in Section 7.1.6. As the evidence from previous experiments suggest BP2 performs better than BP1, for the remaining experiments only BP2 is used. Similar to the previous section, maximum $maxColumns = 1000$ columns are allowed to be generated by both VNS and GA at each iteration. As $maxColumns$ is the only parameter used in the proposed VNS based column generator, parameter tuning for VNS is omitted. We now illustrate parameter tuning for the GA.

7.3.1 Parameter Tuning for GA

The parameters used in the proposed GA are the population size (*populationSize*), the generation size (*generations*), the probability of mutation (*mutationRate*), and the tournament size i.e. the tournament rate (*tour-

namentRate). In this experiment, the *mutationRate* is set to 0.02 and the *tournamentRate* is set to 0.1 after some initial tuning. Table 7.14 shows the results with the algorithm with different population sizes and different number of generations for the two most challenging problem instances LB8-1 and LB8-2. Each instance was run five times and the average result of both instances is given in column *Avg.*. The *maxIteration* is set to 5 as increasing it further gives very little further improvement. With the consideration of algorithm efficiency, we choose the combination of *populationSize*=500 and *generations*=500.

Table 7.14: Experiment results for evaluating *populationSize* and *generations*

populationSize	*generations*	*Avg.*
10	10	25043
100	10	22154
200	10	21797
500	10	21280
10	100	21930
100	100	20991
200	100	20588
500	100	20079
10	200	21647
100	200	20676
200	200	20128
500	200	19999
10	500	21205
100	500	20228
200	500	20081
500	500	19879
1000	1000	19775
2000	2000	19724

7.3.2 Comparing VNS and GA based branch and pricing approaches

Due to the very large amount of computational time required, we select a total of six instances, two from the real-life instance set and four from the artificial instance set used by Chapter 6. The instance names starting with **NP** are real-life instances while those starting with **LB**, **TB**, **LU** and **TU** represent (**Loose, Balanced**), (**Tight, Balanced**), (**Loose, Unbalanced**) and (**Tight, Unbalanced**) configurations of artificial instances. The first digit of each instance name indicates the length of the planning horizon (e.g. NP4-1 is a real-life instance with a 4-shift planning horizon).

The stopping criterion *maxIteration* is set to 10 for both VNS and GA. Figure 7.6, 7.7, 7.8, 7.9, 7.10, and 7.11 list the mean objective values of 30 runs of experiments. Horizontal axis defines the number of RMP iterations while the vertical axis given the objective values. Additionally, the confidence intervals of the above results are applied to perform a two-tailed t-test and it can be seen from Table 7.15 that all P-values are less than 0.05. The results suggest that GA (based column generator) is a faster converging and more robust algorithm thanks to its population based search framework and capacity to evolve a set of routes instead of a single one.

As experiments show that the VNS converges in 30 RMP iterations, for a fairer comparison, we set *maxIteration* to 30 for both GA and VNS for all instances. In addition, a comparison was also made with previous results obtained by the hybrid method, metaheuristic methods and lower bound

Table 7.15: T-test results of confidence intervals at RMP=10 (Mean of 30 runs)

Instance	P-value
TU8-7	0.0031
TB4-4	0.0011
NP8-1	0.0069
NP4-1	0.0405
LU4-6	0.0379
LB8-1	0.0004

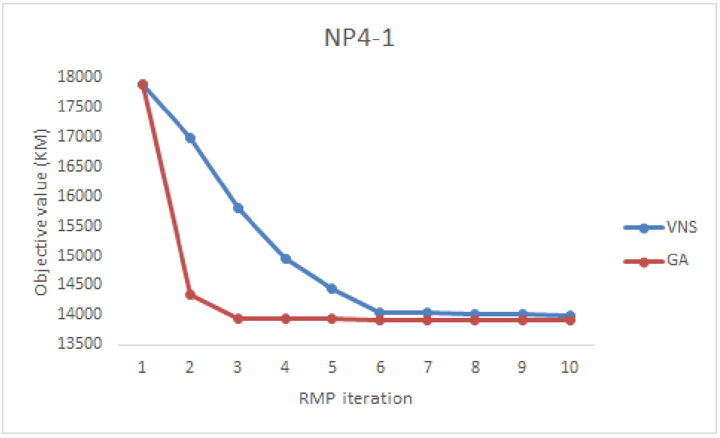

Figure 7.6: Comparison of GA and VNS (NP4-1)

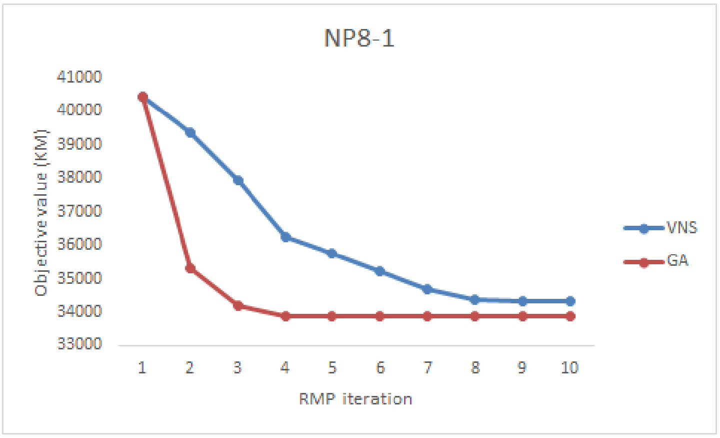

Figure 7.7: Comparison of GA and VNS (NP8-1)

reported in Chapter 6.

Table 7.16 presents the running time and objective values obtained by the branch-and-price algorithm using VNS and GA column generators (**BP-VNS**, **BP-GA**), **Hybrid** method (3-stage hybrid method proposed in

Table 7.16: Comparisons with previous results

	BP-VNS		*BP-GA*		*Hybrid*		*VNS*		*SAHH*	
Instance	T	Obj.	T	Obj.	T	Obj.	T	Obj.	T	Obj.
NP4-1	126	13978	405	13860	33301	**13509**	4800	14453	310	14471
NP4-2	112	16667	462	**16621**	15742	16636	4800	16593	386	16595
NP4-3	132	17110	417	17106	11178	**16879**	4800	17138	590	17383
NP4-4	273	22100	509	**21980**	18537	21886	4800	22302	545	22142
NP4-5	384	26184	1195	**26166**	20647	26731	4800	26216	1022	26239
NP6-1	1017	34054	3742	**34022**	160079	34055	7200	35209	1613	35122
NP6-2	1360	33490	1868	33490	138486	**33316**	7200	33808	955	33653
NP6-3	45	16150	198	**16094**	3978	16192	7200	16660	211	16247
NP6-4	356	26146	1262	**26126**	58898	26260	7200	26272	698	26316
NP6-5	545	16883	984	**16817**	104446	16881	7200	17950	492	17800
NP8-1	730	33889	1133	**33789**	148067	35685	9600	34181	822	34095
NP8-2	825	30576	1612	**30554**	147241	30633	9600	31639	869	31310
NP8-3	1049	28281	1260	**28281**	121074	28314	9600	28450	878	28451
NP8-4	1211	43643	1731	**43630**	66438	44224	9600	43955	1631	43943
NP8-5	898	25419	1415	**25389**	131369	25452	9600	25742	1128	26182
LB4-1	89	15852	447	15766	13438	**15763**	4800	16011	292	15865
LB4-2	61	14975	283	14777	3812	**14319**	4800	15291	414	15059
TB4-3	30	11027	128	**10364**	1415	10867	4800	11027	288	11092
TB4-4	21	12671	157	**12172**	186	12508	4800	13577	383	13495
LU4-5	66	18242	183	**17676**	1590	18500	4800	19884	276	19717
LU4-6	65	19403	215	**19394**	1783	20316	4800	19741	233	19859
TU4-7	6	12869	113	**12804**	79	13033	4800	13760	232	14377
TU4-8	12	18920	125	17956	138	**17025**	4800	17846	491	17815
LB8-1	375	18251	1803	**18097**	138988	18133	9600	18542	1899	18325
LB8-2	444	22265	2909	**20928**	157354	22834	9600	23068	1266	22990
TB8-3	73	21670	224	**20456**	148	21338	9600	21657	2602	21689
TB8-4	112	28001	193	**25316**	561	28167	9600	28398	2391	28305
LU8-5	73	23288	248	22453	4380	**21226**	9600	24587	915	24787
LU8-6	226	23528	659	**22690**	13202	23261	9600	24412	1204	24261
TU8-7	58	32680	166	32334	140	**31094**	9600	35595	484	35581
TU8-8	58	27884	197	**26958**	66	27406	9600	28197	434	28162
Large	8355	105793	7801	**100119**	n.a.	n.a.	9600	142258	15848	141252
Average	349	22777	847	**22389**	48928	22659	7200	23295	837	23269

BP-VNS:branch-and-price with VNS based column generator;
BP-GA:branch-and-price with GA based column generator;
Hybrid:the 3-stage hybrid method;
VNS:variable neighbourhood search metaheuristic;
SAHH:simulated annealing hyperheuristic;
Col.:Total columns generated;
Obj.:Objective Value(km);
T:Total running time(s);
n.a.:The algorithm fails to solve the problem within 48h;
Average: did not account in results of **Large** instance.
Result ranking(best to worst):BP-GA>Hybrid>BP-VNS>SAHH>VNS
Runtime ranking(fastest to slowest):BP-VNS>SAHH>BP-GA>VNS>Hybrid

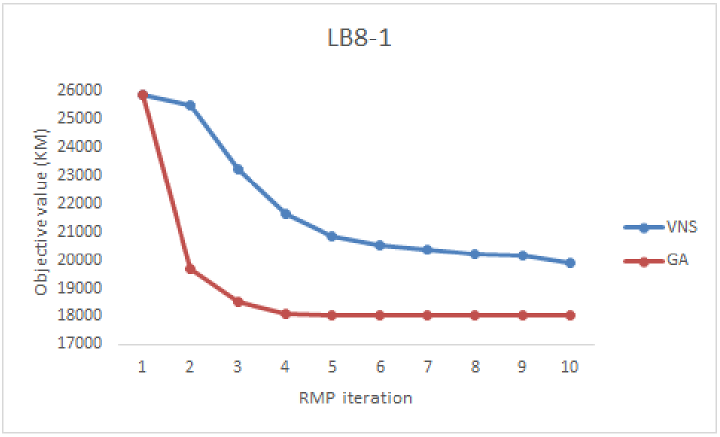

Figure 7.8: Comparison of GA and VNS (LB8-1)

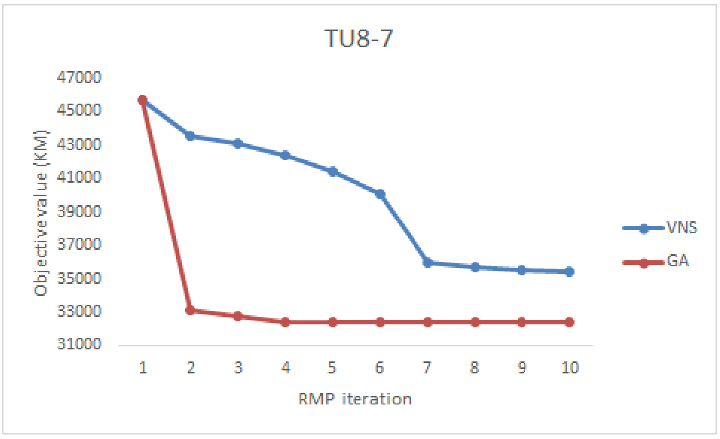

Figure 7.9: Comparison of GA and VNS (TU8-7)

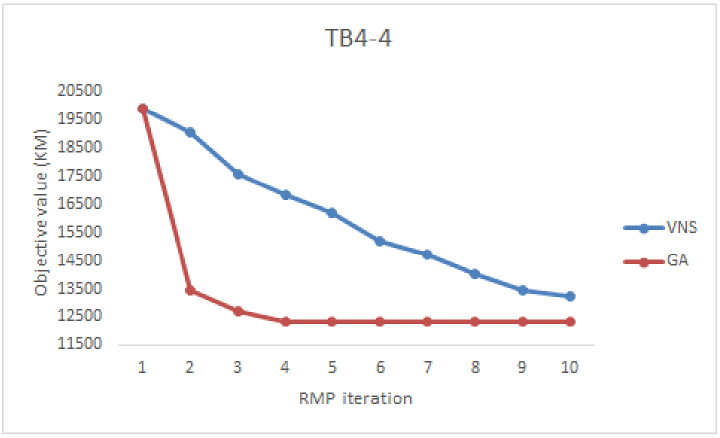

Figure 7.10: Comparison of GA and VNS (TB4-4)

Chapter 6), metaheuristic methods using simulated annealing hyperheuristic (**SAHH**), and reactive shaking variable neighbourhood search (**VNS**).

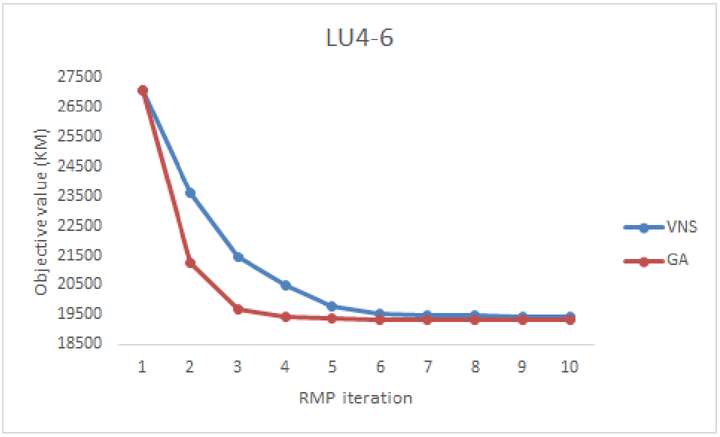

Figure 7.11: Comparison of GA and VNS (LU4-6)

In terms of objective values, the average results in increasing order are as follows: BP-GA < Hybrid < BP-VNS < SAHH < VNS. The best results are highlighted in bold. The average running times show an increasing order as follows: BP-VNS < SAHH < BP-GA < VNS < Hybrid. These tests suggest that both BP-GA and BP-VNS are able to find better solution in less time than most of the existing algorithms.

The solution coding scheme and pricing methods limit the search space for the algorithm, so its efficiency is increased compared with the results obtained by metaheuristics (VNS and SAHH). The Hybrid method performs well for tight instances, but it does less well for large and loose instances. The reason is that the Hybrid method employs an integer programming solver so its solution time increases exponentially with large problem sizes. The proposed branch-and-price methods are able to find effective columns in order to reduce the problem size, therefore, compared with the Hybrid method, the solving time of branch-and-price is significantly decreased for large instances. However, the advantage of branch-and-price may not be obvious for small problem instances (i.e. tight and small instances) as the iterative RMP solving comprises a significant proportion of the run time of the algorithm. In addition to the current experiment settings, there are many options can be considered, for example, the diversity maintenance (e.g. techniques for keeping members of the GA population diverse), it would be good to incorporate diversity maintenance in our GA solution in future.

7.4 Summary

We have shown an innovative branch-and-price approach and metaheuristic pricing that succeeds in finding solutions for a multi-shift full truckload vehicle routing problem on both real-life and artificial instances. The result indicates that the proposed solution methods improves the current solutions both in terms of the computational time and the solution quality. We believe the present findings have the potential implications for efficiently solving real-life drayage container operation problems with long planning horizon covering multi-shifts.

Chapter 8

Conclusions and Future Work

The container transportation industry is under fierce competition and pressure to improve its efficiency and reduce energy use and increasingly more studies have been devoted to the optimisation of operations at container terminals. In this thesis, we study a multi-shift full truckload vehicle routing problem (MFTLVRP), using data from a real-life problem faced by port of Ningbo. The problem is also common for any large port with multiple docks being operated simultaneously. The main work in this thesis is summarised below:

8.1 Summary of Work

The main work in this thesis is in three main areas:

8.1.1 From modelling perspective

As stated in the Section Background and Motivation 1.1, the overall aim of this research is to investigate novel approaches that can be used to solve

MFTLVRPs. This thesis illustrates several related models and discusses related potential issues. Representing each task/customer/load as a node in graph is a common practice for VRP-style formulations. However, the increasing number of tasks usually results in a huge graph. To address this issue, a set covering model is developed based on a novel route representation and implicit solution encoding. It has been shown that one of the helpful benefits of this encoding is the transformation of a previous VRP-based non-linear model into a linear/integer model, so it can be solved using various integer programming techniques. Further study of this model has shown the size of the search space can be further reduced by heuristically decomposing the problem into a master problem and a subproblem which can be efficiently solved by metaheuristics.

8.1.2 From hybridising exact and metaheuristic methods perspective

MFTLVRP is closely related to VRPPD, which is NP-hard. There is no known polynomial time bound algorithm that can guarantee an optimal solution. Nowadays, an increasing number of hybridisations of metaheuristics and exact approaches are being developed. These methods usually provide good results as they take advantage of the strengths of both methods. In this thesis, we focus on hybridisation of branch-and-price and metaheuristic approaches.

Branch-and-price (B&P) is an effective integer programming method for problems with larger numbers of columns, most of which are non-basic in the optimal solutions (that means only very small parts of the columns contribute to optimal solutions). It is a potentially very good method for the set covering formulation proposed in this thesis, where the feasible route set is very large, leading to a model with a huge number of columns, while the optimal solution is a very small subset of it.

We propose to use metaheuristics for the column generation subproblem of branch-and-price. The reasons are twofold: Firstly, the reduced cost of the routes cannot be directly computed from the dual problem of the set covering model because of the implicit solution encoding. Secondly, even if the accurate cost information can be obtained, the subproblem to compute the optimal column to add in the restricted master problem is also NP-Hard. The goal of the metaheuristics is to identify new columns with negative reduced costs. We propose a variable neighbourhood search (VNS) and a genetic algorithm (GA) column generator, as they are widely adopted frameworks to implement these ideas. Particularly it is hoped that a GA is suitable to tackle the pricing subproblem. There are two reasons: Firstly, at each branch-and-price iteration, we need to obtain multiple routes with the most negative reduced costs. A GA naturally can help find a population of routes through evolution. Secondly, we believe that high quality routes may share some common structures which could be evolved more efficiently through crossover operations in a GA framework.

The experimental results of both real-life and artificial instances show that the hybridisation of branch-and-price and metaheuristics improves performance compared with previous results solved by 3-stage hybrid method, hyperheuristic methods using simulated annealing hyperheuristic (SAHH), and reactive shaking variable neighbourhood search (VNS).

8.1.3 From traffic forecasting perspective

Accurate travel time prediction is useful for logistics companies managing freight transportation, as unreliable travel time is viewed as the most problematic challenge faced in freight operations [83]. Our study of truck travel time prediction in a port drayage network has two major motivations: Firstly, as stated, the set covering model relies on travel times between docks to generate a feasible route set. The method proposed in this study is an approach to estimate travel time parameters for a container-based logistics planning system. By analysing real-life GPS data obtained from the container truck fleet, we found that increasing travel time patterns appear in peak time. This motivated us to investigate further to estimate travel times more accurately and efficiently. Secondly, most studies focus on developing traffic forecasting techniques for urban roads and freeways. However, studies on the prediction of real-world container truck travel times are somewhat limited. This study attempts to fill this research gap. The outcome of this study is a reliable autoregressive integrated moving-average (ARIMA) model based on real-life GPS data obtained from a truck fleet that transports containers for the Port of Ningbo.

8.2 Summary of Each Chapter

8.2.1 Chapter 4

This chapter described the MFTLVRP and presents a set covering integer linear programming model for it. A lower bound of the problem is also obtained by relaxing the time window constraints to the nearest shifts and transforming the problem into a service network design problem. The underlining features of the set covering model are analysed and compared with other node based formulations.

8.2.2 Chapter 5

This chapter demonstrated a study of automated container truck travel time prediction based on real-life GPS data using ARIMA. We also implemented a back-propagation NN model and a SVM with Gaussian kernel model and compared their forecasting performance with ARIMA. The results indicate that for the traffic data under drayage operation scenario, ARIMA model appears to perform best, indicating that ARIMA model together with the data preparation method discussed in this chapter is a more reliable approach to estimate travel time parameters for the container based logistics planning system. The outcomes of this study are used to generate feasible routes for the model presented in Chapter 4.

8.2.3 Chapter 6

This chapter presented a 3-stage hybrid solution method for the model presented in Chapter 4. The method is also tested on a set of real-life and artificial instances. It was found that for real-life instances, the solution obtained from the set covering model is very close to the lower bound, suggesting that the time window may not be the driving factor for the low transport efficiency but the demand imbalance between different ports is. In addition, it was shown that the proposed model and method is able to find solutions that are very close to the lower bounds.

8.2.4 Chapter 7

This chapter has shown an innovative branch-and-price approach succeeds in solving a MFTLVRP on both real-life and artificial instances. The result indicates that hybridisation of both branch-and-price and metaheuristic improve the current solutions both in terms of the computational time and the solution quality.

8.3 Future work

A number of possible avenues of research can be pursued to further the work presented in this thesis:

8.3.1 Improve solving efficiency

Research on hybridisation of exact and metaheuristics is still in its early days. In addition to the branch-and-price presented in this thesis, other integer optimisation techniques can be investigated and incorporated to address this problem. In order to improve the column generation process, the column generator can be further developed by using other metaheuristics or introducing other neighbourhood functions. In practice, multi-threading parallel computing techniques can be used to speed up the solution process.

8.3.2 Handle dynamic situation

The proposed hybrid framework can also be applied to other (real-world) drayage problems. As this study focuses on a static problem, it would be of interest to extend our model to a dynamic problem where some tasks may be modified in the middle of a shift. In real-word applications, there may exist some unknown factors that can affect this problem, such as non-normal travelling time of tasks, and loading and unloading times. Current solutions include re-optimising the schedule. In addition, addressing these issues may also require research into new stochastic models [168] and solution methods integrating real-time GPS information of the fleet, all of which are deserving of future research (e.g. [158], [192] and [201]).

8.3.3 Improve travel time prediction

Future work should also extend beyond the limited scope of this study, which only investigates three predictors. There are more factors that influence this prediction model, in particular, the capacity utilisation of ports. The models should also be further validated with more and larger data instances. We also suggest that the multi-GNSS (Global Navigation Satellite System) technique be adopted by fleets, as it is more accurate than the single GPS, so that more precise travel times can be obtained.

8.3.4 Relocation of depot

It is shown that the proposed model and solution methods are able to find solutions that are very close to the lower bounds. In order to further improve transportation efficiency, the problem needs to be solved at higher level-tactical network planning where load imbalance has to be addressed. Network planning models can be proposed to determine the best location for the depot (or a set of depots), where the containers can be stored and directly distributed to customers. These models should take into account the welfare of all (e.g. [27], [26], [6], and [118]) involved at the Port of Ningbo-Zhoushan.

Bibliography

[1] The VRP web. http://www.bernabe.dorronsoro.es/vrp/, 2006. [Online; accessed 19-July-2015]. [cited at p. 41]

[2] Container ports of China. http://www.portcontainer.cn/, 2016. [Online; accessed 15-June-2016]. [cited at p. 1, 3]

[3] The largest container ports worldwide in 2014, based on throughput (in million TEUs). http://www.statista.com/statistics/264171/turnover-volume-of-the-largest-container-ports-worldwide/, 2016. [Online; accessed 9-May-2016]. [cited at p. 1]

[4] Mohamed S. Ahmed and Allen R. Cook. Analysis of freeway traffic time-series data by using box-jenkins techniques. *Transportation Research Record*, (722):1–9, 1979. [cited at p. 100, 113]

[5] Enrique Alba, Francisco Almeida, M Blesa, J Cabeza, Carlos Cotta, Manuel Díaz, Isabel Dorta, Joaquim Gabarró, Coromoto León, J Luna, et al. Mallba: A library of skeletons for combinatorial optimisation. In *Euro-Par 2002 Parallel Processing*, pages 927–932. Springer, 2002. [cited at p. 34]

[6] Ghazal Assadipour, Ginger Y Ke, and Manish Verma. Planning and managing intermodal transportation of hazardous materials with capacity s-

election and congestion. *Transportation Research Part E: Logistics and Transportation Review*, 76:45–57, 2015. [cited at p. 192]

[7] Ruibin Bai. *An investigation of novel approaches for optimising retail shelf space allocation.* PhD thesis, University of Nottingham, 2005. [cited at p. 11]

[8] Ruibin Bai, Graham Kendall, Rong Qu, and Jason A.D. Atkin. Tabu assisted guided local search approaches for freight service network design. *Information Sciences*, 189:266 – 281, 2012. [cited at p. 64, 67]

[9] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9):1229–1246, 1996. [cited at p. 18]

[10] Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone. *Genetic programming: an introduction.* Morgan Kaufmann Publishers San Francisco, 1998. [cited at p. 30]

[11] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998. [cited at p. 18, 19]

[12] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952. [cited at p. 15]

[13] Richard E Bellman and Stuart E Dreyfus. *Applied dynamic programming.* Princeton university press, 2015. [cited at p. 16]

[14] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Rejoinder on: Static pickup and delivery problems: a classification scheme and survey". *TOP*, 15(1):45–47, 2007. [cited at p. 67]

[15] Christian Blum and Xiaodong Li. Swarm intelligence in optimization. In *Swarm Intelligence*, pages 43–85. Springer, 2008. [cited at p. 31]

[16] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011. [cited at p. 35]

[17] Natashia Boland, John Dethridge, and Irina Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58–68, 2006. [cited at p. 20, 21]

[18] Kris Braekers, An Caris, and Gerrit K. Janssens. A deterministic annealing algorithm for a bi-objective full truckload vehicle routing problem in drayage operations. *Procedia - Social and Behavioral Sciences*, 20(0):344 – 353, 2011. [cited at p. 75]

[19] Kris Braekers, An Caris, and Gerrit K Janssens. Time-dependent routing of drayage operations in the service area of intermodal terminals. 2012. [cited at p. 71]

[20] Kris Braekers, An Caris, and Gerrit K Janssens. Integrated planning of loaded and empty container movements. *OR spectrum*, 35(2):457–478, 2013. [cited at p. 62]

[21] Kris Braekers, An Caris, and Gerrit K Janssens. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple de-

pots. *Transportation Research Part B: Methodological*, 67:166–186, 2014. [cited at p. 69, 71, 75]

[22] Kris Braekers, An Caris, and GerritK. Janssens. Integrated planning of loaded and empty container movements. *OR Spectrum*, 35(2):457–478, 2013. [cited at p. 71, 72, 75]

[23] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 2015. [cited at p. 40]

[24] Daniel T. Brunner, Gary Cross, Catherine McGhee, Jack Levis, and Dudley E. Whitney. Toward increased use of simulation in transportation. In *Winter Simulation Conference*, pages 1169–1176, 1998. [cited at p. 99]

[25] Edmund K Burke, Steven Gustafson, and Graham Kendall. Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62, 2004. [cited at p. 30]

[26] Claudia Caballini, Simona Sacone, and Mahnam Saeednia. Planning truck carriers operations in a cooperative environment. *IFAC Proceedings Volumes*, 47(3):5121–5126, 2014. [cited at p. 192]

[27] Claudia Caballini, Simona Sacone, and Mahnam Saeednia. Cooperation among truck carriers in seaport containerized transportation. *Transportation Research Part E: Logistics and Transportation Review*, 93:38–56, 2016. [cited at p. 192]

[28] An Caris and Gerrit K Janssens. A local search heuristic for the pre-and end-haulage of intermodal container terminals. *Computers & Operations Research*, 36(10):2763–2772, 2009. [cited at p. 53, 54, 74, 81]

[29] An Caris and Gerrit K. Janssens. A deterministic annealing algorithm for the pre- and end-haulage of intermodal container terminals. *IJCAET*, 2(4):340–355, 2010. [cited at p. 74]

[30] Jianjun Chen. *An Intelligent Container Transportation System Using Novel Modelling and Metaheuristics*. PhD thesis, University of Nottingham, 2016. [cited at p. 59, 73, 89, 90, 92, 131, 140, 167]

[31] Jianjun Chen, Ruibin Bai, Rong Qu, and Graham Kendall. A task based approach for a real-world commodity routing problem. In *Computational Intelligence In Production And Logistics Systems (CIPLS), 2013 IEEE Workshop on*, pages 1–8. IEEE, 2013. [cited at p. x, 131, 134, 135, 140, 147, 157]

[32] Qingfeng Chen, Kunpeng Li, and Zhixue Liu. Model and algorithm for an unpaired pickup and delivery vehicle routing problem with split loads. *Transportation Research Part E: Logistics and Transportation Review*, 69:218 – 235, 2014. [cited at p. 68]

[33] SY Chen, SN Talukdar, and NM Sadeh. Job-shop-scheduling by a team of asynchronous agents. In *IJCAI-93 Workshop on Knowledge-Based Production, Scheduling and Control*, 1993. [cited at p. 34]

[34] Raymond K Cheung, Ning Shi, Warren B Powell, and Hugo P Simao. An attribute–decision model for cross-border drayage problem. *Transportation Research Part E: Logistics and Transportation Review*, 44(2):217–234, 2008. [cited at p. 73]

[35] Marielle Christiansen and Kjetil Fagerholt. Robust ship scheduling with multiple time windows. *Naval Research Logistics (NRL)*, 49(6):611–625, 2002. [cited at p. 77]

[36] N. Christofides and J.E. Beasley. Period routing problem. *Networks*, 14(2):237–256, 1984. [cited at p. 76]

[37] Paul C Chu and John E Beasley. A genetic algorithm for the multi-dimensional knapsack problem. *Journal of heuristics*, 4(1):63–86, 1998. [cited at p. 34]

[38] Ki Ho Chung, Chang Seong Ko, Jae Young Shin, Hark Hwang, and Kap H-wan Kim. Development of mathematical models for the container road transportation in Korean trucking industries. *Computers & Industrial Engineering*, 53(2):252–262, 2007. [cited at p. 73]

[39] D Clements, J Crawford, D Joslin, G Nemhauser, M Puttlitz, and M Savelsbergh. Heuristic optimization: A hybrid AI/OR approach. In *Workshop on Industrial Constraint-Directed Scheduling*, 1997. [cited at p. 33]

[40] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105–119, 1997. [cited at p. 58, 77]

[41] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. Vehicle routing. *Handbooks in operations research and management science*, 14:367–428, 2007. [cited at p. 40]

[42] Luca Coslovich, Raffaele Pesenti, and Walter Ukovich. Minimizing fleet operating costs for a container transportation company. *European Journal of Operational Research*, 171(3):776 – 786, 2006. [cited at p. 74]

[43] David Coufal and Esko Turunen. Short term prediction of highway travel time using data mining and neuro-fuzzy methods. In *Proc. IFSA*, pages 175–182, 2003. [cited at p. 100]

[44] Teodor G Crainic and Jean-Marc Rousseau. Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem. *Transportation Research Part B: Methodological*, 20(3):225–242, 1986. [cited at p. 62]

[45] Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000. [cited at p. 1, 55, 57, 62]

[46] Teodor Gabriel Crainic. Long-haul freight transportation. In *Handbook of transportation science*, pages 451–516. Springer, 2003. [cited at p. 62]

[47] Teodor Gabriel Crainic and Gilbert Laporte. Planning models for freight transportation. *European journal of operational research*, 97(3):409–438, 1997. [cited at p. 62]

[48] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983. [cited at p. 18]

[49] George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954. [cited at p. 42]

[50] George B Dantzig. Application of the simplex method to a transportation problem. *Activity analysis of production and allocation*, 13:359–373, 1951. [cited at p. 13]

[51] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959. [cited at p. 40]

[52] Jörg Denzinger and Tim Offermann. On cooperation between evolutionary algorithms and other search paradigms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999. [cited at p. 34]

[53] U. Derigs, M. Pullmann, U. Vogel, M. Oberscheider, M. Gronalt, and P. Hirsch. Multilevel neighborhood search for solving full truckload routing problems arising in timber transportation. *Electronic Notes in Discrete Mathematics*, 39:281–288, 2012. [cited at p. 67]

[54] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):pp. 342–354, 1992. [cited at p. 20]

[55] Martin Desrochers and Québec. Groupe d'études et de recherche en analyse des décisions École des hautes études commerciales Montréal. *An algorithm for the shortest path problem with resource constraints.* Montréal: École des hautes études commerciales, 1988. [cited at p. 20, 52]

[56] Jacques Desrosiers and Marco E Lübbecke. Branch-price-and-cut algorithms. *Wiley encyclopedia of operations research and management science*, 2011. [cited at p. 19]

[57] Marco Dorigo. *Optimization, Learning and Natural Algorithms (in Italian).* PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992. [cited at p. 31]

[58] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006. [cited at p. 31]

[59] Marco Dorigo and Thomas Stützle. Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics*, pages 227–263. Springer, 2010. [cited at p. 31]

[60] Moshe Dror. Note on the complexity of the shortest path models for column generation in vrptw. *Operations Research*, 42(5):977–978, 1994. [cited at p. 20]

[61] Irina Dumitrescu and Thomas Stützle. Combinations of local search and exact algorithms. In *Applications of Evolutionary Computing*, pages 211–223. Springer, 2003. [cited at p. 32]

[62] R.W. Eglese. Routeing winter gritting vehicles. *Discrete Applied Mathematics*, 48(3):231 – 244, 1994. [cited at p. 76]

[63] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472 – 1483, 2009. [cited at p. 40]

[64] Andries P Engelbrecht. *Fundamentals of computational swarm intelligence.* John Wiley & Sons, 2006. [cited at p. 31]

[65] X. Fei, C.-C. Lu, and K. Liu. A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction. *Transportation Research Part C: Emerging Technologies*, 19(6):1306–1318, 2011. [cited at p. 111]

[66] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *A Quarterly Journal of Operations Research*, 8:407–424, 2010. [cited at p. 150]

[67] Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. [cited at p. 20]

[68] Francesco Ferrucci and Stefan Bock. Real-time control of express pickup and delivery processes in a dynamic environment. *Transportation Research Part B: Methodological*, 63:1 – 14, 2014. [cited at p. 69]

[69] Marshall L Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 50(12):1861–1871, 2004. [cited at p. 17]

[70] Alex S Fraser. Simulation of genetic systems by automatic digital computers vi. epistasis. *Australian Journal of Biological Sciences*, 13(2):150–162, 1960. [cited at p. 27]

[71] P.W. Frizzell and J.W. Giffin. The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers & Operations Research*, 22(6):655 – 667, 1995. [cited at p. 76]

[72] Peter G Furth, Brendon Hemily, Theo H. J. Muller, and James G Strathman. *Using Archived AVL-APC Data to Improve Transit Performance and Management.* 2006. [cited at p. 95]

[73] Peter G. Furth and Theo H. J. Muller. Service reliability and optimal running time schedules. *Transportation Research Record Journal of the Transportation Research Board*, 2034(2034):55–61, 2007. [cited at p. 95]

[74] Yuee Gao, Yaping Zhang, Hejiang Li, Ting Peng, and Siqi Hao. Study on the relationship between comprehensive transportation freight index and GDP in china. *Procedia Engineering*, 137:571–580, 2016. [cited at p. 1]

[75] Michael R Garey and David S Johnson. A guide to the theory of np-completeness. *WH Freemann, New York*, 1979. [cited at p. 11]

[76] Michel Gendreau and Jean-Yves Potvin. Tabu search. In EdmundK. Burke and Graham Kendall, editors, *Search Methodologies*, pages 165–186. Springer US, 2005. [cited at p. 25]

[77] Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977. [cited at p. 24]

[78] Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989. [cited at p. 24]

[79] Fred Glover. Tabu search-part ii. *ORSA Journal on computing*, 2(1):4–32, 1990. [cited at p. 24]

[80] Fred Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Springer Science & Business Media, 2003. [cited at p. 23]

[81] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988. [cited at p. 30]

[82] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008. [cited at p. 77]

[83] Thomas F Golob and Amelia C Regan. Freight industry attitudes towards policies to reduce congestion. *Transportation Research Part E: Logistics and Transportation Review*, 36(1):55–77, 2000. [cited at p. 97, 188]

[84] Ralph E Gomory. An algorithm for integer solutions to linear programs. *Recent advances in mathematical programming*, 64:260–302, 1963. [cited at p. 18]

[85] Manfred Gronalt, Richard F. Hartl, and Marc Reimann. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research*, 151(3):520 – 535, 2003. [cited at p. 72]

[86] Tioga Group et al. *Truck Drayage Productivity Guide.* Number 11. Transportation Research Board, 2011. [cited at p. 2]

[87] Deepti Gupta and Shabina Ghafir. An overview of methods maintaining diversity in genetic algorithms. *International journal of emerging technology and advanced engineering*, 2(5):56–60, 2012. [cited at p. 30]

[88] Gabriel Gutiérrez-Jarpa, Guy Desaulniers, Gilbert Laporte, and Vladimir Marianov. A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, 206(2):341–349, 2010. [cited at p. 67]

[89] Pierre Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Congress on numerical methods in combinatorial optimization, Capri, Italy*, pages 70–145, 1986. [cited at p. 24]

[90] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001. [cited at p. 26]

[91] Pierre Hansen, Nenad Mladenovic, and José A. Moreno-Pérez. Variable neighborhood search. *European Journal of Operational Research*, 191(3):593–595, 2008. [cited at p. 27]

[92] Pierre Hansen, Nenad Mladenovic, and Dionisio Pérez-Brito. Variable neighborhood decomposition search. *J. Heuristics*, 7(4):335–350, 2001. [cited at p. 26]

[93] Pierre Hansen, Nenad Mladenovic, and Dragan Urosevic. Variable neighborhood search and local branching. *Computers & OR*, 33(10):3034–3045, 2006. [cited at p. 26]

[94] Nguyen Thi Hien and Nguyen Xuan Hoai. A brief overview of population diversity measures in genetic programming. In *Proc. 3rd Asian-Pacific Workshop on Genetic Programming, Hanoi, Vietnam*, pages 128–139. Citeseer, 2006. [cited at p. 30]

[95] Frederick S Hillier. *Introduction to operations research.* Tata McGraw-Hill Education, 1995. [cited at p. 13, 17, 22, 30]

[96] Petros Ioannou Anastasios Chassiakos Hossein Jula, Maged Dessouky. Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review*, 41(3):235 – 259, 2005. [cited at p. 72]

[97] Ta-Yin Hu and Wei-Ming Ho. Simulation-based travel time prediction model for traffic corridors. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–6. IEEE, 2009. [cited at p. 101]

[98] Shan-Huen Huang and Pei-Chun Lin. Vehicle routing scheduling for municipal waste collection system under the keep trash off the ground policy. *Omega*, 55:24 – 37, 2015. [cited at p. 78]

[99] Yetkin Ileri, Mokhtar Bazaraa, Ted Gifford, George Nemhauser, Joel Sokol, and Erick Wikum. An optimization approach for planning daily drayage operations. *Central European Journal of Operations Research*, 14(2):141–156, 2006. [cited at p. 74]

[100] Akio Imai, Etsuko Nishimura, and John Current. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European Journal of Operational Research*, 176(1):87 – 105, 2007. [cited at p. 74, 75]

[101] Laetitia Jourdan, Matthieu Basseur, and E-G Talbi. Hybridizing exact methods and metaheuristics: A taxonomy. *European Journal of Operational Research*, 199(3):620–629, 2009. [cited at p. 35]

[102] Hossein Jula, Maged Dessouky, and Petros A Ioannou. Real-time estimation of travel times along the arcs and arrival times at the nodes of dynamic stochastic networks. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):97–110, 2008. [cited at p. 100]

[103] Brian Kallehauge, Jesper Larsen, OliB.G. Madsen, and MariusM. Solomon. Vehicle routing problem with time windows. In Guy Desaulniers, Jacques Desrosiers, and MariusM. Solomon, editors, *Column Generation*, pages 67–98. Springer US, 2005. [cited at p. 20, 21, 149, 150]

[104] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007. [cited at p. 31, 32]

[105] M. G. Karlaftis and E. I. Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C-emerging Technologies*, 19:387–399, 2011. [cited at p. 100, 117]

[106] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011. [cited at p. 31, 32]

[107] Mubassira Khan. *Application of choice modeling methods to describe commercial vehicle travel behavior in urban areas.* PhD thesis, 2015. [cited at p. 98]

[108] Gitae Kim, Yew-Soon Ong, Chen Kim Heng, Puay Siew Tan, and Nengsheng Allan Zhang. City vehicle routing problem (city vrp): a review. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1654–1666, 2015. [cited at p. 40]

[109] Dominik Kirchler and Roberto Wolfler Calvo. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological*, 56:120 – 135, 2013. [cited at p. 68]

[110] Gunnar W Klau, Ivana Ljubić, Andreas Moser, Petra Mutzel, Philipp Neuner, Ulrich Pferschy, Günther Raidl, and René Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. In *Genetic and evolutionary computation–gecco 2004*, pages 1304–1315. Springer, 2004. [cited at p. 33]

[111] Niklas Kohl. Exact methods for time constrained routing and related scheduling problems. 1995. [cited at p. 21]

[112] Masakazu Kojima, Shinji Mizuno, and Akiko Yoshise. *A primal-dual interior point algorithm for linear programming*. Springer, 1989. [cited at p. 15]

[113] KN Krishnanand and Debasish Ghose. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm intelligence*, 3(2):87–124, 2009. [cited at p. 32]

[114] Manoj Kumar, Mohammad Husian, Naveen Upreti, and Deepti Gupta. Genetic algorithm: Review and application. *International Journal of Information Technology and Knowledge Management*, 2(2):451–454, 2010. [cited at p. 30]

[115] Nadia Lahrichi, Teodor Gabriel Crainic, Michel Gendreau, Walter Rei, Cerasela Crisan, and Thibaut Vidal. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. Technical report, Technical Report, CIRRELT, 2012. [cited at p. 34]

[116] Rahma Lahyani, Mahdi Khemakhem, and Frédéric Semet. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14, 2015. [cited at p. 40]

[117] Michela Lai, Teodor Gabriel Crainic, Massimo Di Francesco, and Paola Zuddas. An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transportation Research Part E: Logistics and Transportation Review*, 56:108–118, 2013. [cited at p. 73]

[118] Jasmine Siu Lee Lam and Yimiao Gu. A market-oriented approach for intermodal network optimisation meeting cost, time and environmental re-

quirements. *International Journal of Production Economics*, 171:266–274, 2016. [cited at p. 192]

[119] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960. [cited at p. 17]

[120] G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009. [cited at p. 40]

[121] G. Laporte and I.H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262, 1995. [cited at p. 40]

[122] Jong-Seok Lee, Cheol Hoon Park, and Touradj Ebrahimi. Theory and applications of hybrid simulated annealing. In *Handbook of Optimization - From Classical to Modern Approach*, pages 395–422. 2013. [cited at p. 26]

[123] Jan Karel Lenstra and AHG Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. [cited at p. 12]

[124] A.N. Letchford and R.W. Eglese. The rural postman problem with deadline classes. *European Journal of Operational Research*, 105(3):390 – 400, 1998. [cited at p. 76]

[125] Chi-Sen Li and Mu-Chen Chen. A data mining based approach for travel time prediction in freeway with non-recurrent congestion. *Neurocomputing*, 133:74–83, 2014. [cited at p. 106, 111]

[126] Liang Liu, Clio Andris, and Carlo Ratti. *Computers, Environment and Urban Systems*, 34(6):541–548, 2010. [cited at p. 64]

[127] Ran Liu, Xiaolan Xie, Vincent Augusto, and Carlos Rodriguez. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, 230(3):475 – 486, 2013. [cited at p. 67]

[128] David G Luenberger. *Introduction to linear and nonlinear programming*, volume 28. Addison-Wesley Reading, MA, 1973. [cited at p. 15]

[129] Simona Mancini. A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research Part C: Emerging Technologies*, 2015. [cited at p. 78]

[130] Anna Marino, Adam Prügel-Bennett, and Celia A Glass. Improving graph colouring with linear programming and genetic algorithms. In *In*. Citeseer, 1999. [cited at p. 34]

[131] Nikola Marković, Željko Drobnjak, and Paul Schonfeld. Dispatching trucks for drayage operations. *Transportation Research Part E: Logistics and Transportation Review*, 70:99–111, 2014. [cited at p. 2, 62]

[132] Gideon Mbiydzenyuy, Mattias Dahl, and Johan Holmgren. Road travel time prediction-a micro-level sampling approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1613–1618. IEEE, 2013. [cited at p. 101]

[133] Martijn Mes, Matthieu van der Heijden, and Peter Schuur. Look-ahead strategies for dynamic pickup and delivery problems. *OR Spectrum*, 32(2):395–421, 2010. [cited at p. 75]

[134] Martijn Mes, Matthieu van der Heijden, and Aart van Harten. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59 – 75, 2007. [cited at p. 75]

[135] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. [cited at p. 25]

[136] Hokey Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377 – 386, 1989. [cited at p. 64, 67]

[137] Pablo A. Miranda, Carola A. Blazquez, Rodrigo Vergara, and Sebastian Weitzler. A novel methodology for designing a household waste collection system for insular zones. *Transportation Research Part E: Logistics and Transportation Review*, 77:227 – 247, 2015. [cited at p. 78]

[138] John E Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, pages 65–77, 2002. [cited at p. 18]

[139] Snežana Mitrović-Minić and Ramesh Krishnamurti. The multiple tsp with time windows: vehicle bounds based on precedence graphs. *Operations Research Letters*, 34(1):111–120, 2006. [cited at p. 48]

[140] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997. [cited at p. 26]

[141] Thamar E Mora, Abu B Sesay, Jorg Denzinger, H Golshan, G Poissant, and C Konecnik. Fuel optimization using biologically-inspired computational models. In *2008 7th International Pipeline Conference*, pages 167–173. American Society of Mechanical Engineers, 2008. [cited at p. 34]

[142] Edward K Morlok and Lazar N Spasovic. Redesigning rail-truck intermodal drayage operations for enhanced service and cost performance. In *Journal of the Transportation Research Forum*, volume 34, 1994. [cited at p. 2]

[143] Theo H J Muller and Peter G Furth. Integrating bus service planning with analysis, operational control and performance monitoring. In *ITS America 10th Annual Meeting and Exposition: Revolutionary Thinking, Real Results*, 2000. [cited at p. 95]

[144] Rajeev Namboothiri and Alan L. Erera. Planning local container drayage operations given a port access appointment system. *Transportation Research Part E: Logistics and Transportation Review*, 44(2):185 – 202, 2008. Selected Papers from the National Urban Freight Conference. [cited at p. 3, 73]

[145] George L Nemhauser and Laurence A Wolsey. Integer and combinatorial optimization john wiley & sons. *New York*, 1988. [cited at p. 18]

[146] Roberto Fernandes Tavares Neto and Moacir Godinho Filho. Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Eng. Appl. of AI*, 26(1):150–161, 2013. [cited at p. 31]

[147] Phuong Khanh Nguyen, Teodor Gabriel Crainic, and Michel Toulouse. A tabu search for time-dependent multi-zone multi-trip vehicle routing

problem with time windows. *European Journal of Operational Research*, 231(1):43–56, 2013. [cited at p. 77]

[148] Marcelo Norsworthy and Elena Craft. Emissions reduction analysis of voluntary clean truck programs at us ports. *Transportation Research Part D: Transport and Environment*, 22:23–27, 2013. [cited at p. 2]

[149] Jenny Nossack and Erwin Pesch. A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research*, 230(3):666 – 680, 2013. [cited at p. 2, 70, 75, 89]

[150] Aaron Luntala Nsakanda, Wilson L Price, Moustapha Diaby, and Marc Gravel. Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem. *European journal of operational research*, 178(2):634–638, 2007. [cited at p. 30]

[151] Iwao Okutani and Yorgos J Stephanedes. Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):1–11, 1984. [cited at p. 99]

[152] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991. [cited at p. 18]

[153] D.G. Pandelis, C.C. Karamatsoukis, and E.G. Kyriakidis. Finite and infinite-horizon single vehicle routing problems with a predefined customer sequence and pickup and delivery. *European Journal of Operational Research*, 231(3):577 – 586, 2013. [cited at p. 67]

[154] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Courier Corporation, 1982. [cited at p. 11]

[155] Julie Paquette, Jean-François Cordeau, Gilbert Laporte, and Marta MB Pascoal. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transportation Research Part B: Methodological*, 52:1 – 16, 2013. [cited at p. 68]

[156] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403 – 2435, 2007. [cited at p. 67]

[157] National Cooperative Freight Research Program. *National Cooperative Freight Research Program: A Status Report.* Transportation Research Board, 2011. [cited at p. 2, 97]

[158] Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016. [cited at p. 191]

[159] Jakob Puchinger and Günther R Raidl. An evolutionary algorithm for column generation in integer programming: an effective approach for 2d bin packing. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 642–651. Springer, 2004. [cited at p. 35]

[160] Jakob Puchinger and Günther R Raidl. *Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification.* Springer, 2005. [cited at p. vii, 33]

[161] Jakob Puchinger, Günther R Raidl, and Gabriele Koller. *Solving a real-world glass cutting problem.* Springer, 2004. [cited at p. 34]

[162] Luigi Di Puglia Pugliese and Francesca Guerriero. A computational s-tudy of solution approaches for the resource constrained elementary short-

est path problem. *Annals of Operations Research*, 201(1):131–157, 2012. [cited at p. 21]

[163] Luigi Di Puglia Pugliese and Francesca Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, 2013. [cited at p. 21]

[164] Line Blander Reinhardt, David Pisinger, Simon Spoorendonk, and Mikkel M Sigurd. Optimization of the drayage problem using exact methods. *INFOR: Information Systems and Operational Research*, pages 1–19, 2016. [cited at p. 62]

[165] Geraldo Ribeiro Filho and LA Nogueira Lorena. Constructive genetic algorithm and column generation: an application to graph coloring. In *Proceedings of APORS*, 2000. [cited at p. 35]

[166] John Rice and Erik Van Zwet. A simple and effective method for predicting travel times on freeways. *Intelligent Transportation Systems, IEEE Transactions on*, 5(3):200–207, 2004. [cited at p. 99]

[167] Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008. [cited at p. 21]

[168] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, pages 1–17, 2015. [cited at p. 191]

[169] Peter Ross. Hyper-heuristics. In *Search methodologies*, pages 529–556. Springer, 2005. [cited at p. 23]

[170] Louis-Martin Rousseau, Michel Gendreau, Gilles Pesant, and Filippo Focacci. Solving VRPTWs with constraint programming based column generation. *Annals of Operations Research*, 130(1-4):199–216, 2004. [cited at p. 20]

[171] Ilkyeong Moon Ruiyou Zhang, Won Young Yun. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(6):904 – 914, 2009. [cited at p. 70, 74]

[172] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer, 1998. [cited at p. 35]

[173] Wei Shen and Laura Wynter. Real-time road traffic fusion and prediction with GPS and fixed-sensor data. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1468–1475. IEEE, 2012. [cited at p. 99]

[174] Yindong Shen, Jia Xu, and Jingpeng Li. A probabilistic model for vehicle scheduling based on stochastic trip times. *Transportation Research Part B Methodological*, 85:19–31, 2016. [cited at p. 95, 96]

[175] Samaneh Shiri and Nathan Huynh. Optimization of drayage operations with time-window constraints. *International Journal of Production Economics*, 176:7 – 20, 2016. [cited at p. 2]

[176] Karen Smilowitz. Multi-resource routing with flexible tasks: an application in drayage operations. *Iie Transactions*, 38:577–590, 2006. [cited at p. 74]

[177] Brian L Smith and Michael J Demetsky. Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering*, 123(4):261–266, 1997. [cited at p. 117]

[178] Patrick Soriano and Michel Gendreau. Diversification strategies in tabu search algorithms for the maximum clique problem. *Annals of Operations Research*, 63(2):189–207, 1996. [cited at p. 24]

[179] LS Srinath. Parametric linear programming. In *Linear Programming*, pages 142–150. Springer, 1982. [cited at p. 15]

[180] J Michael Steele. Theoretical and computational aspects of simulated annealing. *Journal of the American Statistical Association*, 85(410):596–597, 1990. [cited at p. 26]

[181] Sebastian Sterzik and Herbert Kopfer. A tabu search heuristic for the inland container transportation problem. *Computers and Operations Research*, 40(4):953 – 962, 2013. [cited at p. 72]

[182] Thomas Stützle, Manuel López-Ibáñez, and Marco Dorigo. A concise overview of applications of ant colony optimization. *Wiley Encyclopedia of Operations Research and Management Science*, 2011. [cited at p. 31]

[183] Z. Sun and G. Fox. Traffic flow forecasting based on combination of multidimensional scaling and svm. *International Journal of Intelligent Transportation Systems Research*, 12(1):20–25, 2014. [cited at p. 100]

[184] Sarosh Talukdar, Lars Baerentzen, Andrew Gove, and Pedro De Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, 4(4):295–321, 1998. [cited at p. 34]

[185] C.C.R. Tan and J.E. Beasley. A heuristic algorithm for the period vehicle routing problem. *Omega*, 12(5):497–504, 1984. [cited at p. 76]

[186] Lixin Tang, Jiao Zhao, and Jiyin Liu. Modeling and solution of the joint quay crane and truck scheduling problem. *European Journal of Operational Research*, 236(3):978–990, 2014. [cited at p. 62]

[187] PVV Theja and Lelitha Vanajakshi. Short term prediction of traffic parameters using support vector machines technique. In *Emerging Trends in Engineering and Technology (ICETET), 2010 3rd International Conference on*, pages 70–75. IEEE, 2010. [cited at p. 118]

[188] Darsono Tjokroamidjojo, Erhan Kutanoglu, and G. Don Taylor. Quantifying the value of advance load information in truckload trucking. *Transportation Research Part E: Logistics and Transportation Review*, 42(4):340 – 357, 2006. [cited at p. 75]

[189] Vigo Daniele Toth, Paolo. *The Vehicle Routing Problem.* SIAM, 2001. [cited at p. 40, 44, 46, 50, 64]

[190] Pierre Trudeau and Moshe Dror. Stochastic inventory routing: Route design with stockouts and route failures. *Transportation Science*, 26(3):171–184, 1992. [cited at p. 77]

[191] JWC Van Lint. Online learning solutions for freeway travel time prediction. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):38–47, 2008. [cited at p. 100]

[192] Rinde RS van Lon and Tom Holvoet. Towards systematic evaluation of multi-agent systems in large scale and dynamic logistics. In *PRIMA 2015:*

Principles and Practice of Multi-Agent Systems, pages 248–264. Springer, 2015. [cited at p. 191]

[193] Michel Vasquez, Jin-Kao Hao, et al. A hybrid approach for the 0-1 multidimensional knapsack problem. In *IJCAI*, pages 328–333, 2001. [cited at p. 34]

[194] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012. [cited at p. 78]

[195] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, 234(3):658–673, 2014. [cited at p. 34]

[196] Milorad Vidović, Dražen Popović, Branislava Ratković, and Gordana Radivojević. Generalized mixed integer and vns heuristic approach to solving the multisize containers drayage problem. *International Transactions in Operational Research*, 2016. [cited at p. 62]

[197] Milorad Vidović, Gordana Radivojević, and Branislava Raković. Vehicle routing in containers pickup up and delivery processes. *Procedia-Social and Behavioral Sciences*, 20:335–343, 2011. [cited at p. 74]

[198] Monize Sâmara Visentini, Denis Borenstein, Jing Quan Li, and Pitu B. Mirchandani. Review of real-time vehicle schedule recovery methods in transportation services. *Journal of Scheduling*, 17(6):541–567, 2014. [cited at p. 96]

[199] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Short-term traffic forecasting: Where we are and where we're going. *Transporta-*

tion Research Part C: Emerging Technologies, 43, Part 1(0):3 – 19, 2014. [cited at p. 98]

[200] Stefan Voss. Book review: Morco dorigo and thomas stützle: Ant colony optimization (2004) ISBN 0-262-04219-3, MIT press, cambridge. *Math. Meth. of OR*, 63(1):191–192, 2006. [cited at p. 31]

[201] Zun Wang. *Truck GPS Data in Freight Planning: Methodologies and Applications for Measurement and Forecasting.* PhD thesis, 2015. [cited at p. 191]

[202] ZX Wang, Felix TS Chan, SH Chung, and Ben Niu. Minimization of delay and travel time of yard trucks in container terminals using an improved ga with guidance search. *Mathematical Problems in Engineering*, 2015, 2015. [cited at p. 98]

[203] Daniel S Weile and Eric Michielssen. Genetic algorithm optimization applied to electromagnetics: A review. *Antennas and Propagation, IEEE Transactions on*, 45(3):343–353, 1997. [cited at p. 30]

[204] Shengqiang Wen and Pengfei Zhou. A Container Vehicle Routing Model with Variable Traveling Time. In *IEEE International Conference on Automation and Logistics*, 2007. [cited at p. 75]

[205] N. Wieberneit. Service network design for freight transportation: A review. *OR Spectrum*, 30(1):77–112, 2008. [cited at p. 56, 57, 61, 62]

[206] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, Apr 1997. [cited at p. 70]

[207] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014. [cited at p. 17]

[208] Amelia C.Regan Xiubin Wang. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological*, 36(2):97 – 112, 2002. [cited at p. 59, 70, 72, 74, 89]

[209] Zhaojie Xue, Wei-Hua Lin, Lixin Miao, and Canrong Zhang. Local container drayage problem with tractor and trailer operating in separable mode. *Flexible Services and Manufacturing Journal*, 27(2-3):431–450, 2015. [cited at p. 62]

[210] Zhaojie Xue, Canrong Zhang, Wei-Hua Lin, Lixin Miao, and Peng Yang. A tabu search heuristic for the local container drayage problem under a new operation mode. *Transportation Research Part E: Logistics and Transportation Review*, 62:136–150, 2014. [cited at p. 73]

[211] Chen Xumei, Gong Huibo, and Jingnan Wang. Brt vehicle travel time prediction based on svm and kalman filter. *Journal of transportation systems engineering and information technology*, 12(4):29–34, 2012. [cited at p. 118]

[212] Xin-She Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010. [cited at p. 31]

[213] B. Yu, T. Ye, X.-M. Tian, G.-B. Ning, and S.-Q. Zhong. Bus travel-time prediction with a forgetting factor. *Journal of Computing in Civil Engineering*, 28(3), 2014. [cited at p. 106]

[214] Guoqiang Yu and Changshui Zhang. Switching arima model based forecasting for traffic flow. In *Acoustics, Speech, and Signal Processing, 2004.*

Proceedings.(ICASSP'04). IEEE International Conference on, volume 2, pages ii–429. IEEE, 2004. [cited at p. 100, 113]

[215] Fang Yuan and Ruey Long Cheu. Incident detection using support vector machines. *Transportation Research Part C: Emerging Technologies*, 11(34):309 – 328, 2003. [cited at p. 111]

[216] Ivan Zelinka, Václav Snášel, and Ajith Abraham, editors. *Handbook of Optimization - From Classical to Modern Approach*, volume 38 of *Intelligent Systems Reference Library*. Springer, 2013. [cited at p. 26]

[217] Hui Zhang, Ruiyou Zhang, MM Huang, and Haibo Shi. Modeling and analyses of container drayage transportation problem with the objective of low carbons. In *Control and Decision Conference (CCDC), 2015 27th Chinese*, pages 4654–4658. IEEE, 2015. [cited at p. 62]

[218] Ruiyou Zhang, Won Young Yun, and Il Kyeong Moon. Modeling and optimization of a container drayage problem with resource constraints. *International Journal of Production Economics*, 133(1):351 – 359, 2011. Leading Edge of Inventory Research. [cited at p. 74]

[219] Ruiyou Zhang, WonYoung Yun, and Herbert Kopfer. Heuristic-based truck scheduling for inland container transportation. *OR Spectrum*, 32(3):787–808, 2010. [cited at p. 48, 70, 71, 74, 80, 81, 88, 89, 92]

[220] Tao Zhang, W.Art Chaovalitwongse, and Yuejie Zhang. Integrated ant colony and tabu search approach for time dependent vehicle routing problems with simultaneous pickup and delivery. *Journal of Combinatorial Optimization*, 28(1):288–309, 2014. [cited at p. 68]

[221] Sun Zhanquan, Feng Jinqiao, and Liu Wei. Travel time forecasting based on phase space reconstruction and svm. In *Optoelectronics and Image Processing (ICOIP), 2010 International Conference on*, volume 2, pages 692–695. IEEE, 2010. [cited at p. 118]

[222] Qiuhong Zhao. *A heuristic algorithm based on single point search.* Science press, 2013. [cited at p. 42, 43]

[223] Kenny Q Zhu. Population diversity in genetic algorithm for vehicle routing problem with time windows. *Department of Computer Science, National University of Singapore*, 2003. [cited at p. 30]