


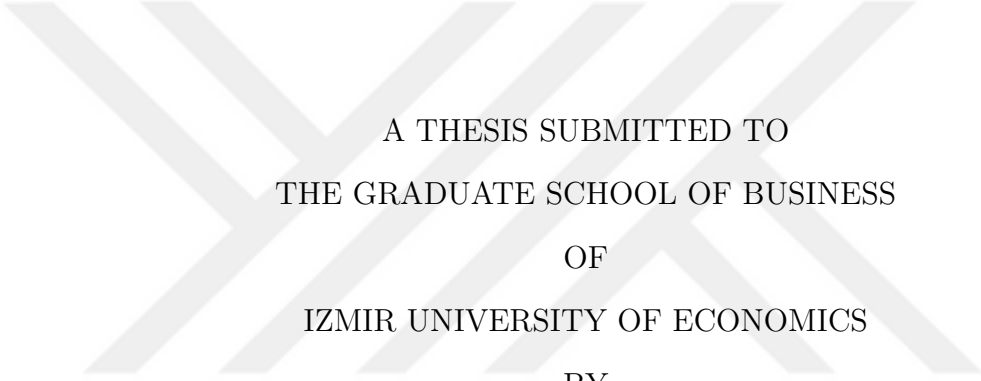
**SHIPMENT CONSOLIDATION AND
DISPATCHING PROBLEM WITH
TRANSSHIPMENT TERMINALS**



SİNEM TOKCAER

February 2018

SHIPMENT CONSOLIDATION AND DISPATCHING PROBLEM WITH TRANSSHIPMENT TERMINALS



A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF BUSINESS
OF
IZMIR UNIVERSITY OF ECONOMICS
BY

SİNEM TOKCAER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE GRADUATE SCHOOL OF BUSINESS

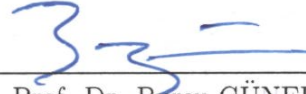
February 2018

Approval of the Graduate School of Social Sciences



Prof. Dr. Hasan Fehmi BAKLACI
Director of the Institute

I certify that this thesis satisfies all the requirements as a thesis for the degree of
Doctor of Philosophy.

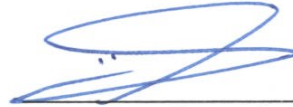


Assoc. Prof. Dr. Burcu GÜNERİ ÇANGARLI
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully
adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.



Asst. Prof. Dr. Ahmet CAMCI
Co-Advisor



Assoc. Prof. Dr. Özgür ÖZPEYNİRCİ
Advisor

Examining Committee Members

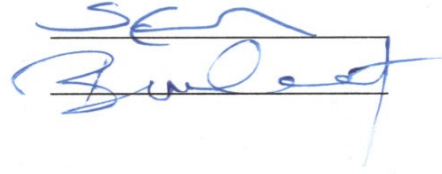
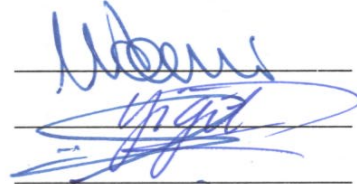
Assoc. Prof. Dr. Muhittin H. DEMİR

Assoc. Prof. Dr. Yiğit KAZANÇOĞLU

Assoc. Prof. Dr. Özgür ÖZPEYNİRCİ

Assoc. Prof. Dr. Selin ÖZPEYNİRCİ

Asst. Prof. Dr. Önder BULUT



ABSTRACT

SHIPMENT CONSOLIDATION AND DISPATCHING PROBLEM WITH TRANSSHIPMENT TERMINALS

Tokcaer, Sinem

Ph.D. in Business Administration

Supervisor: Assoc. Prof. Dr. Özgür ÖZPEYNİRCİ

Co-Advisor: Asst. Prof. Dr. Ahmet CAMCI

December 2017, 120 Pages

Shipment consolidation and dispatching problem is an operational planning problem of long haul freight forwarders in their daily operations planning. Freight forwarders aim at planning the delivery of less-than truckload (LTL) customer orders within specified time-windows by using sub-contracted haulers' vehicles. The orders are delivered either directly or by using a cross-dock. Each transshipment decision implies a cost proportional to the size of the order. The main path of a vehicle is defined regarding the farthest destination in the vehicle and excessive deviations from the main path is not allowed. The cost of a vehicle is defined either by the prices indicated in the annual contracts between freight forwarder and sub-contracted vehicle owner or by the spot market prices. Main objective of the freight forwarder is to minimize the total cost of vehicles, including the cost of rented vehicles and transshipped orders.

Respectively, we introduce 2 variants of Shipment Consolidation and Dispatching Problem (SCDP) with aforementioned assumptions. In the first problem, the annual contracts with sub-contracted haulers define the cost of vehicle's route. In the second problem, the cost is defined by the sum of fixed cost identified by the farthest destination in the vehicle and the extra charge of additional stops.

For both problems, we propose a feasibility check mechanism for the generated vehicles, which also considers 5 real life assumptions; stability, orientation, weight distribution, loading sequence and stacking constraints.

The analytic solution methodologies that we propose throughout the thesis produce good quality solutions in a very short computation time. Additionally, we tested proposed solution methodologies on a real-life instance, and the obtained solutions provide approximately 10% savings for both algorithms, and are applicable in real life. In this sense, proposed solution methodology may benefit international freight forwarders in three perspectives; (i) cost savings, (ii) efficient use of human resources, and (iii) time utilization.

This thesis is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK, project no: 214M195).

Keywords: Transportation Planning, Shipment Consolidation, Dispatching, Transshipment Terminals.

ÖZET

AKTARMA TERMİNALLİ YÜK BİRLEŞTİRME VE SEVKİYAT PROBLEMİ

Tokcaer, Sinem

İşletme Doktora Programı

Tez Yöneticisi: Doç. Dr. Özgür ÖZPEYNİRCİ

Ortak Tez Yöneticisi: Yrd. Doç. Dr. Ahmet CAMCI

Aralık 2017, 120 Sayfa

Uzak mesafe ve uluslararası yük taşımacılığı, rekabetin oldukça fazla ve hizmet sağlayıcıların en iyi hizmeti, en düşük maliyetle vermek durumunda olduğu bir pazardır. Bu bakımdan, hizmet sağlayıcıları, hızlı ve ölçek ekonomisinden faydalanan verimli ve düşük maliyetli yük birleştirme sistemleri geliştirmektedirler. Yük konsolidasyonu ve yükleme planlaması problemi, farklı müşterilere ait, farklı boyut ve miktardaki yüklerin bir araç içerisinde yüklenerek, müşterinin talep ettiği zaman aralığı içerisinde dağıtımını sağlamaya yönelik bir problem olup, en temel amacı oluşan maliyetleri en azlamaktır. Bunun yanı sıra, aracın seyahat süresini kısaltmak amacıyla, ürünlerin son adresine aracın varış ülkesinde bulunan bir aktarma terminali üzerinden teslim edilmesi ise hizmet sağlayıcıları tarafından sıklıkla kullanılan bir yöntemdir.

Yukarıdaki varsayımlar doğrultusunda, bu tez ile iki tür Yük Birleştirme ve Yük Planlaması problemi tanımlanmıştır. İlk problemde, araç rotaları ve maliyetleri yıllık sözleşmeler doğrultusunda belirlenmekte, ikinci problemde ise araçtaki yüklerin arasında en uzak mesafeye sahip olan yükün teslimat noktası araç rotasını ve maliyetini belirlemektedir. Bunun yanı sıra, denge, oryantasyon, ağırlık dağılımı, yükleme sırası ve istifleme gibi gerçek hayatta var olan yükleme kısıtlarını

dikkate alan bir kontrol mekanizması her iki problemin çözüm yöntemine entegre edilmiştir.

Yapılan deneyler sonucunda, geliştirilen çözüm yöntemlerinin kısa sürede yüksek kaliteli çözüm ürettiği görülmüştür. Ayrıca, önerilen çözüm yöntemleri bir örnek üzerinde test edilmiş ve yaklaşık %10 civarında iyileştirmenin mümkün olduğu görülmüştür. Bu anlamda, analitik çözüm yöntemlerinin uzak mesafe parsiyel yük taşımacılığı hizmeti sağlayıcılarına (i) maliyet, (ii) insan kaynağının etkin kullanımı ve (iii) zaman kullanımı açısından fayda sağlayabileceği öngörülmektedir.

Bu tez kapsamında yürütülen çalışmalar Türkiye Bilimsel ve Teknolojik Araştırma Kurumu tarafından desteklenmiştir (TÜBİTAK, Proje no: 214M195).

Anahtar sözcükler: Ulaşım Planlaması, Yük Birleştirme, Yükleme Planlaması, Aktarma Terminali.

To Deniz;

My Little Angel

I'm sorry for the time I had to steal from you...

Acknowledgements

People say PhD is a "journey", however all journeys has an ending. My PhD experience was like a never-ending roller coaster with many ups and downs. When I first came up with this subject matter, all methods applied herein this thesis was something like a foreign language (maybe Chinese), and I found myself in a cycle of "learning-coding-debugging-learning while debugging-coding-debugging" and so on. At this point, where I delivered my thesis, there were two brilliant academic advisors supporting me; Assoc. Prof. Dr. Dr. Özgür Özpeynirci and Asst. Prof. Dr. Ahmet Camci, who thought me everything I know, starting from the scratch. Without their immense knowledge and hard work, I would not have accomplished this thesis with so many outputs. They also have gone beyond their academic duties, and put so much effort in dispelling my anxieties and worries about my capacity, and pushing me to do better. I would like to express my sincere gratitude to both of them for their patience, unrelenting support and true friendship. It was an honor for me to be first their student, then colleagues and finally business partners.

Besides my advisors, I am also grateful to my committee members; Assoc. Prof. Dr. Selin Özpeynirci and Assoc. Prof. Dr. Muhittin Hakan Demir for their insightful comments and challenging questions, which compelled me to make this thesis better. I would like to add a special thanks to Assoc. Prof. Dr. Selin Özpeynirci for her sincere friendship.

There are also a number of people behind this thesis, who deserve to be both acknowledged and thanked here. Firstly, I am grateful to my husband Murat Tokcaer for believing in me. When I start to pursue an academic career, he knew that we step in a psychological and economic crisis; however, he supported me

in all possible ways to accomplish this task in my hand. I am also indebted to my family, my mom Jale Huntürk for her passionate vision of me as a doctor and relentless care she showed all through my life, and to my dad S. Erkan Huntürk for his love and encouragement. I am thankful to my brother Tuna Huntürk for supporting me spiritually and taking me to his boat tours, and Begüm Huntürk for being more than a real sister.

My time at İzmir University of Economics was enjoyable with many friends that became a part of my life. I would like to thank Cansu Yıldırım for being “the blossoming rose of the same sapling” with me, and sharing her academic experience, supporting me in both academic and administrative duties. I am thankful to Mert Günerergin for all those cheerful moments we shared, and logistic service he provided all through my pregnancy. I would like to thank Murad Canbulut, who patiently listened all my complaints about job, life and everything else, and shared his wisdom with me. I am thankful to Gökçe Erbuğa for her positive energy and great sense of humor, which always made me laugh. I would also like to thank Doğan Başkır and Muhittin Sağnak for being so kind and supportive friends.

I would like to thank my fellows of 9th floor; Ezgi Özkan, Esra Onater, Birce Doğrucalı, Cansu Tayaksi, Funda Savaş and Funda Sarıcı for the warm and cheerful environment they have created. They were always ready to support me when I am in trouble. It was a pleasure for me to meet you girls. I am thankful to Arya Sevgen for being an unexpected fellow when I feel so alone, and a perfect copilot in car, sharing long travels to the university. I would like to thank Seda Lafcı for all her assistance and being my academic sister.

My sincere thanks also goes to academic staff of Department of Logistics Management, who have contributed immensely to my personal and professional time at

Izmir University of Economics. I especially thank to Assoc. Prof. Dr. Bengü Oflaç and Asst. Prof Dr. Aysu Göçer for their heartfelt friendship in all aspects.

All throughout this process, there were some people, whom were always with me holding my hand. In the same vein, I would like to thank Başak Çallıođlu for feeling the need of a friendship more than 20 years, and understanding whatever I do. I am also thankful to Ayşe Erođlu, Dilek and Hakan Tolungüç for sharing good and bad times of this pace of my life.

Finally, I would like to reserve a special thanks to my daughter Deniz Tokcaer for giving me strength to stand upright against the troubles, teaching me to be more patient, and growing me older. I am so lucky to have you as my zest of life and motivation to go on further.

Table of Contents

Abstract	iii
Özet	v
Acknowledgements	viii
Table of Contents	xiii
List of Tables	xv
List of Figures	xvi
List of Abbreviations	xvii
1 Introduction	1
1.1 Problem Definition	2
1.2 Motivation and Objectives	4
1.3 Organisation of the Thesis	6
2 Literature Review	9
2.1 Shipment Consolidation	9
2.1.1 Temporal Consolidation	11
2.1.2 Spatial Consolidation	12
2.2 Transportation Planning Problems	12
2.2.1 Network Design Problems	13

2.2.2	Vehicle Routing Problems	14
2.2.3	Intermediate Facilities in Transportation Planning Problems	15
2.3	Literature Gap	17
3	Shipment Consolidation and Dispatching with Transshipment	
	Terminals and Fixed Routes	19
3.1	Introduction	19
3.2	Problem Statement and Mathematical Model	21
3.2.1	Mathematical Model	21
3.2.2	Computational Complexity	27
3.3	Lower Bound Algorithms	28
3.3.1	Lower Bound 1: Relax Integrality Constraints	29
3.3.2	Lower Bound 2: Relax Capacity Constraints	30
3.4	Variable Neighborhood Search	32
3.5	Experiments	36
3.5.1	Randomly Generated Instances	36
3.5.2	Preliminary Experiments	38
3.5.3	Computational Results	41
3.6	Conclusion and Further Research	46
4	Shipment Consolidation and Dispatching with Transshipment	
	Terminals and Spot Market Prices	48
4.1	Introduction	48
4.2	Literature Review	49
4.3	Problem Formulation	52
4.3.1	Assumptions	52
4.3.2	Mathematical Model	54
4.4	Solution Methodology	58

4.4.1	Dantzig-Wolfe Decomposition	59
4.4.2	Column Generation	61
4.4.3	Branch-and-Price Algorithm	61
4.5	Computational Experiments	64
4.5.1	Performances of Original Formulation and Branch-and-Price	67
4.5.2	Computational Performance of B&P	69
4.6	Conclusion and Future Works	76
5	Integrated Three Dimensional Bin Packing Problem and SCD- TT	78
5.1	Introduction	78
5.2	Literature Review	79
5.3	Mathematical Model	82
5.4	Heuristic Approach	85
5.5	Computational Experiments	86
5.5.1	Preliminary Experiment	88
5.5.2	SCD-TTFR with Loading Constraints	89
5.5.3	SCD-TTSM with Bin Packing	93
5.6	Conclusion and Future works	98
6	Conclusion and Future Research	101
	Bibliography	106
	Curriculum Vitae	119

List of Tables

3.1	Neighborhood Set N_k	34
3.2	The levels of parameters controlling instance generation	37
3.3	All possible problem combinations	39
3.4	Performances of all possible problem combinations (%) in terms of CPU time	39
3.5	Results of preliminary experiments	40
3.6	The performance of MM2 under different number of predefined routes in the route set	41
3.7	Performances of Mathematical Models within 3600 sec	42
3.8	Performances of Lower Bound Algorithms within 3600 sec	44
3.9	Performances of Variable Neighbourhood Search (VNS) Algorithm and Mathematical Models	45
4.1	Reported Performances of Preliminary Experiment	65
4.2	Performances of Original Formulation and Branch-and-Price	69
4.3	Solution Quality of Branch-and-Price	69
4.4	Computation Time Performance of Branch-and-Price	71
4.5	Number of Columns Added	73
4.6	Number of Nodes Visited	74
5.1	Performances of Container Loading Problem (CLP) and Moura and Oliveira (2005)	88
5.2	Utilization Rates of VNS without and with CLP	91

5.3	Utilization Rate of Rejected Vehicles by CLP	92
5.4	CPU and Quality Performances of VNS without and with CLP . .	93
5.5	Number of added columns and CPU of Branch-and-Price (B&P) with and without CLP	95
5.6	Effect of CLP on Number of Vehicles and Cost in the Optimal Solution	96
5.7	Utilization Rate of B&P without and with CLP (%)	97
5.8	Utilization Rate of Rejected Vehiles by CLP Algorithm (%) . . .	97

List of Figures

1.1	Delivery Networks	4
4.1	Available days for departure for three orders	53
4.2	Convergence Curves of B&P Algorithm without MIP	66
4.3	Convergence Curves of B&P Algorithm with MIP	66
4.4	Mean Deviation of Lower Bounds	68
4.5	Convergence Curves of B&P Algorithm for an instance	75
4.6	Number of Nodes and Columns Compared to Computational Per- formances	75
5.1	Column Generation Procedure with CLP Algorithm on a Given Node	94

List of Acronyms

3D-BPP Three Dimensional Bin Packing Problem.

B&P Branch-and-Price.

CLP Container Loading Problem.

LTL less-than-truckload.

SCD-TT Shipment Consolidation and Dispatching Problem with Transshipment Terminals.

SCD-TTFR Shipment Consolidation and Dispatching Problem with Transshipment Terminals and Fixed Routes.

SCD-TTSM Shipment Consolidation and Dispatching Problem with Transshipment Terminals and Spot Market Prices.

VNS Variable Neighbourhood Search.

VRP Vehicle Routing Problem.

Chapter 1

Introduction

Long haul and international freight transportation is a highly competitive market, where the freight forwarder companies have to deliver the best service with low prices. In order to meet this challenge, the freight forwarders mostly establish their own consolidation systems to achieve economies of scale and efficient use of the owned or rented vehicles. In this sense, freight forwarders aim to plan the delivery of less-than-truckload (LTL) customer orders with different sizes and volumes within specified time-windows; while their main objective is to minimize the total cost. Additionally, most of the freight forwarders use transshipment terminals for various reasons including the reduction of cost or travelling time. Freight forwarders make such consolidation plans frequently in their daily operations, and usually find solutions manually. If the freight forwarder has a large amount of LTL shipments, it is usual to assign one or more staff, namely dispatchers, only for planning the consolidation of orders. This thesis aims to define Shipment Consolidation and Dispatching Problem with Transshipment Terminals (SCD-TT) of the freight forwarders, and develop an analytical solution methodology for the problem.

1.1 Problem Definition

The assumptions of real-life problem can be categorized under four main structures; (i) order structure, (ii) vehicle structure and loading constraints, (iii) route and cost structure, and (iv) delivery structure.

Order structure defines the previously known information on the orders; such as, the number of pieces, which are mostly pallets or boxes, length, height, width and weight for each piece, required time windows for delivery (release day and deadline), destination and any special conditions required for a safe handling of the order (hazardous materials, stackability etc.).

Vehicle structure and loading constraints define the capacity for vehicles and the constraints to be considered while making dispatching plans. For a stable and safe loading, the items in a vehicle should be loaded in a way that they do not fall down or tilt over each other, and the weight distribution in a vehicle should be balanced. Also, the amount of weight on an item must not exceed the load bearing strength threshold of that item (Bischoff, 2006). For instance, the decision maker should consider the unstackable and fragile shipments that cannot be placed on top of each other. Additionally, the orders should be loaded on the vehicle regarding the unloading sequence, and orders delivered through a transshipment terminal may be mixed and grouped together to benefit from use of space as those items will be sorted in the transshipment terminal before delivery to the final destination.

Route and cost structure originates particularly from the business environment. Since freight forwarders rent vehicles on a one-way trip basis, they don't have to consider returns of the vehicles to origin. Moreover, the total length of a trip is mostly more than 5,000 kilometres in long haul transportation, thus the routing decision does not have to be as precise as it should be in vehicle routing of short-

haul transportation. Furthermore, dynamic business environment requires quick decision making, so freight forwarders adopt simplifying procedures to support easy routing decisions with minimal data requirement. In this regard, the farthest delivery point (stop) in a vehicle defines the main route of that vehicle, and extra deliveries (stops) with the same vehicle is allowed, only if the delivery point requires minimum deviation from the main route. Since travelling long way off the main route results in greater overall cost and loss of time, excessive deviations from the main path are not allowed for additional stops. Even if an extra stop is along the route, it increases travelling time, and may result in late deliveries. Therefore, less stops along the route is favourable, and usually a definite number of extra stops are allowed.

The cost structure is strongly related with routing decisions, and the distance to farthest delivery point determines the fixed cost of that vehicle. Each extra stop is incurred by an extra fixed cost, which is also associated with the limit on deviation from the main route. The fixed cost of a vehicle may be defined in two ways; (i) annual contracts made with the sub-contracted haulers and (ii) spot market prices. Such annual contracts outline the routes defined by zones or regions, and the price of a route is fixed with respect to the farthest distance in that zone/region. Predefined fixed routes also define the possible stops along that route. In this way, excessive deviations from the main route is prevented. Spot market prices are subject to seasonal changes, therefore, fixed cost includes both operating cost, which enables price fluctuations, and per kilometre cost multiplied by the farthest distance in the vehicle.

Delivery structure is shaped by the agreements with agencies in different countries, and each shipment is delivered directly to its destination or through the transshipment terminals of the contracted foreign agency. Thus, there is a hybrid

delivery structure, where a vehicle is allowed to visit both destinations and transshipment terminals on the same trip. The main objective of using transshipment terminals is to decrease the travelling time as well as total distance and total cost by reducing the number of stops. Such a hybrid delivery structure also extends the range of service by enabling local deliveries to unvisited zones or countries. Each item delivered through a transshipment terminal incurs a cost of transportation to final destination, which is proportional to the size of the item.

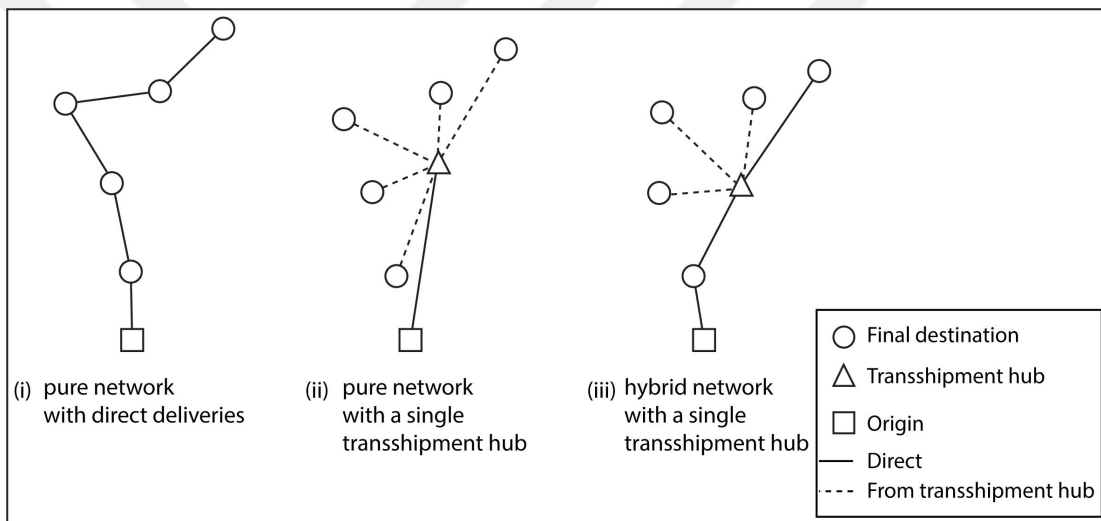


Figure 1.1: Delivery Networks

1.2 Motivation and Objectives

Although above stated problem is a common operational problem in real-life practices, and frequently faced in daily operations planning of freight forwarders, the problem is usually solved manually by the decision maker, namely dispatcher. The planning phase may take several hours, as the problem inherently has various decisions to be considered. Moreover, the dispatching plan proposed by the

dispatcher may not be the optimal solution, and may be subject to human error. The dynamic environment of the business also requires quick decisions of changes in dispatching plans. For instance, the orders which are supposed to be ready on departure day may be delayed, or the number of boxes in an order may increase/decrease; thus, substantial changes in dispatching plans may be necessary and it may be difficult to manage the process.

With all these aspects of the ongoing business operations with manual dispatching plans, the outcome is higher operating costs and risk of errors and inefficient use of human resources and time. In this respect, the objective of this thesis is to design an analytical solution approach for the shipment consolidation and dispatching in the operations planning of freight forwarders. An analytical solution methodology may benefit international freight forwarders in three perspectives; (i) cost savings, (ii) efficient use of human resources, and (iii) time utilization. For instance, Erera et al. (2013) showed possible cost savings of 300,000 USD (approximately 6-7%) with an application on a similar problem with a freight forwarder in United States of America.

The thesis also contributes to existing literature by examining the problem with a different approach. Firstly, simplified routing and following cost structure is not frequently studied in the literature. To the best of our knowledge, Koca and Yıldırım (2012) addressed to routes defined by the farthest destination along the route, and fixed costs associated with that destination. Secondly, hybrid delivery structure, which allows stops at both destinations and transshipment terminals with the same vehicle, is not frequently studied in the literature as well (Guastaroba et al., 2016). Thirdly, practical shipment consolidation problem of freight forwarders has differences from the consolidation and transportation problems studied in the literature. Thus, the thesis contributes to existing literature by

introducing a variant of shipment consolidation and dispatching problem, and analyzing the main characteristics. The problem is NP-hard and it embraces research directions for further studies.

1.3 Organisation of the Thesis

In the light of aforementioned problem definition, we define two different problems with similar assumptions, yet different in cost structures. The first problem focuses on the problem type, in which the fixed cost of a vehicle is defined with respect to the annual contracts made by the haulers. In this scope, we define the problem formulation with predefined routes, and propose a mathematical model and a heuristic algorithm that decides the route, date of departure of the vehicle and delivery type for the orders. In the second problem, we formulate the problem wherein the spot market defines the fixed cost of a vehicle. We aim to develop an exact solution methodology for the problem, and identify the fixed cost of each individual vehicle regarding the farthest destination in respective vehicle.

In order to decrease computational complexities, the solution methodology of both identified problems disregards the bin packing assumptions, such as weight distribution, stackability etc. In both problems, we constrain that the total volume, weight and loading meter of orders in a vehicle cannot exceed vehicle capacity. Hence, the solutions obtained in both problem may be infeasible or suboptimal when the bin packing assumptions are concerned. Therefore, we develop a bin packing algorithm, which checks the feasibility of obtained solutions, and embed the bin packing heuristic to the algorithms of both problems.

In this respect, this thesis is organized as follows. In the second chapter, we discuss the relevant literature on the proposed problem, and reveal the similarities and

differences of the problem from the existing literature.

In the third chapter, we define the assumptions of the problem with predefined routes with fixed costs, namely Shipment Consolidation and Dispatching Problem with Transshipment Terminals and Fixed Routes (SCD-TTFR), develop the mathematical model and valid inequalities to enhance the model, examine the computational complexity, propose two lower bound algorithms and an upper bound algorithm. Subsequently, we explain randomly generated instances, and examine the results of computational experiments conducted on both mathematical models, lower and upper bound algorithms.

In the fourth chapter, we focus on the Shipment Consolidation and Dispatching Problem with Transshipment Terminals and Spot Market Prices (SCD-TTSM). We firstly define the assumptions of SCD-TTSM problem, and introduce a mathematical model. Then we apply Dantzig-Wolfe Decomposition approach, and propose a column generation and branch-and-price algorithm. Finally, we conduct computational experiments with the randomly generated instances used in Chapter 3 on the solution approach.

In the fifth chapter, we propose the solution methodology for the bin packing assumptions of real-life application. We firstly analyse the Three Dimensional Bin Packing Problem (3D-BPP) assumptions with reference to the recent literature. Then we propose a mathematical model and a heuristic algorithm, and integrate the heuristic algorithm to SCD-TTFR and SCD-TTSM. Finally, we report the results of computational experiments conducted on the instances used in Bischoff and Ratcliff (1995a), and compare the performance of our algorithm with Moura and Oliveira (2005). Moreover, we perform experiments on 3D-BPP integrated SCD-TTFR and SCD-TTSM, and report the effect of 3D-BPP assumptions on both problems.

In the last chapter, we discuss the results of the experiments, and illustrate the possible savings for a freight forwarder. We also point to the limitations of the thesis and further research directions.



Chapter 2

Literature Review

This chapter presents an overview of the literature on Shipment Consolidation and Dispatching Problem with Transshipment Terminals SCD-TT from different perspectives of transportation problems. In order to examine the place of SCD-TT in transportation problems with broad range of focuses, we categorize the literature based on two decisions; shipment consolidation and transportation planning problems. Accordingly, in the first section, we discuss the decision of shipment consolidation, including types, benefits and disadvantages. In the second section, we address the transportation planning problems with different levels of planning. We also discuss the intermediate facilities in transportation planning problems.

2.1 Shipment Consolidation

Shipment consolidation is a logistics strategy, where at least two or more customer orders are dispatched on the same vehicle to create larger batches (Higginson and Bookbinder, 1994), and decrease the total cost of transportation (Hall, 1987a).

Yet, lower cost of transportation may end up with some penalties. While consolidating items with different release dates, some orders may await until consolidation time; thus, an inventory holding cost incurs. Consolidation also requires handling and sorting in consolidation terminals, herein that an additional handling cost appears. Additionally, consolidation of items to different destinations leads to longer routes, thus late deliveries ensuing from increasing travelling times. The additional costs or cost factors pertaining to consolidation requires a deliberate strategy to secure the best outcome (Hall, 1987a).

There are various classifications of shipment consolidation strategies in the existing literature. Originally Brennan classifies shipment consolidation strategies in three; temporal, spatial and product consolidation (Brennan, 1981). Items are consolidated in larger batches regarding the time of release of the larger batch in temporal consolidation, whereas spatial consolidation concerns with the aggregation of items regarding geographical aspects. Product consolidation simply consolidates different types of products to create larger batch of shipments. Other than Brennan's classification, Sheffi (1986) proposed 6 classes of shipment consolidation, which are; vehicle units, containers, channels, network, time and tours. Based on Brennan's classification of temporal and spatial consolidation, Hall (1987a) classified consolidation strategies in three; inventory consolidation (as a form of temporal consolidation), vehicle and terminal consolidation (as two different forms of spatial consolidation).

Hereinafter, we will focus on classification of Brennan on shipment consolidation, while discussing only temporal and spatial consolidation, as product consolidation literature on product consolidation is scarce (Min and Cooper, 1990).

2.1.1 Temporal Consolidation

Temporal consolidation strategy focuses on the “*time*” to create a larger batch of orders; in other words, the main decision of this strategy is to determine “*when*” to release a group of orders. Release time of orders is essentially important as it determines the batch size, and requires a dispatching rule, namely operating routine (Abdelwahab and Sargious, 1990). In order to develop an operating routine, there are two main decisions to be taken; the time to dispatch a vehicle, and the size of a batch (Çetinkaya and Lee, 2000). Accordingly, there are three commonly used consolidation policies, which are;

- *Q policy* requires a batch of Q units to release a shipment. This policy mainly focuses on maximum utilization shipment units, hence inherently implies cost savings.
- *T policy* requires a determined shipping date to a dispatch orders; thus, a shipment is released in every T units of time. The policy target is usually set regarding the customer requirements on shipping date; hence, the main objective is customer service level (Higginson and Bookbinder, 1994).
- *Hybrid policy*: holds orders until the earliest of the quantity (Q) or time (T) target is achieved. This policy is commonly known as time-and-quantity policy, and intends to get the best outcome of time or quantity policy.

Because the main motivation of temporal consolidation is to obtain a suitable consolidation policy, the studies on it inherently aims at measuring the impact of the examined policy by using some performance metrics, such as cost, holding time, arrival rates etc. Both analytical models (Higginson and Bookbinder, 1995; Gupta and Bagchi, 1987; Çetinkaya and Lee, 2000; Bookbinder and Higginson, 2002) and simulation models (Closs and Cook, 1987; Higginson and Bookbinder,

1994) are frequently studied to indicate the best performing consolidation policy with different criteria and factors. So, several criteria have been considered while examining different consolidation policies; for instance, Çetinkaya et al. (2014) used service based criteria other than cost-based criteria of the previous literature and Baykasoglu and Kaplanoglu (2011) proposed multi-agent based decision making approach to make good decisions in a dynamic business environment. Çetinkaya (2005) provides a detailed review on shipment consolidation and inventory decisions.

2.1.2 Spatial Consolidation

Spatial consolidation occurs on “*space*” (Harks et al., 2014), in other words geographic characteristics of items, and consolidates small items by creating vehicle routes or paths. The main focus of spatial consolidation research is similar to shortest-route problems as well as network design problems with consolidation terminals (Min and Cooper, 1990). The early studies on temporal consolidation focus on consolidation of LTL orders, which usually weight between a few hundred and a few thousand kilograms (Jarrah et al., 2009). Powell and Sheffi (1983) initially studied consolidation in the LTL industry as fixed charge network design problem, and followed by several studies (Powell, 1986; Sheffi, 1986; Powell and Sheffi, 1989). Additionally, main aim in most of the early studies is to make a comparison between networks with direct and consolidated distribution strategies, and analysing the cost effectiveness (Daganzo, 1988), advantages and disadvantages of consolidation (Hall, 1987a; Campbell, 1990) as well as optimization on network flow (Hall, 1987b; Min, 1996). Recent literature discuss spatial consolidation as a form of network flow or network design problem, vehicle routing problem, in which both spatial and temporal aspects of the problem are concerned.

2.2 Transportation Planning Problems

Transportation planning problems are classified into three classes (Crainic and Laporte, 1997; Crainic, 2000, 2003); (i) strategic, (ii) tactical and (iii) operational problems. Strategic planning problems refer to the long term decisions or decisions that require vast amount of investment; such as, facility location or fleet size decisions. Tactical decision problems take account of mid-term decisions, like assigning freight through network. Operational planning problems relate to the short-term decisions, such as scheduling, crew assignment etc. As long as the operations in SCD-TT are subject to short-term decisions, it relates to the operational planning problems.

In this section we will focus on the network design problems, vehicle routing problems. Although network design problems are considered as tactical planning problems (Crainic, 2000), the vast amount of researches on service network design problems, especially the networks with hybrid delivery structure is promising, thus we firstly examine the related service network design problems and proposed solution methodologies in the literature. As for the operational planning problems, SCD-TT relates to Vehicle Routing Problem (VRP), thus we survey the associated literature. Finally, we investigate the literature on intermediate facilities on both problems, network design and VRP, in order to identify the role of transshipment terminals in SCD-TT.

2.2.1 Network Design Problems

Network Design Problems are frequently studied to tackle with the planning of transportation and distribution systems (Crainic, 2000). Since they are generalization of location formulations (Crainic and Laporte, 1997), network design

models aim to choose links in a network along with capacity constraints, and decide features of network, such as; frequency of the service (Powell and Sheffi, 1983; Crainic, 1984) , traffic assignment along the routes (Attanasio et al., 2007), consideration of service level (Jarrah et al., 2009). The network design models are applied to several industries and mostly on express shipment delivery problems over long distances (Barnhart and Schneur, 1996; Kim et al., 1999), liner shipping network design problems (Liu et al., 2014), problems arise in railway like empty car distribution problems (Marin and Salmerón, 1996), multimodal transportation (Crainic and Rousseau, 1986), and operations of LTL carriers (Powell and Sheffi, 1983; Powell, 1986). As exact solution methods are impractical for large instances, several heuristics have been applied, such as, genetic algorithm (Cunha and Silva, 2007), tabu search (Estrada and Robusté, 2009), ant colony optimization (Barcos et al., 2010). Decomposition and column generation are also frequently applied to deal with large instance sets (Powell and Sheffi, 1989; Barnhart and Schneur, 1996; Irnich, 2002; Andersen et al., 2011). For a detailed survey on network design problems arise in transportation, the reader may refer to Crainic (2000) and Wieberneit (2008).

2.2.2 Vehicle Routing Problems

VRP finds a set of routes to deliver a given set of customer orders (Fischetti et al., 1994), while the main objective of the problem may be the minimization of the total cost (Fischetti et al., 1994), as well as total distance travelled (Desrosiers et al., 1986), total route duration (Savelsbergh, 1992), or the fuel consumption (Kuo, 2010). Recently, many extensions of VRP have been studied in the literature. As a well-known problem, capacitated VRP (Dantzig and Ramser, 1959) minimizes the total cost of transportation, and has only capacity constraint to

be satisfied. Studies on VRP with time windows (Solomon, 1984; Berger et al., 2003; Khebbache-Hadji et al., 2013) generally assume that the collection or deliveries of customer orders should be done within specified time slots, therefore the routes has to be constructed accordingly. Routing problems with loading constraints (Gendreau et al., 2006) extend VRP by including bin-packing constraints. As long as freight forwarders use contracted vehicles and do not deal with the returns of vehicle to origin destination, the problem of freight forwarders is considered as Open-VRP (Sariklis and Powell, 2000; Fu et al., 2005) in the literature. There are also studies that include the pick-up and delivery (Savelsbergh, 1992), precedence requirements to visit customers (Dumas et al., 1991), and using cross-dock terminals along the route (Lee et al., 2006).

2.2.3 Intermediate Facilities in Transportation Planning Problems

In general, an intermediate facility is a location, where products or raw materials are processed, sorted, or consolidated for the transportation between a set of origins to a set of destinations. As an important entity in transportation planning problems, an intermediate facility has various types accordingly with the role it plays. Guastaroba et al. (2016) examined the roles of intermediate facilities considered in the literature, and categorize 3 different roles, which are cross-docks, in-transit merge centres, intermediate warehouses called distribution centres. Similarly, Higginson and Bookbinder (2005) also listed the various roles that distribution centres have along the supply chain as an intermediate facility. They define distribution centre as a warehouse, where storage function is limited or non-existent. In this respect, they define 6 major roles of distribution centres, which are; make-bulk/break-bulk consolidation centre, cross-dock, transshipment

facility, assembly facility, product-fulfilment centre, and depot for returned goods.

Although these roles are interrelated, and both Higginson and Bookbinder (2005) and Guastaroba et al. (2016) referred to cross-docks as a transshipment centre, Higginson and Bookbinder (2005) made a clear distinction between cross-docks and transshipment centres. Cross-docking is a more customer-oriented form of transshipment, which aims at improving customer service by providing continuous flow of items in minimum time (Gümüş and Bookbinder, 2004). Whereas transshipment is a process in which items change vehicles or transportation modes (Beuthe and Kreutzberger, 2008), and has a major role in carrier perspective. In general terms, transshipment centres provide service to end-of-lines where local delivery is necessary as well as possibility to make changes vehicles or transportation modes.

The advantages of intermediate facilities basically relate to the enhancement of provided services. For instance, cross-docks provides faster product flow, cuts in inventory and improved customer service (Higginson and Bookbinder, 2005). From shippers perspective, transshipment centres attend to utilize vehicle routes by decreasing the number of stops, while increasing the number of items delivered with the same vehicle (Daganzo, 1988). In this sense, environmental concerns related to vehicle traffic, especially semi-trailer trucks used in long-haul transportation, decreases with the use of intermediate facilities to urban areas (Higginson and Bookbinder, 2005). In contrast, the use of intermediate facilities increases the total cost of transportation, including the costs of handling and delivery to final destination. Also, there is an increased risk of damage and loss due to handling of items several times, and transit time required during in-transit operations increases total transit time, thus the likelihood of late deliveries. Moreover, the distribution networks with intermediate facilities requires complex planning

and coordination mechanisms to ensure continuous flow of transported materials (Guastaroba et al., 2016).

Considering aforementioned advantages and disadvantages, there is a body of literature on intermediate facilities along the distribution network. When the value of cross-docking as an intermediate facility is concerned, (Galbreth et al., 2008) addressed to three research questions and specified that cross-docking items with higher holding costs, stable and low demand is more favourable. (Gümüş and Bookbinder, 2004) also revealed that direct deliveries are usually optimal only when demands are near full truck capacity. The applied methods are also varied and progressed with the developments in computational technologies. Both optimization models, exact and metaheuristic algorithms are applied to design the network with transshipment centres. For a detailed survey on intermediate facilities in transportation problems, the reader may refer to Guastaroba et al. (2016).

2.3 Literature Gap

In this chapter, we review the existing literature related to SCD-TT in planning perspective. Respectively, temporal consolidation strategies mainly focus on the “time” or “quantity” to dispatch a vehicle; thus, the problem mostly relates to the inventory holding cost/time and customer service level. However, main concern of SCD-TT is basically related to both spatial and temporal attributes of the items in three aspects: (i) route, (ii) delivery type and (iii) time. Moreover, the nature of the problem is deterministic, so the attributes related to the items to be consolidated are predetermined. Then, the question of SCD-TT becomes “*how*” to consolidate items in terms of vehicles and time.

The practical problem has distinctive characteristics from the problems studied in the literature; especially in terms of delivery network, route and cost structure. As for the delivery network, the role of intermediate facilities in the practical problem is to provide end-of-line delivery, thus may be handled on the carrier strategy to provide additional services. In this respect, we will refer to intermediate facilities in SCD-TT as “*transshipment terminals*” throughout the text, depending on the clear description of Higginson and Bookbinder (2005). Moreover, the hybrid delivery structure, which allows stops at both destinations and transshipment terminals with the same vehicle, requires further attention (Guastaroba et al., 2016). Regarding the cost structure, routes defined by the farthest delivery point and ensuring fixed routing cost is not frequently studied in the literature. To the best of our knowledge, Koca and Yildirim (2012) addressed to such a cost structure.

With respect to transportation planning problems, both SCD-TT and network design problems involve LTL operations, freight consolidation decisions, and make intensive use of consolidation operations (Crainic, 2000); yet, there are differences among two problems. Moreover, costs in networks are associated with node-to-hub, hub-to-node or inter-hub basis, but SCD-TT has a fixed cost associated with the routes. In real life practices, there are several intermediate facilities in large international networks (Guastaroba et al., 2016), thus integration of one or more intermediate facilities, such as consolidation centres, along the same flow requires further attention (Wieberneit, 2008).

VRP optimizes the vehicle routes considering the capacities, and extensions of VRP may offer solution to the time and loading aspects of the practical problem. Although SCD-TT may be formulated as open VRP with time-windows and hybrid delivery structure along the transshipment centres, the cost structure of VRP and SCD-TT are different. VRP has a node-to-node structure, whereas

SCD-TT has pre-defined routes or routes defined with the farthest destination. Respectively, the cost structure of VRP is node-to-node, while SCD-TT has fixed costs for routes. In this respect, we do not model the problem as VRP extension.

With respect to the aforementioned gaps in the literature, we aim to provide a different approach to shipment consolidation and dispatching problem by defining a new problem, which has a hybrid delivery network with transshipment terminals and routing costs, which is adopted by the freight forwarders to simplify decision making, as it is described in the Section 1.1.

Chapter 3

Shipment Consolidation and Dispatching with Transshipment Terminals and Fixed Routes

3.1 Introduction

In this chapter, we will discuss the real-life problem, in which the fixed cost of a vehicle is defined by the annual contracts made with the sub-contracted haulers, namely Shipment Consolidation and Dispatching with Transshipment Terminals and Fixed Routes SCD-TTFR. In the annual contracts, the routes are predefined by zones or regions. Such predefined routes also include the possible stopping points, which doesn't require excessive deviations from the main path to the travelled zone/region. The price of the routes are also identified in the annual contracts with respect to the farthest destination in that zone/region, even if the vehicle does not visit that farthest destination. Commonly, the zones/regions

in the annual contracts include cities having approximate distance to the origin, eventually the cost of visiting that zone is more or less the same with small deviations. Therefore, the cost of visiting a city in a zone gives a good approximate cost instead of real cost of visiting that city.

In the recent literature, Ghiani et al. (2004) proposed the problem of a manufacturer, where set of orders has to be delivered to its destination using predefined routes over a planning horizon. The problem considers consolidation of orders in vehicles with fixed routes regarding release date and deadline, where orders can either be shipped by a common carrier or by rented/owned trucks. On the study of Ghiani et al. (2004), Attanasio et al. (2007) proposed an algorithm, which disregards capacity constraints and add cuts to eliminate infeasible solutions. The problem has similarities with SCD-TTFR in terms of order structure and costs associated with the routes. However, the structure of routes are different in two problems; SCD-TTFR has extended routes with possible stops along the routes, while the problem of Ghiani et al. (2004) and Attanasio et al. (2007) has fixed stopping points along routes. Moreover, the number of stops for a vehicle is limited and each stop is associated with an extra cost. Additionally, there is only one dimension (weight) included in the problem of Ghiani et al. (2004) and Attanasio et al. (2007), whereas SCD-TTFR have three dimensions (weight, volume and loading meter). Delivery structure is also different as SCD-TTFR allows delivery from transshipment terminals. In this context, we propose an extension of the mathematical model proposed by Ghiani et al. (2004).

This chapter is organised as follows. We firstly define the assumptions of the problem, propose the mathematical model and its extensions. Secondly, we examine the computational complexity, and introduce two lower bound algorithms. Then we propose Variable Neighborhood Search algorithm, and examine the per-

formance of both mathematical models and upper bound algorithm on randomly generated instances.

3.2 Problem Statement and Mathematical Model

In this section, we state the problem, develop the mathematical model, and examine the computational complexity. Based on the real life problem, we define the SCD-TTFR problem with the following assumptions;

- Information on orders, such as dimensions, destination, release date and deadline, are deterministic and initially known.
- The orders can be delivered either on wheels or by using a transshipment terminal. Each transshipment decision implies a cost proportional to the size of the order.
- Routes are previously defined, and possible stopping points on each route are known.
- The costs of routes are fixed, defined with respect to the farthest destination along the route.
- Fixed costs of routes includes a limited number of stops, and after that number, each additional stop incurs an extra cost of stopping upto maximum number of stops which cannot be exceeded.
- The number of stops a vehicle can do is limited, hence, the delivery duration is not affected by the number of stops.

3.2.1 Mathematical Model

With respect to the above assumptions, we formulate the mathematical model for SCD-TTFR as follows;

Indices and Sets

K	Set of orders, $k \in K$
I	Transshipment Terminals, $i \in I$
J	Destinations, $j \in J$
T	Days in planning horizon, $t \in T$
N	Homogeneous trucks, $n \in N$
R	Set of routes, $r \in R$
A_r	Possible stopping points along route r , $A_r \subset J, \forall r$
B_r	transshipment terminal points along route r , $B_r \subset I, \forall r$
H_r^t	Set of orders that may depart on day t by direct delivery to its destination j on route r , where $H_r^t \subset K \forall r, t$
G_{ir}^t	Set of orders that may depart on day t , by delivering from transshipment terminal i on route r , where $G_{ir}^t \subset K \forall i, r, t$

Parameters

Details of orders, $\forall k$;

v_k	Total volume
w_k	Total weight
l_k	Total length
r_k	Release day
d_k	Deadline
p_k	destination, where $p_k \in J$

Vehicle capacities;

ν	volume capacity
-------	-----------------

γ	weight capacity
δ	length capacity
τ_{jr}	Transit time to destination j on route r , $\forall j, r$
λ_{ri}	Transit time to transshipment terminal i on route r , $\forall i, r$
ρ_{ij}	Transit time from transshipment terminal i to destination j , $\forall i, j$
μ	Limit on additional number of stops
ϕ	Number of stops included in the fixed cost, where $1 \leq \phi \leq \mu$
f_r	Fixed cost of route r , $\forall r$
c_{ik}	Cost of transshipping order k from transshipment terminal i $\forall i, k$
α	Additional cost of each stop after ϕ stops
M	Very big number

Regarding above parameters, note that;

$$k \in G_{ir}^t \text{ if } r_k \leq t \leq d_k - \lambda_{ri} - \rho_{ipk}, \forall k, i, r, t$$

$$k \in H_r^t \text{ if } r_k \leq t \leq d_k - \tau_{pkr}, \forall k, r, t$$

Decision variables:

$$y_{kr}^{tn} = \begin{cases} 1 & \text{if order } k \text{ is delivered directly by } n^{\text{th}} \text{ vehicle with route } r \text{ on day } t \\ 0 & \text{otherwise} \end{cases}$$

$$x_{kir}^{tn} = \begin{cases} 1 & \text{if order } k \text{ is shipped via transshipment terminal } i \text{ by } n^{\text{th}} \text{ vehicle with} \\ & \text{route } r \text{ on day } t \\ 0 & \text{otherwise} \end{cases}$$

$$\kappa_{jr}^{tn} = \begin{cases} 1 & \text{if destination } j \text{ is assigned to } n^{\text{th}} \text{ vehicle with route } r \text{ on day } t \\ 0 & \text{otherwise} \end{cases}$$

$$\theta_{ir}^{tn} = \begin{cases} 1 & \text{if transshipment terminal } i \text{ is assigned to } n^{\text{th}} \text{ vehicle with route } r \text{ on day } t \\ 0 & \text{otherwise} \end{cases}$$

$$s_r^{tn} = \begin{cases} 1 & \text{if } n^{\text{th}} \text{ vehicle with route } r \text{ is departed on day } t \\ 0 & \text{otherwise} \end{cases}$$

$$u_r^{tn} = \text{Number of stops on } n^{\text{th}} \text{ vehicle with route } r \text{ on day } t$$

Minimize;

$$Z = \sum_{r \in R} \sum_{t \in T} \sum_{n \in N} \left[f_r s_r^{tn} + \alpha u_r^{tn} + \sum_{i \in I} \sum_{k \in K} c_{ik} x_{kir}^{tn} \right] \quad (3.1)$$

Subject to;

$$\sum_{i \in B_r} \sum_{k \in G_{ir}^t} v_k x_{kir}^{tn} + \sum_{\substack{k: k \in H_r^t \\ p_k \in A_r}} v_k y_{kr}^{tn} \leq \nu s_r^{tn} \quad \forall r, t, n \quad (3.2)$$

$$\sum_{i \in B_r} \sum_{k \in G_{ir}^t} w_k x_{kir}^{tn} + \sum_{\substack{k: k \in H_r^t \\ p_k \in A_r}} w_k y_{kr}^{tn} \leq \gamma s_r^{tn} \quad \forall r, t, n \quad (3.3)$$

$$\sum_{i \in B_r} \sum_{k \in G_{ir}^t} l_k x_{kir}^{tn} + \sum_{\substack{k: k \in H_r^t \\ p_k \in A_r}} l_k y_{kr}^{tn} \leq \delta s_r^{tn} \quad \forall r, t, n \quad (3.4)$$

$$\sum_{r \in R} \sum_{i \in B_r} \sum_{n \in N} \sum_{t: k \in G_{ir}^t} x_{kir}^{tn} + \sum_{\substack{r: r \in R \\ p_k \in A_r}} \sum_{n \in N} \sum_{t: k \in H_r^t} y_{kr}^{tn} = 1 \quad \forall k \quad (3.5)$$

$$\sum_{k \in G_{ir}^t} x_{kir}^{tn} \leq M \theta_{ir}^{tn} \quad \forall i, r, t, n \quad (3.6)$$

$$\sum_{k \in H_r^t} y_{kr}^{tn} \leq M \kappa_{jr}^{tn} \quad \forall j, r, t, n \quad (3.7)$$

$$\kappa_{jr}^{tn} \leq s_r^{tn} \quad \forall j, r, t, n \quad (3.8)$$

$$\theta_{ir}^{tn} \leq s_r^{tn} \quad \forall i, r, t, n \quad (3.9)$$

$$\sum_{i \in B_r} \theta_{ir}^{tn} + \sum_{j \in A_r} \kappa_{jr}^{tn} \leq \phi + u_r^{tn} \quad \forall r, t, n \quad (3.10)$$

$$u_r^{tn} \leq \mu \quad \forall r, t, n \quad (3.11)$$

$$x_{kir}^{tn}, y_{kr}^{tn}, s_r^{tn}, \kappa_{jr}^{tn}, \theta_{ir}^{tn} \in \{0, 1\} \quad \forall k, i, j, r, t, n \quad (3.12)$$

$$u_r^{tn} \geq 0 \quad \forall r, t, n \quad (3.13)$$

The objective function (3.1) minimizes the total cost of shipping all orders, including fixed costs of routes and additional stops, and the cost of transshipping orders from transshipment terminals to their final destination. Constraints (3.2), (3.3) and (3.4) ensure that the total volume, weight and loading meter of orders, which are loaded on the vehicle $n \in N$ with route $r \in R$, departing on day $t \in T$, cannot exceed the capacity of that vehicle. The constraints also ensure that the destination of order k is a stop along route r , and due date is satisfied with respect to release date and transit time required to deliver that order. Constraint (3.5) ensures that each order $k \in K$ is shipped and delivered either on wheels or via transshipment terminals. Constraint (3.6) ensures that the transshipment terminal $i \in I$ is assigned to vehicle $n \in N$ with route $r \in R$, departed on day $t \in T$ only if any order k in that vehicle is shipped via transshipment terminal i . Constraint (3.7) ensures that the destination $j \in J$ is assigned as a stopping point to vehicle $n \in N$ with route $r \in R$, departed on day $t \in T$ only if order k in that vehicle is delivered directly. Constraint (3.8) ensures that the destination $j \in J$ is assigned to the vehicle $n \in N$ with route $r \in R$, departed on day $t \in T$. Constraint (3.9) ensures that the transshipment terminal $i \in I$ is assigned to the vehicle $n \in N$ with route $r \in R$, departed on day $t \in T$. Constraints (3.10) and (3.11) guarantee that the limit on number of stops is not exceeded. Constraint (3.12) states that decision variables x_{ki}^{rtn} , y_{kr}^{tn} , θ_{ir}^{tn} , κ_{jr}^{tn} and s_r^{tn} are binary variables, and constraint (3.13) guarantees that u_r^{tn} is a positive variable.

We improved the above mathematical model by redefining the M values, and adding two types of valid inequalities. We explain these enhancements below;

1) *Symmetry Breaking Constraints:* As set N creates symmetric solutions, we have to prevent the use of higher indexed vehicle by skipping the lower indexed vehicle. Therefore, we add symmetry breaking constraints that ensure the use of vehicles in sequence, such that;

$$s_r^{tn} \geq s_r^{t(n+1)} \quad \forall r, t, n = \{1, 2, \dots, |N - 1|\} \quad (3.14)$$

2) *Defining an Upper Bound for Big M :* The very big number, M used in constraint sets (3.6) and (3.7), in equations may be reduced as the number of orders assigned to a destination or transshipment terminal cannot exceed the number of orders that can be delivered using route $r \in R$ within the specified time-windows. Therefore, we simply count the number of orders that can be delivered through transshipment terminal i by satisfying the required time windows, and redefine M for equations (3.6) and (3.7) as Mx_{irt} and My_{irt} respectively;

$$xm_{ir}^t = \sum_{\substack{k \in G_{ir}^t \\ i \in B_r}} 1 \quad \forall i, r, t$$

$$ym_{jr}^t = \sum_{\substack{k \in H_r^t \\ p_k \in A_r}} 1 \quad \forall j, r, t$$

Therefore, the equations (3.6) and (3.7) are restated as;

$$\sum_{k \in G_{ir}^t} x_{kir}^{tn} \leq xm_{ir}^t \theta_{ir}^{tn} \quad \forall i, r, t, n \quad (3.15)$$

$$\sum_{k \in H_r^t} y_{kr}^{tn} \leq ym_{jr}^t \kappa_{jr}^{tn} \quad \forall j, r, t, n \quad (3.16)$$

3) *Direct Relationship between order assignment and vehicle departure:* Although constraint sets (3.2), (3.3) and (3.4) ensures departure of vehicle n on route r on day t if an order is assigned to that vehicle, we added a constraint that defines a

direct relation between order assignment to destination or transshipment terminal and vehicle departure decision. Such a constraint that creates a tighter bound is stated as;

$$y_{kr}^{tn} + \sum_{i \in I} x_{kir}^{tn} \leq s_r^{tn} \quad \forall k, r, t, n \quad (3.17)$$

With respect to above mentioned additional constraints, throughout the text, we will mention MM1 as the basic mathematical model, and MM2 as the enhanced mathematical model. The constraints (3.2), (3.3), (3.4), (3.5), (3.8), (3.9), (3.10), (3.11), (3.12) and (3.13) are common in both mathematical models. MM1 uses constraints (3.6), (3.7) and (3.14); whereas MM2 uses constraints (3.14)-(3.17).

3.2.2 Computational Complexity

In this section, we examine the computational complexity of SCD-TTFR by showing that a special case of SCD-TTFR is a multi-dimensional bin-packing problem.

Theorem 1 *SCD-TTFR is NP-hard in strong sense.*

Proof 1 *Consider a special case of SCD-TTFR with the following assumptions;*

- *There is only one route in set of routes, $R = \{1\}$. Thus, fixed cost of route R can be denoted as f ,*
- *There is only one destination $J = \{1\}$, and thus $A_r = \{1\}$,*
- *There is only one day in planning horizon $T = \{1\}$,*
- *There is no transshipment terminals $I = \{1\}$, thus $B_r = \emptyset$, and there is no transit time or cost associated with transshipment terminals. Moreover,*

there is no need to make a decision if the order $k \in K$ will be delivered via transshipment terminals. Then, $x_{kir}^{tn} = 0$ and $\theta_{ir}^{tn} = 0$,

- There is no limit on the number of stops. Thus $u_r^{tn} = 0$,
- The destination and release date of each order are identically the same, $p_k = 1 \quad \forall k \in K$ and $r_k = 1 \quad \forall k \in K$,
- The orders can be delivered within the same day, thus, transit time to destination equals 0, $\tau_{rp_k} = 0$, Therefore, deadline of each order $k \in K$ will equal to 1, $d_k = 1 \quad \forall k \in K$.

In this context, the special case of SCD-TTFR can be formulated as follows;

$$\begin{aligned}
 & \text{Minimize} && \sum_{n \in N} f \cdot s_n \\
 & \text{Subject to;} && \sum_{k \in K} y_{kn} v_k \leq \nu s_n && \forall n \\
 & && \sum_{k \in K} y_{kn} w_k \leq \gamma s_n && \forall n \\
 & && \sum_{k \in K} y_{kn} l_k \leq \delta s_n && \forall n \\
 & && \sum_{n \in N} y_{kn} = 1 && \forall k \\
 & && y_{kn}, s_n \in \{0, 1\} && \forall k, n
 \end{aligned}$$

Above problem is a multi-dimensional Bin-Packing problem, which is a NP-hard problem in strong sense (Coffman Jr et al., 1996) and a special case of SCD-TTFR. Thus, SCD-TTFR is NP-hard in strong sense.

3.3 Lower Bound Algorithms

In this section, we provide two lower bound algorithms both work iteratively. Initially the algorithms relaxes a set of constraints. At each iteration, the algorithm solves the relaxed problem optimally (or reports the best feasible solution found within a time limit). The algorithm analyzes the solution and adds a subset of the relaxed constraints that are violated. Then it solves the updated problem. The algorithm stops after a certain computation time or there exists no violated constraints. The first lower bound algorithm (LB1) relaxes the integrality constraints, whereas the second lower bound algorithm (LB2) relaxes the capacity constraints.

3.3.1 Lower Bound 1: Relax Integrality Constraints

For the Lower Bound 1 Algorithm (LB1), we relax the integrality constraints and solve the problem; then, by analyzing the solution of the relaxed problem, we force some of the decision variables to be binary, and resolve the restated model. We continue the same operation iteratively until the stopping condition is satisfied.

We formulate the relaxed SCD-TTFR as **P1**;

$$\begin{aligned} \text{Minimize} \quad & Z = f(x) \\ \text{Subject to} \quad & Ax \leq B \\ & 0 \leq x \leq 1 \end{aligned}$$

Where x is the decision variable vector, $x \in X$, and X is a feasible solution set. LB1 algorithm redefines x as a vector composed of continuous and binary variables such that $x = [x_c, x_b]$, where $x_b \in Bin$ represent the binary components of the solution vector. The set Bin is the set of indices such that x_b must be binary.

The redefined problem **P2** is as follows;

$$\begin{aligned}
& \text{Minimize} && Z = f(x) \\
& \text{Subject to} && Ax \leq B \\
& && x = [x_c, x_b] \\
& && 0 \leq x_c \leq 1, \quad c \in X \setminus Bin \\
& && x_b \in Bin, \quad b \in Bin
\end{aligned}$$

LB1 algorithm solves **P2** iteratively, finds a solution $x' = [x'_c, x'_b]$, computes $[lo, up]$ values by using a specific approach. Assuming that the values of decision variables that are close to 0.5 creates difficulty in each iteration, the algorithm finds the highest value which is lower than 0.5, and subtracts a 10% of margin to define lo for each decision variable vector ($lo = 0.9 \max_{(c:x'_c \leq 0.5)}(x'_c)$). Similarly, the algorithm finds smallest value which is greater than 0.5, and adds a 10% of margin to define up ($up = 1.1 \min_{(c:x'_c > 0.5)}(x'_c)$). Respectively, algorithm updates Bin as follows;

$$\text{if } lo \leq x'_c \leq up \Rightarrow Bin = Bin \cup c$$

The algorithm stops when the decision variables of SCD-TTFR corresponding to the orders are all binary, or time limit is exceeded.

Algorithm 3.1 Lower Bound 1

Initialization: Relax integrality constraints as $0 \leq x \leq 1$, and redefine decision variable vector x ; such that, $x = [x_c, x_b]$, where $x_b \in \{0, 1\}$, $b \in Bin$ and $0 \leq x_c \leq 1$, $c \in X \setminus Bin$. Initially $Bin = \emptyset$. Define time limit.

Main Step: Until time limit is exceeded, or $|Bin| = |K|$, repeat following steps;

Step 1: Solve redefined problem **P2** and obtain a solution x' , where $[x'_c, x'_b]$

Step 2: Update $[lo, up]$

Step 3: Update Bin , if $lo \leq x'_c \leq up$

3.3.2 Lower Bound 2: Relax Capacity Constraints

Lower Bound 2 Algorithm (LB2) removes the capacity constraints, (3.2), (3.3) and (3.4), and applies an iterative algorithm that adds cuts, where the total amount of orders in a vehicle exceeds its capacity. We continue in the same manner iteratively until the stopping condition is satisfied.

After removing the capacity constraints, we add some valid inequalities. Even though there are no capacities set to the problem, a feasible solution must hold that the number of vehicles are more than the total amount of orders divided by the vehicle capacity; thus, a valid inequality may be such that;

$$\sum_{r \in R} \sum_{t \in T} \sum_{n \in N} s_r^{tn} \geq \max \left(\left\lceil \sum_{k \in K} \frac{v_k}{\nu} \right\rceil, \left\lceil \sum_{k \in K} \frac{w_k}{\gamma} \right\rceil, \left\lceil \sum_{k \in K} \frac{l_k}{\delta} \right\rceil \right) \quad (3.18)$$

If we assume that $timecap_t$ is the maximum number of orders that can depart on day t in the same vehicle (regardless of the capacity), with respect to the release date, deadline and minimum transit time required for delivery of the order, any feasible solution should hold that the number of orders in a vehicle must not be more than $timecap_t$;

$$timecap_t = \max_{r \in R} \left(\left| \left(\bigcup_{i \in I} G_{ir}^t \right) \cup H_r^t \right| \right) \quad (3.19)$$

$$\sum_{i \in I} \sum_{k \in K} x_{kir}^{tn} + \sum_{k \in K} y_{kr}^{tn} \leq timecap_t \quad \forall r, t, n \quad (3.20)$$

Let EC be the set of vehicles that total amount of orders in it exceeds the capacity, and λ_e is the set of orders in vehicle $e \in EC$, where $k \in \lambda_e$. Then, we added cuts to the relaxed problem with no capacity constraints, which are formulated as

follows;

$$\sum_{i \in I} \sum_{k \in \lambda_e} x_{kir}^{tn} + \sum_{k \in \lambda_e} y_{kr}^{tn} \leq |\lambda_e| - 1 \quad \forall e, r, t, n \quad (3.21)$$

The LB2 algorithm iteratively solves the problem, updates set EC , and adds cuts to the problem. If there are no vehicles with excess capacity; hence, there are no cuts to be added, or time limit is exceeded, the algorithm stops.

Algorithm 3.2 Lower Bound 2

Initialization: Relax capacity constraints, and add valid inequalities; which ensures that, the number of vehicles is more than the total amount of orders divided by the vehicle capacity, and the number of orders in a vehicle must not be more than $timecap_t$, where $timecap_t = \max_{r \in R} \left(\left| \bigcup_{i \in I} G_{ir}^t \cup H_r^t \right| \right)$.

Main Step: Until time limit is exceeded or there are no vehicles with excess capacity, repeat following steps;

Step 1: Update the set of vehicles with excess capacity, $e \in EC$

Step 2: Update the number of orders in the vehicles existed in EC , $\lambda_e, \forall e \in EC$

Step 3: Add cuts, where the number of orders in the vehicle e is less than $(\lambda_e - 1)$

3.4 Variable Neighborhood Search

In this section, we propose a VNS algorithm to the SCD-TTFR problem. VNS is a local search algorithm with a systematic change of neighborhood in order to escape local minima (Mladenović and Hansen, 1997). A local search heuristic usually uses one neighborhood structure, instead VNS uses more than one neighborhood as global minimum is a local minimum with respect to all possible neighborhood structures (Hansen and Mladenović, 2001). Increasingly far neighborhoods from the incumbent solution are explored, and the algorithm moves to another solution if the solution is better than the incumbent solution.

Algorithm 3.3 represents the proposed VNS algorithm to SCDP-CDs problem. Assuming that x is the initial solution, we choose a solution x' at random in the first neighborhood, and apply enumeration algorithm around the chosen solution. If the local minimum x'' obtained after enumeration is better than the incumbent solution ($f(x'') < f(x)$), then the search is removed to recently obtained solution point $x \leftarrow x''$. Otherwise, the algorithm proceeds to the next neighborhood, and returns back to the first neighborhood, after all neighborhoods are explored, until a stopping condition is satisfied. We define three stopping conditions: (i) maximum CPU time, (ii) maximum global iteration number, and (iii) maximum outer iteration number.

Algorithm 3.3 Variable Neighborhood Search

Initialization: Find an initial solution x , set the neighborhood structures $N_k, k = 1, \dots, k_{max}$, define the stopping condition.

Main Step: Until $k = k_{max}$, repeat the following steps;

(a) **Shaking:** Generate a random feasible point x' from k^{th} neighborhood of x , $x' \in N_k(x)$.

(b) **Local Search:** Apply enumeration algorithm around x' and find a local minimum x'' .

(c) **Move or Not:** If local optimum x'' is better than the incumbent solution x , move to this local optimum ($x \leftarrow x''$) and continue search with N_1 , set $k \leftarrow 1$; otherwise set $k \leftarrow k + 1$.

In order to obtain an initial solution, we modify the First Fit Decreasing Algorithm (FFD), which is a simple and commonly used construction algorithm for bin-packing problems. The algorithm sorts the orders in decreasing chargeable weight, packs the largest order into the first available vehicle if and only if it is feasible in terms of capacity and transit time, and stops when all orders are packed in a vehicle.

In *shaking* procedure, we use five operators to construct neighborhood structures N_k , where k denotes the neighborhoods, $k = 1, \dots, k_{max}$. Here, we apply

each operator 3 times in different levels, thus we have 24 neighborhoods in total ($k_{max} = 24$) (Table 3.1). The operators are; move, swap, perturbation, remove and regeneration; and defined as:

Move: Move randomly selected η different orders to the randomly selected vehicles if move action is feasible. If no such move action exists, skip to the next neighborhood.

Swap: Randomly select η pair of orders (η_1, η_2) , such that $\eta_1 \neq \eta_2$. Swap orders if both orders fit to their corresponding vehicle and the swap action is feasible. If no such swap action exists, skip to the next neighborhood.

Perturbation: Remove all orders in randomly selected η vehicles, create new feasible vehicles with the remaining orders, place removed orders to recently created vehicles randomly. If no feasible solution exists, or all removed orders cannot be placed, skip to the next neighborhood.

Remove: Remove η vehicles with the most residual capacity, and place all orders in $T - \eta$ vehicles, where T is the number of vehicles in the incumbent solution, by using Best-Fit-Decreasing (BFD) algorithm. If all orders cannot be placed in $T - \eta$ vehicles, skip to the next neighborhood.

Regeneration: Create a new solution with T vehicles, where T is the number of vehicles in the incumbent solution, by using BFD algorithm. If all orders cannot be placed in $T - \eta$ vehicles, skip to the next neighborhood.

We control the randomness during shaking procedure in order to increase possible actions regarding three assumptions; (i) small orders have relatively higher probability to be placed in other vehicles, (ii) a change made in relatively expensive vehicles have a higher probability to provide a decrease in total cost, and

Table 3.1: Neighborhood Set N_k

k	Operator	Level
1-2-3	Move	$\eta = 1$
4-5-6	Move	$\eta = 2$
7-8-9	Swap	$\eta = 1$
10-11-12	Swap	$\eta = 2$
13-14-15	Perturbation	$\eta = 1$
16-17-18	Perturbation	$\eta = 2$
19-20-21	Remove	$\eta = 1$
22-23-24	Regeneration	-

(iii) selecting orders which are ready on the same day have higher probability to provide a feasible solution. Therefore, during *move* operation, we randomly select the orders by giving priority to relatively small orders placed in relatively expensive vehicles. In *swap* operator, the orders with small and similar in volume are swapped. In *perturbation* operator, the vehicles with relatively higher cost are selected. *Remove* and *regeneration* operations benefit from assumption (iii). Accordingly, BFD algorithm starts from a randomly selected day, packs all items that can depart on that day in the best fitting feasible vehicle, then skips to the next day.

In ***local search*** procedure, we apply an enumeration algorithm to the vehicles that altered during shaking procedure, and find the new costs of those vehicles (Algorithm 3.4). Considering that I is the set of transshipment terminals, where $I = 1, 2, 3, \dots, i$, there are 2^i alternative ways to deliver all orders in a vehicle. If we consider an example having two transshipment terminals, the alternatives are; (i) deliver all items directly, (ii) deliver some of the items by using only transshipment terminal A, (iii) deliver some of the items by using only transshipment terminal B, and (iv) deliver some of the items by using both transshipment terminals A and B. In this sense, enumeration algorithm finds the least cost of all alternatives,

and picks the best value.

Algorithm 3.4 Enumeration

Initialization: Define the set of delivery alternatives $A, A = 1, 2, \dots, a$; the set of altered vehicles in the incumbent solution $N = 1, 2, \dots, n_{max}$, and the associated cost of each vehicle as z_n ; set of routes, $R = 1, 2, \dots, r$, and sort routes in increasing cost.

Main step: Until, $n = n_{max}$, repeat the following steps;

(a) **Step 1:** For all r ; if $f_r \leq z_n$, stop and return z_n , otherwise find the cost of each alternative a , and select the least cost as z'_n .

(b) **Adim 2:** if $z'_n < z_n$, set $(z_n \leftarrow z'_n)$ and return z_n .

3.5 Experiments

In this section, we conduct computational experiments on randomly generated instances. We first present the generation of randomly generated instances regarding the real life practices. We explain the preliminary experiments in order to define the characteristics, which increases complexity and thus computation time. Finally, we provide the results of the experiments conducted on MM1, MM2, LB1, LB2 and UB by examining the performances.

3.5.1 Randomly Generated Instances

With respect to the real life assumptions, we randomly generated instances by controlling some parameters. Firstly, we generated instances with different number of orders, where I defines the number of orders in the instance. Note that, we only generate orders, and the set of routes, destinations, transshipment terminals and related transit times are known, and doesn't change for any instance set.

Dimensions of the Orders: Considering the bin-packing assumptions to the prob-

lem, we generated the dimensions of the orders in detail. The orders are defined in shape of pallets or boxes with 0.6 and 0.4 probabilities respectively. The items within each order are identical; however different orders have dissimilar dimensions. The items, both pallets and boxes, are in 4 different dimensions, which are assigned with equal probabilities. The weights of boxes are randomly assigned between 4 kg to 8 kg and the weights of pallets are assigned randomly assigned between 150 kg to 1000 kg. The number of items randomly generated between 1 to 650 if the items are boxes, and 1 to 17 if the items are pallets. Total volume, weight and length of each order is computed with respect to the dimensions and number of items.

Destination of the Orders: In order to enable orders to be assigned to same destination and control the density of destinations, we set an upper bound D on the total number of destinations in an instance, and define 3 different levels of D . We randomly select D destinations, then randomly assign orders to those destinations. In this respect, the levels of D doesn't guarantee that there are exactly D number of destinations in the instance, they only control the density of destinations. The small level of D assigns more orders to the same destination, while there are less orders going to the same destination for the bigger levels of D .

Release Date and Deadline of the Orders: We randomly generate the release date of orders, r_k , with respect to the planning horizon and transit time required to deliver to the destination of order. Random variable β is defined to determine the deadline of the orders. We define an upper bound, β_{up} , on β , in order to control the elasticity of orders' required time windows for delivery. We randomly select β , where where $0.10 \leq \beta \leq \beta_{up}$, and define deadline of the orders; such that, $d_k = (mintt_k + r_k)(1 + \beta)$, where $mintt_k$ is the minimum transit time required to

deliver order k . As long as β is close to 0.1, deadline for the order will be close to the release date, and vice versa. In this respect, we generated two levels of parameters. In level 1, the orders have no elasticity for delivery ($E = 0$), while there are some available days to await orders for departure in level 2 ($E = 1$).

Table 3.2: The levels of parameters controlling instance generation

Parameter	Level				
	1	2	3	4	5
I	10	20	30	50	100
E	0	1			
D	1	2	3		

Table 3.2 shows the levels of parameters controlling the instance generation. We generated 10 instances for each combination of parameters, and thus we have 300 instances in total.

3.5.2 Preliminary Experiments

In this section, we discuss the results of three preliminary experiments we conducted with two motivations; (i) to determine the best performing mathematical model, (ii) analyse the performance of mathematical models on instances with different parameters.

As the first preliminary experiment, we analyse the effects of three enhancements; reducing the parameter M , adding symmetry breaking constraint (14) and constraint on direct relation (17). We solve all possible combinations (Table 3.3) in a small instance set, and report the performances of all possible mathematical models. In Table 3.4, we compare the CPU time of each model to the best performing mathematical model for that instance, and examine the average per-

formances over tested instances. In this context, the mathematical model with all constraints (P8) outperformed all others. The next best two mathematical models are the ones with symmetry breaking and direct relationship constraints (P7), and with symmetry breaking constraint (P3) respectively. Although P7 is better than P3, their performances are very close. In accordance with these results, we prefer to use P3 (namely MM1) and P8 (namely MM2) in all experiments. Moreover, we used P8 as the base mathematical model for lower bound algorithms.

Table 3.3: All possible problem combinations

	P1	P2	P3	P4	P5	P6	P7	P8
Reduced M		✓			✓	✓		✓
Breaking Symmetry			✓			✓	✓	✓
Direct Relationship				✓	✓		✓	✓

Table 3.4: Performances of all possible problem combinations (%) in terms of CPU time

Instance	P1	P2	P3	P4	P5	P6	P7	P8
1	528.7	528.7	30.1	379.2	404.0	206.5	0.0	19.7
2	821.1	313.1	18.5	262.7	146.9	14.5	7.8	0.0
3	127.1	95.7	16.1	52.8	33.7	17.1	0.0	15.9
4	994.7	506.9	40.3	660.0	159.2	61.7	8.3	0.0
5	3411.6	1190.7	25.8	963.3	639.6	42.4	3.9	0.0
6	14046.6	6760.6	119.7	1490.1	2168.8	6.8	268.7	0.0
7	244.6	270.9	20.3	165.8	175.1	30.9	0.0	11.8
8	0.0	4.2	2.2	12.9	6.8	8.7	5.9	11.1
9	194.7	465.7	23.9	162.2	137.9	154.8	0.2	0.0
10	22.1	60.5	29.6	12.2	0.0	33.0	28.7	24.2
Average CPU	2039.11	1019.70	32.65	416.13	387.19	57.64	32.37	8.28

For the second preliminary experiment, we solve MM2 in GAMS 22.8 using Intel Xeon X5482 with 7.4 GHz and 10 GB RAM. We limit the computation time by 3600 sec, and terminate the run if the optimal solution cannot be found within

the limited time. We examine the effect of number of orders in an instance, (I), number of different destinations among the instance (D) and elasticity of orders' departures (E), which indicates the number of days that an order may wait until departure. We define 2 levels of difficulty for E and 3 levels for D .

Table 3.5: Results of preliminary experiments

I	E	D	#	# solved	Max CPU	Avg CPU
10	0	1	10	10	2.4	1.3
		2	10	10	1.4	1.1
		3	10	10	1.3	1.0
10	1	1	10	10	41.1	9.0
		2	10	10	370.6	39.2
		3	10	10	31.4	8.1
20	0	1	10	10	6.9	3.7
		2	10	10	9.1	3.6
		3	10	10	13.0	4.0
20	1	1	10	6	3604.3	766.9
		2	10	8	3604.7	806.5
		3	10	10	3330.5	472.4

Table 3.5 shows some preliminary results of experiments, by explaining the effect of I , E and D , on number of instances optimally solved, average and maximum computation time. The results show that I and E have significant effect on the number of instances solved optimally and computation time, while there is no significant effect of D . As the number of items in an instance increase, the computation time increases, and the number of items optimally solved decreases. Similarly, E has a negative effect on the instances optimally solved and computation time. As the elasticity increases, the number of orders that may depart in each day increases resulting with a number of more possible combinations for each day. In this sense, the computation time increases as expected.

As the third experiment, we tested the effects of number of predefined routes in

the route set on performance of the mathematical models. In this respect, we generated 4 different set of routes, where $|R| = \{5, 10, 20, 50\}$. The best performing mathematical model, MM2, is tested in GAMS 22.8 using Intel Xeon X5482 with 7.4 GHz and 10 GB RAM on 30 relatively easy instances with $I = \{10, 20, 30\}$. As shown on table 3.6, the computation time increases correspondingly with the increase in number of predefined routes. Paired two sample T-test also shows that the increase in computation time is significant. Based upon the fact that several phases, especially local search procedure of VNS loops in route set, we expect the same increase in computation time, if the number of predefined routes increases in VNS algorithm. Therefore, we used the route set with less elements, $|R| = 5$. This is also consistent with the real-life practices, as there are limited number of routes in an annual contract for the sake of simplicity.

Table 3.6: The performance of MM2 under different number of predefined routes in the route set

I	Average CPU			
	$ R = 5$	$ R = 10$	$ R = 20$	$ R = 50$
10	3.0	4.4	7.0	20.0
20	4.4	9.1	10.4	27.5
30	10.1	21.7	26.7	244.6
Average	5.8	11.7	14.7	97.4

3.5.3 Computational Results

In this section, we provide the results of the experiments conducted on MM1, MM2, LB1, LB2 and UB by examining the performances.

3.5.3.1 Experiments on Mathematical Models and Lower Bound Algorithms

We solved MM1, MM2, LB1 and LB2 in GAMS 22.8 using Intel X5482 6.4 GHz with and 10 GB RAM. We set the upper limit for computation time as 3600 sec for mathematical models. We limit each iteration of lower bound algorithms with 120 seconds, and limit overall computation time at 3600 seconds.

Table 3.7 shows the performances of mathematical models, by indicating the number of instances solved optimally, maximum and average CPU times for the instances solved optimally, and the average gaps. MM1 solves 168 instances optimally within 3600 sec, while MM2 solved 169. When we apply paired T-test on the CPU times, MM2 is significantly better than MM1, whereas there is no significant difference between the average gaps. For relatively easy instances, where $E = 0$, there is again no significant difference between MM1 and MM2, while mean CPU time of MM2 is significantly better than MM1 for relatively difficult instances ($E = 1$).

Table 3.7: Performances of Mathematical Models within 3600 sec

I	E	#	MM1			MM2		
			# opt	Gap (%)	CPU	# opt	Gap (%)	CPU
10	0	30	30	-	1.2	30	-	1.1
	1	30	30	-	18.8	30	-	6.2
20	0	30	30	-	3.8	30	-	3.7
	1	30	24	1.2	382.0	24	1.0	334.4
30	0	30	30	-	41.1	30	-	21.5
	1	30	8	2.9	1092.8	7	2.8	1206.9
50	0	30	16	1.6	1183.7	18	1.5	972.8
	1	30	-	7.6	-	-	7.6	-
100	0	30	-	7.9	-	-	8.0	-
	1	30	-	18.8	-	-	16.1	-
Average		300	168	4.0	248.4	169	3.7	217.9

Table 3.8 shows the performances of lower bound algorithms, as well as the comparison of best possible values obtained with mathematical models. When we compare LB1 and LB2 algorithms in terms of CPU time, there is no significant difference between them. In order to indicate the best performing lower bound algorithm, we examine the gap of LB1 and LB2 to the maximum of known best possible values. We apply paired T-test on the gaps to examine solution quality, and mean gap of LB1 ($GAP_{LB1} = 0.02$) is significantly better than those of LB2 ($GAP_{LB2} = 0.04$) for all instances where $E = 0$, such that $P(GAP_{LB1} < GAP_{LB2}) = 0$. On contrary, mean gap of LB2 ($GAP_{LB2} = 0.01$) is significantly better than those of LB1 ($GAP_{LB1} = 0.02$) for instances where $E = 1$, such that $P(GAP_{LB2} < GAP_{LB1}) = 0$. In this sense, LB1 outperforms LB2 in terms of solution quality for relatively easy instances, while LB2 outperforms LB1 for relatively difficult instances. The reason for such a difference in performances parallel to the change in elasticity may be a result of increasing number of vehicles

in a solution, in which there is no elasticity to await orders. When the number of vehicles required to dispatch all orders increase, LB2 algorithm has to add more constraints, thus the solution performance decreases.

When lower bound algorithms are compared to the best possible values found by MM1 or MM2, lower bounds can find the exact best possible value of MMs for 143 instances, and performs better in 35 instances out of 300. Yet, MMs can find better lower bound values in 122 out of 300 instances. Additionally, lower bound algorithms perform approximately 2% worse than the lower bound values obtained by solving mathematical models on the average.

Table 3.8: Performances of Lower Bound Algorithms within 3600 sec

I	E	#	CPU		Gap to Max(BP_{LB1}, BP_{LB2})		Gap to Max(BP_{MM1}, BP_{MM2})	
			LB1	LB2	LB1 (%)	LB2 (%)	LB1 (%)	LB2 (%)
10	0	30	5.2	1.0	0.00	0.00	0.00	0.00
	1	30	2.0	16.0	0.05	0.00	0.05	0.00
20	0	30	45.8	58.3	0.01	0.00	0.01	0.00
	1	30	1929.9	1574.4	3.23	0.65	3.05	0.41
30	0	30	679.6	380.3	0.78	0.54	1.19	0.93
	1	30	3554.0	3424.1	3.03	1.17	4.70	2.77
50	0	30	2816.7	3339.2	0.41	4.77	5.88	10.02
	1	30	3672.5	3663.8	1.50	1.10	1.35	0.89
100	0	30	3687.5	3652.6	0.07	4.31	3.25	7.38
	1	30	3339.3	3726.1	-	0.00	-	1.74
Average		300	1973.3	1983.6	1.01	1.25	2.17	2.41

3.5.3.2 Experiments on VNS

We solved VNS Algorithm in C environment using Intel X5482 6.4 GHz with and 10 GB RAM. In order to identify the maximum outer iteration number, we tested the performances of different levels of outer iteration numbers in a limited number of instances, and select the number with the best performance. We defined stopping conditions; (i) $15 * I$ seconds for maximum CPU time, (ii) 20

for maximum global iteration number, and (iii) $20 * I$ for maximum outer iteration number.

Table 3.9 shows the performances of VNS algorithm and mathematical models. The solution quality of VNS and mathematical models are reported regarding the maximum lower bound achieved by MM1, MM2, LB1 or LB2. The CPU time of mathematical models are reported by indicating the shortest time in 2 ways; (i) if both MM1 and MM2 find the optimal solution, and (ii) the solution time of the best upper bound if there is a gap. In this way, we compare the solution quality and CPU time of VNS by the best performances achieved by both mathematical models. Accordingly, the solution quality of VNS is approximately the same with those of mathematical model on the average. Yet, VNS can find good quality solutions in a very short computation time compared to mathematical models. Although the number of instances that are optimally solved with mathematical models is higher than the ones optimally solved with heuristic algorithm, VNS performs better for the larger problem sizes. For instance, the relative gap of VNS for larger problems with $I = \{50, 100\}$ is 1% better on the average compared to the gap of mathematical models. Additionally, the number of instances, which have a relative gap above 10 % is 16 for VNS, whereas 33 for mathematical models.

Table 3.9: Performances of VNS Algorithm and Mathematical Models

I	E	#	MMs			VNS		
			# opt	Gap (%)	CPU	# opt	Gap (%)	CPU
10	0	30	30	0.00	1.0	30	0.00	5.3
	1	30	30	0.00	6.1	29	0.00	5.0
20	0	30	30	0.00	3.2	29	0.01	16.5
	1	30	27	0.28	748.6	25	0.38	5.5
30	0	30	30	0.00	19.2	29	0.00	31.8
	1	30	9	1.87	2903.8	1	4.00	12.8
50	0	30	19	1.27	1798.3	12	1.37	65.1
	1	30	0	5.79	3622.6	0	6.60	41.6
100	0	30	0	7.59	3629.1	0	8.97	306.3
	1	30	0	13.54	3679.6	0	6.70	299.7
Average		300	175	3.03	1641.2	155	2.80	79.0

3.6 Conclusion and Further Research

In this chapter, we develop a solution approach to SCD-TTFR problem. We define the main problem and its assumptions, propose a mathematical model, which decides on the consolidation of shipments, truck route, intermediate stops and departure day, and examine the computational complexity. We propose three enhancements to the mathematical model; symmetry breaking constraints, defining an upper bound for big M , and direct relationship between order assignment and vehicle departure. Preliminary experiments show that, the mathematical model with all three enhancements outperforms all others, and symmetry breaking constraint is the most striking enhancement. We also define two lower bound algorithms, and an upper bound algorithm. Then, we conduct computational experiments on mathematical models, lower and upper bounds with randomly

generated instances.

The experiments show that, CPU times of MM2 are significantly better than those of MM1 on the average. When we compare LB1 algorithm and LB2 algorithm in terms of CPU time, none of them dominates the other one. However, LB2 is significantly better than LB1 in terms of solution quality for relatively easy instances, while LB2 is better than LB1 for relatively difficult instances. As for the upper bound algorithm, the solution quality of VNS is approximately the same with those of mathematical model on the average. Yet, VNS can find good quality solutions in a very short computation time compared to mathematical models, especially for difficult instances, where $E = 1$.

As an output of this chapter, we introduce a simplified version of SCD-TTFR as case material for undergraduate students including the lecture notes (Tokcaer et al., 2016), and presented the case material as finalist in case competition at INFORMS 2016, Nashville USA.

As for the further research directions, we may improve the quality of lower bound by applying another algorithm. Additionally, the performance of VNS can be enhanced by increasing the iteration limit and including more operations to shaking procedure.

Chapter 4

Shipment Consolidation and Dispatching with Transshipment Terminals and Spot Market Prices

4.1 Introduction

In Chapter 3, we assume that the cost of a vehicle is defined by the annual contracts including a set of predefined routes, which have a set of possible stopping points and associated fixed costs, namely Shipment Consolidation and Dispatching Problem with Transshipment Terminals and Fixed Costs (SCD-TTFR). In this chapter, we will focus on another assumption of real-life practice, in which spot market defines the fixed cost of a rented vehicle. In this case of SCD-TT, the cost of a rented vehicle is again associated with the farthest destination in that

vehicle, yet there are no predefined routes with fixed costs.

In this context, we may follow two solution approaches for the problem with spot market prices, namely SCD-TTSM; (i) to generate new routes and extend the set of routes used in the problem formulation in Chapter 3, (ii) to propose a new problem which has no predefined routes. First assumption requires an algorithm, which generates new routes accordingly with the destinations in an order set, so that the dispatching plans are made with subject to farthest destination in the vehicle. However, as discussed in subsection 3.5.2 of chapter 3, this approach has an outcome of substantial increase in computation time. The second approach entails a new formulation for the problem hence a new solution methodology. As extending the set of fixed routes is not efficient in terms of computation time, we follow the second approach with a new formulation, propose a new mathematical model, and apply a problem specific exact algorithm.

This chapter is organized as follows. We firstly discuss the literature on solution methodology. Then we explain the assumptions of the problem, and define the mathematical model for SCD-TTSM. In section 4.4, we describe the solution methodology with Dantzig-Wolfe decomposition, following column generation and B&P algorithms. Then we tested the performances of both original formulation and Branch-and-Price (B&P) on the randomly generated instances used in Chapter 3. We finally, conclude the chapter by discussing the results and addressing to the future works.

4.2 Literature Review

In this section, we address to the related studies with the SCD-TTSM problem and the similar problems in the literature using the solution methodology. Since SCD-

TTSM problem has a special cost structure, the relevant literature on the problem is rare, in fact, only Koca and Yıldırım (2012) addressed to such a cost structure to the best of our knowledge. Their study examined spare parts distribution system of a major automotive manufacturer in Turkey. In this system, the cost of a given vehicle is defined by the farthest demand point in that vehicle. They modelled the problem as Capacitated Concentrator Location Problem, which is a special case of a network design problem. They proposed a hierarchical approach, which solves the problem in two stages. In the first stage, the demand is aggregated with respect to the capacity requirement of that vehicle. Then in the second stage, the portion of the capacity allocated to the demand is disaggregated with an optimization model. The experiments showed that the hierarchical approach outperformed the direct formulation, especially for the large instances.

As for the solution methodology, column generation is a technique for solving large scale problems with excessive number of decision variables. With a disaggregated formulation of the original problem to a master problem and a sub-problem, an iterative algorithm generates new columns. If column generation provides non-integer solutions, column generation is solved in a branch-and-bound search tree, which is known as B&P algorithm. Following the guiding study of Barnhart et al. (1998), B&P algorithm has been applied on many integer programming problems.

As discussed in Chapter 2, the SCD-TTSM is considered as a transportation planning problem, there is a body of literature, in which column generation and branch-and-price techniques are frequently applied. With respect to the similar problems with SCD-TTSM in the literature, such as SNDP or VRP with time windows, there are many approaches while decomposing the problem, yet mainly two approaches are popular. For the first approach, each column represents a feasible vehicle route, and master problem minimizes the cost of selected routes

(Irnich, 2002; Danna and Pape, 2005; Santos et al., 2013). Second approach assumes that each column represents a feasible working day plan, and master problem minimizes the cost of selected plans.

Considering the first approach, Irnich (2002) applied a branch and price algorithm for a service network design problem of a optimization of the letter mail delivery network, and formulated the master problem as set partitioning model, whereas pricing problem is a shortest path problem, which creates routes subject to capacity constraints and time windows requirements. They also define compatibility of orders, which assigns the orders with at least one common departure time to the same route. In this way, they eliminate the time space.

As regards to the second approach, Azi et al. (2010) introduced a B&P algorithm for VRP with time-windows and multiple-use of vehicles, and reformulate the master problem as a set packing problem, of which every column represents a workday. In their formulation, each pricing problem is again a shortest path problem with resource constraints. They also proposed different branching strategies, such that, branching on customers, vehicles, flow on arcs and two consecutive fractional arcs. They analyzed the performance of the algorithm regarding different characteristics of the problem, and conclude that B&P algorithm can solve the instances upto 25 customers optimally.

Relating to the above described literature, we adopt a similar solution methodology used in the first approach, where each column in the decomposed problem represents a feasible vehicle. Similar to Irnich (2002), we also use the time-windows requirement to define eligibility of orders to be on the same vehicle, and eliminate the time related decisions.

4.3 Problem Formulation

In this section, we firstly define the assumptions of SCD-TTSM, then we explain the formulation of the mathematical model.

4.3.1 Assumptions

The assumptions of SCD-TTSM is particularly based on real-life applications, and similar to the assumptions of SCD-TTFR, yet the route and cost of a vehicle is defined by the farthest destination in that vehicle. With this aspect, the assumptions of the SCD-TTSM are as follows;

- Information on orders, such as dimensions, destination, release date and deadline, are deterministic and initially known.
- The orders can be delivered to their destination either on wheels or by using a transshipment terminal. Each transshipment decision implies a cost proportional to the size of the order.
- The main path of a vehicle is defined regarding the farthest destination in the vehicle, and the fixed cost of the vehicle is associated with that destination.
- All but the farthest destinations in the vehicle are defined as extra stops.
- If there is an extra stop of the vehicle, excessive deviations from the main path is not allowed.
- Fixed costs of the rented vehicles includes a limited number of extra stops, and after that number, each additional stop incurs an extra cost of stopping upto maximum number of stops which cannot be exceeded.

- The number of stops a vehicle can do is limited, hence, the delivery duration is not affected by the number of stops.

Since the orders have time-windows for deliveries (release day and deadline), there is a time dimension in the problem. Hence, transit time to each destination is known and assumed to be constant regardless of other destinations in the vehicle. Within this context, we used H_r^t and G_{ir}^t sets to define time window requirements in Chapter 3. Here, we identify the eligibility of each order pair $k, l \in K$ to be on the same vehicle by defining a parameter, such that;

$$a_{kl} = \begin{cases} 1 & \text{if orders } k \text{ and } l \text{ can be in the same vehicle} \\ 0 & \text{otherwise} \end{cases}$$

To illustrate an example, we assume 3 orders, which have release days as 1, 4 and 6, and deadlines as 10, 12 and 13 respectively. Minimum transit time required to deliver these orders is 6 days. Figure 4.1 illustrates available days for departure for each order with the grey marks. As order (1) and (2) can depart on the same vehicle on day 3, $a_{12} = 1$, whereas order (1) and (3) don't have any days in common for departure, thus $a_{13} = 0$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2														
3														

Figure 4.1: Available days for departure for three orders

With a similar fashion, we define parameter b_{kl} to identify the orders, which cannot be delivered directly with the same vehicle, due to customs restrictions, country

passing problems etc. If the destinations of both orders $k, l \in K$ are possible stopping points on any of the routes, then these two orders may be delivered directly with the same vehicle.

$$b_{kl} = \begin{cases} 1 & \text{if orders } k \text{ and } l \text{ can be delivered directly with the same vehicle} \\ 0 & \text{otherwise} \end{cases}$$

4.3.2 Mathematical Model

With above assumptions, the original formulation (OF) can be formulated as follows;

Indices and Sets

- K Orders, $k \in K$
- I Transshipment Terminals, $i \in I$
- T Vehicles, $t \in T$
- J Destinations, $j \in J$

Parameters

Dimensions of order k ;

- v_k Total volume
- w_k Total weight
- l_k Total length

Vehicle capacities;

- ν Volume
- γ Weight
- δ Length
- p_k Destination of order k , where $p_k \in J$
- d_k Distance to the destination of order k
- des_i Distance to transshipment terminal i

pkm	Per kilometre cost
foc	Fixed operating cost of a vehicle
$cost_{ki}$	Cost of delivering order k from transshipment terminal i
μ	Limit on additional number of stops
ϕ	Number of stops included in the fixed cost, where $1 \leq \phi \leq \mu$
ρ	Additional cost of each stop after ϕ stops
a_{kl}	$\begin{cases} 1 & \text{if orders } k \text{ and } l \text{ can be in the same vehicle} \\ 0 & \text{otherwise} \end{cases}$
b_{kl}	$\begin{cases} 1 & \text{if orders } k \text{ and } l \text{ can be delivered directly with the same vehicle} \\ 0 & \text{otherwise} \end{cases}$

Decision Variables

α_t	Fixed cost of vehicle t
β_t	Number of extra stops for vehicle t
x_k^t	$\begin{cases} 1 & \text{if order } k \text{ is departed in vehicle } t \\ 0 & \text{otherwise} \end{cases}$
y_k^t	$\begin{cases} 1 & \text{if order } k \text{ is delivered directly with vehicle } t \\ 0 & \text{otherwise} \end{cases}$
z_{ki}^t	$\begin{cases} 1 & \text{if order } k \text{ is in vehicle } t \text{ and delivered through transshipment terminal } i \\ 0 & \text{otherwise} \end{cases}$
use_i^t	$\begin{cases} 1 & \text{if vehicle } t \text{ visits transshipment terminal } i \\ 0 & \text{otherwise} \end{cases}$

$$\delta_j^t \quad \begin{cases} 1 & \text{if vehicle } t \text{ visits destination } j \\ 0 & \text{otherwise} \end{cases}$$

Using above sets, parameters and decision variables, the mathematical model is formulated as follows;

$$OF : \text{ Minimize } \sum_{t \in T} \left(\alpha_t + \rho \beta_t + \sum_{i \in I} \sum_{k \in K} cost_{ki} z_{ki}^t \right) \quad (4.1)$$

$$\text{Subject to; } \alpha_t \geq (d_k.pkm + foc)y_k^t \quad \forall k, t \quad (4.2)$$

$$\alpha_t \geq (des_i.pkm + foc)use_i^t \quad \forall i, t \quad (4.3)$$

$$use_i^t \geq z_{ki}^t \quad \forall k, i, t \quad (4.4)$$

$$\delta_{p_k}^t \geq y_k^t \quad \forall k, t \quad (4.5)$$

$$\beta_t \geq \sum_{k \in K} \delta_{p_k}^t + \sum_{i \in I} use_i^t - \phi \quad \forall t \quad (4.6)$$

$$\beta_t \leq \mu \quad \forall t \quad (4.7)$$

$$x_k^t = y_k^t + \sum_{i \in I} z_{ki}^t \quad \forall k, t \quad (4.8)$$

$$\sum_{t \in T} x_k^t = 1 \quad \forall k \quad (4.9)$$

$$x_k^t + x_l^t \leq 1 \quad \forall t, (k, l) \mid a_{kl} = 0 \quad (4.10)$$

$$x_k^t + x_l^t \leq 1 \quad \forall t, (k, l) \mid b_{kl} = 0 \quad (4.11)$$

$$\sum_{k \in K} v_k x_k^t \leq \nu \quad \forall t \quad (4.12)$$

$$\sum_{k \in K} w_k x_k^t \leq \gamma \quad \forall t \quad (4.13)$$

$$\sum_{k \in K} l_k x_k^t \leq \delta \quad \forall t \quad (4.14)$$

$$\alpha_t \geq \alpha_{t-1} \quad \forall t \mid t \geq 2 \quad (4.15)$$

$$x_k^t, y_k^t, z_{ki}^t, use_i^t, \delta_j^t \in \{0, 1\} \quad \forall k, i, t, j \quad (4.16)$$

$$\alpha_t \geq 0 \quad \forall t \quad (4.17)$$

$$\beta_t \in \{0, 1, 2, \dots\} \quad \forall t \quad (4.18)$$

Objective function (4.1) minimizes the total cost including the fixed cost, the cost of extra stops and transshipment costs. Constraint sets (4.2) and (4.3) identify the fixed cost of vehicle regarding the farthest distance in the vehicle. Constraint set (4.4) defines if an order is assigned to a vehicle if it is delivered to its final destination directly or by using a transshipment terminal. Constraint set (4.5) assigns the destination of an order in a vehicle, if the order is delivered directly to its destination with that vehicle. Constraint sets (4.6) and (4.7) identify and limit the number of extra stops. Constraint set (4.8) ensures that the order is assigned to a vehicle if it is delivered directly or by using a transshipment terminal. Constraint set (4.9) ensures that each order is served. Constraint sets (4.10) defines the eligibility of orders to be in the same vehicle regarding the release days and deadlines. Constraint set (4.11) identifies if the orders can be delivered directly with the same vehicle. Constraint sets (4.12), (4.13) and (4.14) satisfy the capacity constraints. Constraint set (4.15) ensure the use of vehicles in sequence. Constraint sets (4.16), (4.17) and (4.18) are the integrality constraints.

4.4 Solution Methodology

Since the structure of aforementioned OF is suitable for a disaggregated formulation, we applied Dantzig–Wolfe decomposition, which is known to be successful in solving large scale problems (Desrosiers and Lübbecke, 2005). With Dantzig–Wolfe decomposition, the OF is reformulated as master problem, which has a large number of variables, while the sub-problem, so called pricing problem, is defined by a subset of the constraints of the original formulation. The master problem having excessive number of variables, thus columns, is difficult to manage (Vanderbeck, 2000). Therefore, a restricted master problem, which has a subset of columns of master problem is solved. Concurrently, sub-problem, so called pricing problem generates further columns by using the dual prices of restricted master problem. If pricing problem generates a new column with a negative reduced cost (for a minimization problem), the column is added to restricted master problem as a new entering column. In this context, master problem and pricing problem are solved iteratively until non-negative reduced cost is identified. Once positive reduced cost returns, the iteration stops as the incumbent solution is optimal for the restricted master problem. However, this optimal solution for restricted master problem does not guarantee the optimality of original formulation. Eventually, if this solution at the end of column generation does not satisfy integrality conditions, it is not optimal to the original problem. In this case, column generation is solved in a branch-and-bound search tree after branching, which is so called branch-and-price algorithm (Barnhart et al., 1998; Desrosiers and Lübbecke, 2005; Lübbecke and Desrosiers, 2005).

In this section, we firstly reformulate OF by applying Dantzig-Wolfe decomposition, then we explain the column-generation algorithm and branch-and-price scheme.

4.4.1 Dantzig-Wolfe Decomposition

Assuming that the route and the cost of a vehicle can be calculated if the orders assigned to that vehicle are known, we decompose the OF by using the Constraint set (4.9), and define the Master Problem as a set covering problem. In this set covering problem, each column represents a feasible vehicle, which has a group of orders in it in a way that capacity constraints and time-windows requirements are satisfied. Then R represents the set of feasible vehicles, and c_r defines the cost of vehicle $r \in R$. For a feasible vehicle, we also define ω_{kt} , the value of which is 1, if order k is in vehicle r and 0 otherwise. Accordingly, the decision variable here will be;

$$x_r = \begin{cases} 1 & \text{if vehicle } r \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Then, we may formulate the master problem (MP) as;

$$MP : \text{ Minimize } \sum_{r \in R} c_r x_r \quad (4.19)$$

$$\text{Subject to; } \sum_{r \in R} \omega_{kr} x_r \geq 1 \quad \forall k \in K \quad (4.20)$$

$$x_r \in \{0, 1\} \quad \forall r \in R \quad (4.21)$$

Objective function (4.19) minimizes the total cost of selected vehicles. Constraint set (4.20) assigns each order to a vehicle, and Constraint set (4.21) assures the integrality of decision variable x_r .

Above MP retains Constraint set (4.9) of original formulation explicitly, whereas remaining constraints are used while generating feasible routes. By solving the linear programming (LP) relaxation of the MP in a column generation procedure, we obtain λ_k variables, which are the negative dual prices of constraint (4.20) for

each order $k \in K$. We can generate further feasible columns by solving the pricing problem (PP), which is a knapsack problem with side constraints and multiple dimensions;

$$PP : \text{ Minimize } \alpha + \rho\beta + \sum_{i \in I} \sum_{k \in K} cost_{ki} z_{ki} - \sum_{k \in K} \lambda_k x_k \quad (4.22)$$

$$\text{ Subject to } \alpha \geq (d_k.pkm + foc)y_k \quad \forall k \quad (4.23)$$

$$\alpha \geq (des_i.pkm + foc)use_i \quad \forall i \quad (4.24)$$

$$use_i \geq z_{ki} \quad \forall k, i \quad (4.25)$$

$$\delta_{pk} \geq y_k \quad \forall k \quad (4.26)$$

$$\beta \geq \sum_{k \in K} \delta_{pk} + \sum_{i \in I} use_i - \phi \quad (4.27)$$

$$\beta \leq \mu \quad (4.28)$$

$$x_k = y_k + \sum_{i \in I} z_{ki} \quad \forall k \quad (4.29)$$

$$x_k + x_l \leq 1 \quad \forall (k, l) | a_{kl} = 0 \quad (4.30)$$

$$x_k + x_l \leq 1 \quad \forall (k, l) | b_{kl} = 0 \quad (4.31)$$

$$\sum_{k \in K} v_k x_k \leq \nu \quad (4.32)$$

$$\sum_{k \in K} w_k x_k \leq \gamma \quad (4.33)$$

$$\sum_{k \in K} l_k x_k \leq \delta \quad (4.34)$$

$$x_k, y_k, z_{ki}, use_i, \delta_j \in \{0, 1\} \quad \forall k, i, j \quad (4.35)$$

$$\alpha \geq 0 \quad (4.36)$$

$$\beta \in \{0, 1, 2, \dots\} \quad (4.37)$$

4.4.2 Column Generation

The column generation (CG) procedure starts with a feasible set of columns $R' \subset R$, which is restricted master problem (RMP). As for the initialization of CG procedure, we define the initial column set R' by assigning each order to a single vehicle having the cost associated with the destination of that order. The linear relaxation of RMP, so called LRMP, gives us the negative dual prices $\lambda_k, \forall k \in K$. LRMP and PP is solved iteratively, and at each iteration, PP finds a new column with the least reduced cost by using the dual prices from the LRMP. LRMP is solved repeatedly until no negative reduced cost is identified, thus no more columns are added. When CG stops at the root node, if the solution obtained is integer, then it is optimal to OF. Otherwise, CG is embedded in a branch-and-bound algorithm, so called branch-and-price (B&P), with the set of columns we generated initially and during column generation algorithm. Before initializing the B&P algorithm, we solve the RMP as a MIP at root-node, and get an integer solution with the columns generated at the root-node. We consider it as an acceleration strategy if the integrality conditions cannot be satisfied at the root-node.

4.4.3 Branch-and-Price Algorithm

In the branch-and-bound algorithm, we firstly define variables to branch. We select a pair of orders, (k, l) , which are on the same column, and the sum of x_r values for that column in the visited node is closest to the target pair value τ , such that;

$$pair_{kl} = \left| \tau - \sum_{\substack{r \text{ in } R; \\ \omega_{kr}=1, \omega_{lr}=1}} x_r \right| \quad \forall k, l$$

Consequently, we select the (k, l) pair with minimum $pair_{k,l}$ value as branching variables, and create two child nodes. At the right child node, we assume that (k, l) order pair should be in the same vehicle, whereas (k, l) order pair should not be in the same vehicle at the left child node. Without having a substantial change in RMP, we only delete the respective columns, which has the (k, l) order pair in different vehicles at the right child node. Similarly, we delete the columns having the (k, l) order pair in the same vehicles at the left child node. As for the PP, we added a relevant constraint with respect to the visited child node, such that; we update PP by adding constraint (4.38), which restrains (k, l) order pair to be on different vehicles at the right child;

$$x_k - x_l = 0 \tag{4.38}$$

If PP is solved at the left child, then we add constraint (4.39) preventing (k, l) order pair to be in the same vehicle;

$$x_k + x_l \leq 1 \tag{4.39}$$

In the use of aforementioned branching scheme, there are three aspects that affects the number of visited nodes; (i) search strategy, (ii) τ value and (iii) node selection. At this point, there are two search strategies to be followed; depth-first and breadth-first. Besides obtaining feasible solutions in shorter computation time (Fayed and Atiya, 2013), depth-first search strategy allows for easy column management in RMP and constraint handling in PP. Depth-first strategy first moves downward the search tree by definition, and moves upwards if the node is fathomed. Thus, we only need to add the deleted columns to RMP, and delete the constraints from PP with respect to branching rule, while leaving a node upwards the search tree. Once the node is left, the respective information on the left node

is no longer necessary. On contrary, breadth-first strategy requires a particular memory structure for column management and constraint handling. Additionally, the process of deleting / adding columns and constraints is expected to be repeated several times during the search. With all these aspects, we explore the search tree by using depth-first strategy.

Once we identify the search strategy, we consider τ value in tandem with node selection. There are two alternatives for node selection, such that we may first visit right child, then left child node, or vice versa. In this context, we tested the algorithm on small instances, and examined the number of visited nodes with different τ values on both alternatives. We conclude that the algorithm visited less nodes and generated less columns with first right-then left node visiting alternative and $\tau = 0.7$. Considering that greater values of τ is an indication of (k, l) order pair's likelihood to be on the same vehicle, first right-then left strategy is to the purpose by imposing (k, l) order pair to be on the same vehicle.

In B&P algorithm (Algorithm 4.1), CG procedure is executed as described in subsection 4.4.2 at each node of search tree, such that RMP is initialized with the columns used in the last visited node. If the optimal solution of the LPMP at a non-root node is not integer, and the optimal objective value of the LPMP is less than the best known upper bound, then we branch on another selected (k, l) order pair. As for the branching strategy, first right child node is visited, and RMP and PP is updated accordingly. If the current node is fathomed, the last generated left child node will be visited. BP algorithm stops when all nodes are visited and stopping condition for time limit is satisfied.

Algorithm 4.1 Branch-and-Price

Initialization: Initialize RMP_n with set of initial columns. Let n define the number of nodes and UB denote the best known upper bound at the root-node.

Main Step: Until all nodes are visited ($n = 0$), or time limit is exceeded, repeat the following steps;

Step 1: *Column Generation Procedure*

(1a) Initialize RMP_n with set of columns as for node n

(1b) Solve LRMP and transfer dual prices λ_k^n of node n to PP_n .

(1c) Solve PP_n to optimality, and get reduced costs, namely ϕ_n . If $\phi_n < 0$, generate a new entering column to RMP_n and go to **Step 1**, else proceed to **Step (1d)**.

(1d) If the optimal solution of the LPMP is non-integer, or the integer optimal objective value of the LPMP is less than the best known upper bound UB^* , go to **Step (2a)**. Else go to **Step (2b)**.

Step 2: *Branching*

(2a) Update $pair_{kl}$, branch on selected (k, l) order pair, update RMP_n and PP_n , such that $n \leftarrow n + 2$ at right child node and $n \leftarrow n + 1$ at left child node. Visit the right child node and go to **Main Step**.

(2b) Fathom current node, set $n \leftarrow n - 1$, and update UB^* if the optimal solution of LRMP is greater than best known UB^* . If there are any unexplored left child nodes ($n > 0$), visit the last created left child and go to **Main Step**.

4.5 Computational Experiments

In the SCD-TTSM, we assume that the routes are no longer fixed, and the distances from origin to each destination is known. Thus, we adopt a distance based costing structure, where the cost of used route is defined regarding the farthest destination in the vehicle. In this sense, there are 4 type of costs in the cost function; per kilometre cost multiplied by the farthest distance, extra stopping costs, fixed cost of operating a vehicle (e.g. Ro-Ro expanses, driver fees etc.) and transshipment costs. In order to obtain similar route costs with those of Chapter 3, we randomly generate different routes, and calculate the fixed costs of those routes accordingly with the cost structure in Chapter 3. Then, we examine the effect of extra stops and farthest distance on the fixed cost by applying regression analysis, and obtain the cost parameters, namely per kilometre cost, extra stopping cost and fixed cost of operating a vehicle, with $R^2 = 0.9$. Transshipment assumptions are the same, thus there is no change in transshipment costs.

We code both OF and B&P on CPLEX Optimization Studio and conduct the computational experiments on the randomly generated instances used in Chapter 3 by using Core i7 7500U with 2.7 GHz and 8GB RAM. We also set the time limit as 3600 seconds as for the stopping condition.

As a preliminary experiment, we solve the RMP as mixed integer programming problem (MIP) at different phases of B&P algorithm as an acceleration strategy. When the column generation procedure stops at root node, we solve RMP as MIP with the columns generated until root node. Then, we test both algorithms, B&P with and without MIP, on 120 instances with different number of orders ($I = \{20, 30\}$) so as to compare the computation times and the solution quality. Table 4.1 shows the performances of the algorithms in terms of gap at root node, CPU in seconds, number of generated columns and number of visited nodes. Both relative gap and CPU time of B&P with MIP is significantly less than those of B&P without MIP ($P=0.03$). Moreover, B&P with MIP generates less columns than the algorithm without MIP ($P=0.03$), yet the number of visited nodes is not significantly different ($P=0.17$).

Table 4.1: Reported Performances of Preliminary Experiment

I	B	Gap at RN (%)		CPU		# of Col		Nodes Visited	
		without MIP	with MIP	without MIP	with MIP	without MIP	with MIP	without MIP	with MIP
20	0	11.9	0.2	1.6	2.0	42.1	42.1	8.3	8.2
	1	11.8	0.5	10.6	11.2	144.0	137.3	123.6	110.8
30	0	15.8	0.5	22.0	20.6	79.3	79.0	90.9	84.7
	1	28.5	1.9	1008.7	958.0	889.1	860.0	6898.8	6740.1
Average		17.0	0.8	260.7	247.9	288.6	279.6	1780.4	1735.9

We also examine the convergence curves of an instance for both algorithms; B&P without MIP (Figure 4.2) and B&P with MIP (Figure 4.3). The convergence curves for both algorithms are similar, and both algorithms can reach near optimal solutions in a short computation time. However, B&P with MIP reaches to 3.71

% of gap at 70th iteration, whereas B&P without MIP provides 11 % of gap at the same iteration, and reaches to 3.71 % at 3471th iteration. Moreover, B&P without MIP continues upto 5868 iterations, which is quite higher than B&P with MIP.

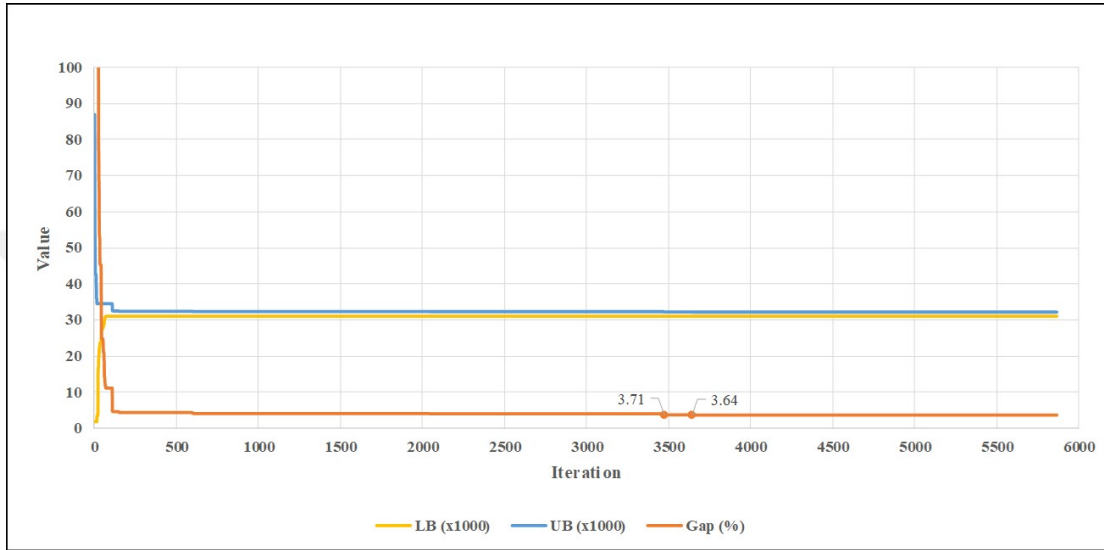


Figure 4.2: Convergence Curves of B&P Algorithm without MIP

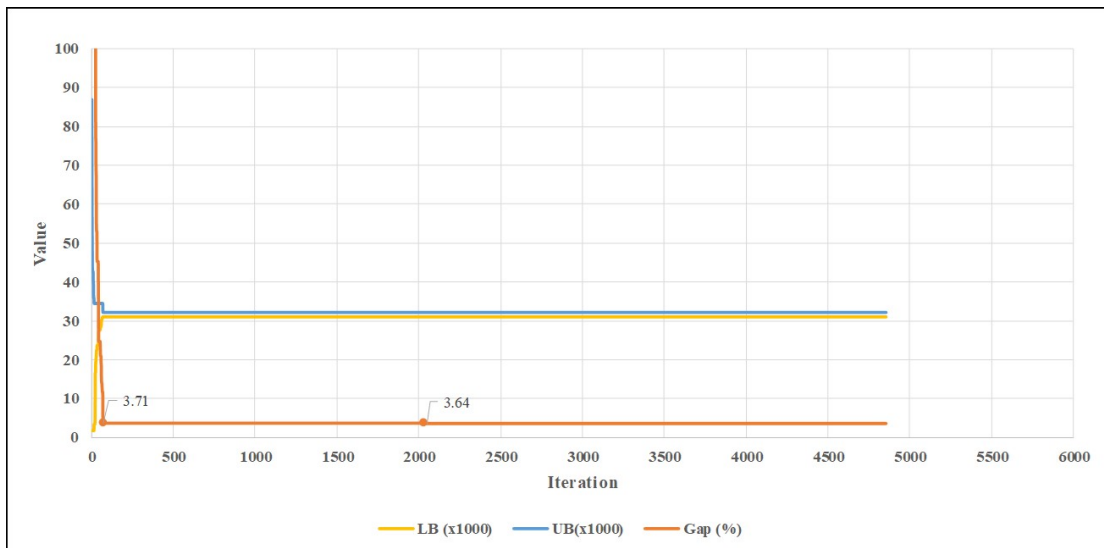


Figure 4.3: Convergence Curves of B&P Algorithm with MIP

In this respect, we assume that solving B&P with MIP enhances the overall performance of the algorithm by decreasing the gap and computation time. However, the number of columns, thus problem size of RMP substantially increases for large scale problems, and consequently, the computation time required to solve RMP as MIP increases. Hence, we solve RMP as MIP only at two phases; (i) at the end of root node because the algorithm generates less columns after root node, and (ii) when B&P algorithm stops as the gap may decrease for the problems that stops due to time limit. We additionally set a time limit as of 180 seconds, when solving RMP as MIP, because that the problem size, thus the computation time, substantially increases for the larger sizes of the order set, especially where $I = \{50, 100\}$.

Hereinafter, we firstly compare the solution quality and computation performance of B&P with those of OF, then examine the results of experiments on B&P with MIP.

4.5.1 Performances of Original Formulation and Branch-and-Price

In this section we compare the performance of original formulation with branch-and-price. We foremost measure the quality of lower bounds of original formulation (OF) and branch-and-price by measuring the mean relative deviation (in %) between the known highest lower bound (Figure 4.4). As for the lower bound of OF, we report the best known bound, whereas lower bound of B&P is linear programming relaxation of restricted master problem (RMP). Our computational experiments suggest that LP relaxation of RMP gives us tighter lower bounds.

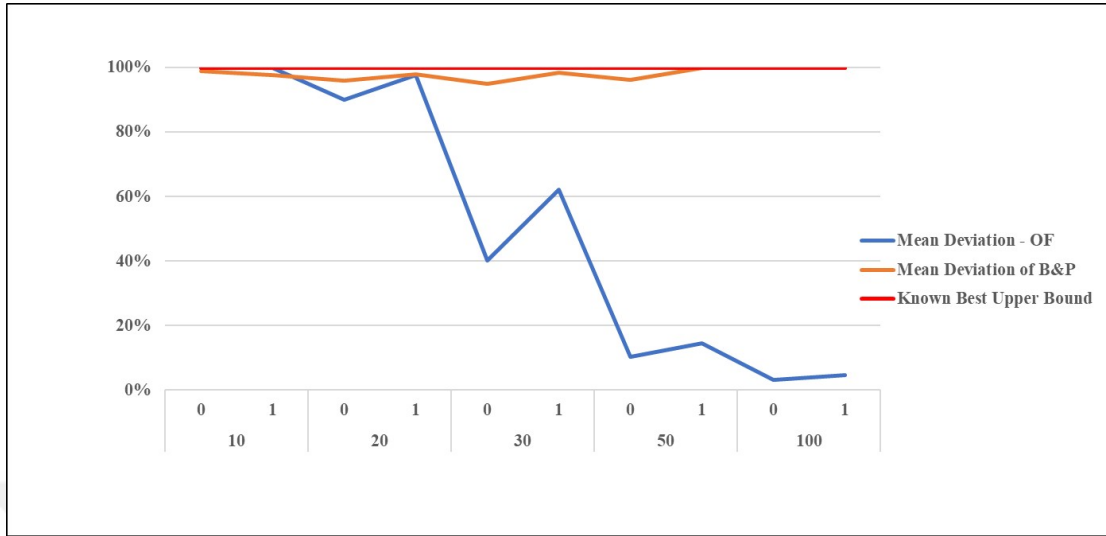


Figure 4.4: Mean Deviation of Lower Bounds

As shown on Figure 4.4, original formulation cannot provide good lower bounds especially for the instances that could not be solved to proven optimality within the time limit; thus, the reported gap of original formulation is considerably high and insignificant. In this sense, we only compare the solution quality and CPU for instances with $I = \{10, 20\}$ in Table 4.2. Out of examined 120 instances, OF can find optimal solutions only for 88 instances, while B&P can solve all to optimality. Additionally, the computation time performance of B&P is expectedly better than OF, in other means, B&P can find optimal solutions in 22 seconds on the average, whereas average CPU time for the instances that OF can solve to optimality is 248 seconds. Note that, the average CPU is 1142 seconds on the average for all examined instances, including the ones that terminate due to time limit.

Table 4.2: Performances of Original Formulation and Branch-and-Price

I	B	#	OF				BP			
			# opt	Gap (%)	Ave CPU	Max CPU	# opt	Gap (%)	Ave CPU	Max CPU
10	0	30	30	0.00	2.18	5.43	30	0.00	0.13	0.26
	1	30	30	0.00	2.36	4.79	30	0.00	0.46	3.13
20	0	30	8	12.16	2810.37	3604.68	30	0.00	2.02	11.40
	1	30	20	2.45	1751.94	3604.69	30	0.00	11.19	72.66
Average		120	88	3.65	1141.71	3604.69	120	0.00	3.45	21.86

4.5.2 Computational Performance of B&P

In this section, we examine the results of experiments on B&P in four aspects; solution quality, computation time, number of columns added and nodes visited.

- *Solution Quality*

Table 4.3 shows the solution quality of B&P algorithm by indicating the number of instances solved optimally, and the relative gap at the root node and the end of the algorithm including MIP results. Key findings regarding the solution quality are as follows;

Table 4.3: Solution Quality of Branch-and-Price

I	E	Root Node		MIP after Root Node		Branch-and-Price		MIP after Branch-and-Price	
		# opt	Gap (%)	# opt	Gap (%)	# opt	Gap (%)	# opt	Gap (%)
10	0	23	1.5	29	0.0	30	0.0	30	0.0
	1	18	7.1	24	0.4	30	0.0	30	0.0
20	0	8	11.9	18	0.2	30	0.0	30	0.0
	1	9	11.8	13	0.5	30	0.0	30	0.0
30	0	2	15.8	5	0.5	30	0.0	30	0.0
	1	3	28.5	3	1.9	24	1.0	24	1.0
50	0	0	28.8	0	2.7	22	2.3	22	2.2
	1	1	48.4	1	2.4	5	2.0	5	1.8
100	0	0	51.5	0	7.4	0	7.0	0	6.7
	1	0	75.8	0	1.9	0	1.7	0	1.2
Average		64	28.1	93	1.8	201	1.4	201	1.3

- B&P algorithm obtains the optimal solution for 201 (67%) instances out of 300, while 64 (21%) of them are solved to optimality at root node.
- The average gap changes under different problem characteristics. For the instances with less orders ($I = \{10, 20\}$), the average gap is 8%, while it is 41.5% for the larger instances (with $I = \{30, 50, 100\}$). For all instances, the gap at root node is 28.1% on the average. If we solve RMP as mixed integer programming problem with the columns generated until the root node, the average gap decreases to 1.8%, and the number of instances solved optimally at the root node increases to 93 (31%).
- If the optimal solution cannot be found at the root node, the average gap decreases to 1.4% when B&P algorithm terminates. In this sense, the average improvement after root node is 0.4%, which is relatively low than expected.
- As the number of orders in an instance (I) increase, the number of instances solved to optimality decreases. The average gap increases for the larger number of orders.
- The elasticity of orders, E , doesn't have a significant effect on the solution quality. However, for the instances with 50 and 100 orders ($I = 50$ and $I = 100$), the average gap of instances with $E = 0$ is significantly different and higher than the gap of instances with $E = 1$.

- *Computation Time*

Table 4.4 shows the computation time performance of B&P algorithm, and the key findings on computation time performance are as follows;

- Average CPU time for all instances is 1261 seconds. As for the instances

solved optimally, the average CPU is 109 seconds, while maximum CPU is 3440 seconds.

- For 157 instances out of 300 (52%), optimal solution is found within 60 seconds.
- CPU time increases as the number of orders increase, such that average CPU is 0.3 seconds for instances with 10 orders, while it is 6.6 seconds for instances with 20 orders.
- CPU time increases, if the orders have an elasticity of waiting time for departure. The average CPU time is 998 seconds for the instances which don't have elasticity for departure ($E = 0$). For the instances with $E = 1$, average CPU is 1524 seconds.

Table 4.4: Computation Time Performance of Branch-and-Price

I	E	Root Node		B&P	
		Ave CPU	Max CPU	Ave CPU	Max CPU
10	0	0.1	0.2	0.1	0.3
	1	0.2	0.4	0.5	3.1
20	0	1.0	1.9	2.0	11.4
	1	1.7	3.1	11.2	72.7
30	0	4.3	6.8	20.6	87.2
	1	7.5	12.3	958.0	3600.2
50	0	26.1	42.1	1370.5	3600.4
	1	57.1	87.9	3049.7	3600.9
100	0	397.9	617.1	3600.8	3602.0
	1	959.4	1377.9	3602.9	3617.1
Average		145.5	1377.9	1261.6	3617.1

- *Number of Columns Added*

Table 4.5 shows the number of columns added during B&P algorithm including the computation time performances as well. In the table, first section shows the results at the root node. Second section shows the number of columns added after root node until the algorithm terminates. The third section represents the total number of columns added when the algorithm stops. The key findings on the number of added columns are summarized as follows;

- On the average, the number of added columns is 489 and the maximum number of added columns is 5513.
- The number of columns added changes under different problem features, such that the number of columns added increases parallel to the number of orders in an instance. For the instances with $I = 10$, the average number of added columns per order is 2, while on the average 4 columns added per order for the instances with $I = 20$, and 16 for the instances with $I = 30$.
- The number of columns added increases, if the orders have an elasticity of waiting time for departure. The average number of added columns per order is 16 for the instances which have elasticity for departure ($E = 1$). For the instances with $E = 0$, average number of columns per order is 4.

Table 4.5: Number of Columns Added

I	E	Root Node		After Root Node		Total	
		CPU	# ColAdd	CPU	# ColAdd	CPU	# ColAdd
10	0	0.11	13.8	0.02	0.1	0.13	13.9
	1	0.21	19.6	0.25	5.3	0.46	24.9
20	0	0.96	35.8	1.06	6.3	2.02	42.1
	1	1.74	54.7	9.45	82.7	11.19	137.3
30	0	4.29	59.6	16.29	19.4	20.58	79.0
	1	7.46	93.7	950.54	766.3	957.99	860.0
50	0	26.13	122.2	1668.32	220.5	1694.45	342.7
	1	57.14	192.4	2992.56	1406.8	3049.70	1599.2
100	0	397.88	302.5	3202.92	257.5	3600.80	560.0
	1	959.43	452.4	2643.42	781.0	3602.85	1233.4
Average		145.53	134.7	1148.48	354.6	1294.02	489.3

- Number of Nodes Visited

Table 4.6 displays the number of visited nodes till the algorithm stops including the number of instances solved to optimality. Considering that 93 instances are solved optimally at root node, for remaining 207 out of 300 instances, the number of visited nodes is 2571 on the average, and for these instances, the maximum number of visited nodes is 34810. The number of visited nodes after root node substantially increases as the the number of instances in an order increases.

Table 4.6: Number of Nodes Visited

I	B	# of opt	# of Nodes	
			Ave	Max
10	0	23	2.3	4
	1	18	18.2	84
20	0	8	11.2	62
	1	9	158.3	1118
30	0	2	90.6	422
	1	3	7489.0	34810
50	0	0	3617.7	11360
	1	1	7108.3	11246
100	0	0	2113.3	3310
	1	0	676.3	1952
Average		64	2571.6	34810

In addition to the aforementioned key findings, we examine the number of columns generated on the convergence curve of an instance (Figure 4.5), and the algorithm can find the optimal solution at a very early stage of iterations, yet it continues to add columns, which are redundant.

Figure 4.6 shows the computational performance of the algorithm including the number of visited nodes and generated columns, under different problem specifications. As shown on the figure, time required for root node increases parallel to the increase in number of orders in an instance, thus, time left for branching is less for the instances, which cannot be solved to proven optimality within the time limit. In this case, the CPU left after root node has a negative effect on the nodes visited and the columns generated, which in return affects the solution quality, especially for large instances. Likewise, for the instances with $I = 100$, the algorithm generates less columns by visiting more nodes where $E = 0$. In

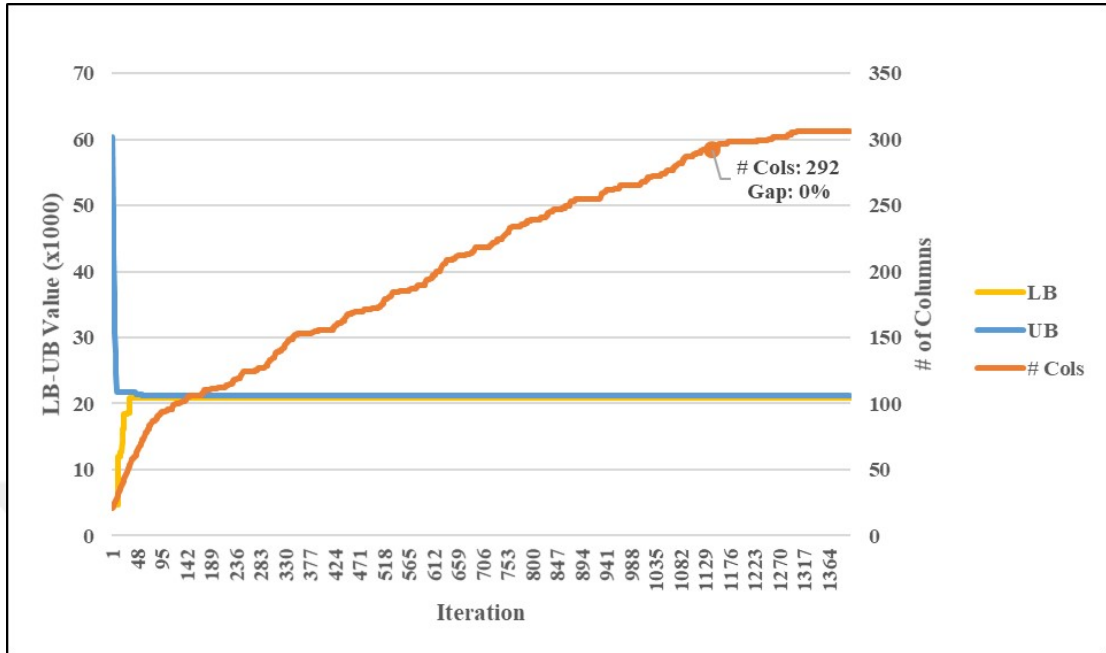


Figure 4.5: Convergence Curves of B&P Algorithm for an instance

contrast, the algorithm generated more columns at each node for instances with $E = 1$. Therefore, the solution quality is better for $E = 1$ than $E = 0$, although CPU time after root node is shorter for $E = 1$ than $E = 0$.

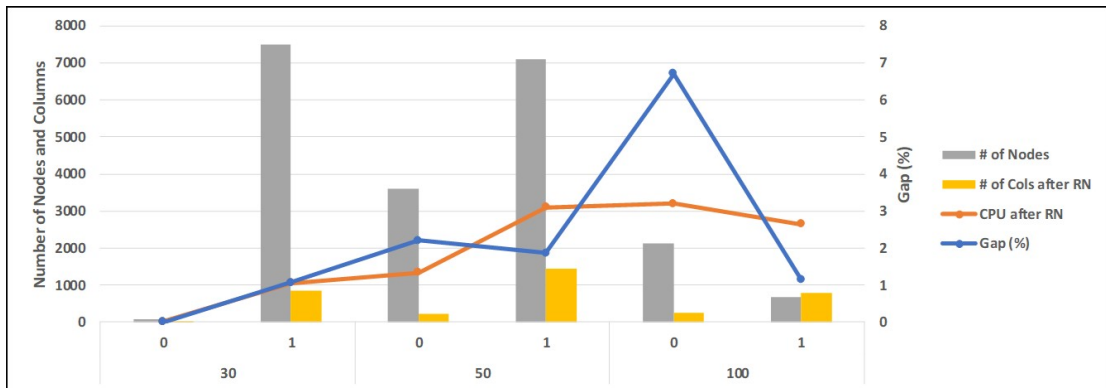


Figure 4.6: Number of Nodes and Columns Compared to Computational Performances

4.6 Conclusion and Future Works

In this chapter, we examine the SCD-TT with spot market prices and propose a mathematical model and an efficient exact algorithm. We decompose the problem, and reformulate the mathematical model in master and pricing problem. Then we propose a branch-and-price algorithm, in which an iterative column generation algorithm generates new columns by solving pricing problem at each iteration.

Our algorithm is able to solve randomly generated instances of previous chapter in reasonable computation times, and provides upper bounds with good quality. Compared to the performance of original formulation (OF), B&P outperforms OF in terms of both computation time and solution quality. Furthermore, B&P provides better lower bounds than the OF. Yet, the lower bounds of B&P, especially for the large scale instances, can be improved by changing search strategy. In the proposed algorithm, we adopt a depth first strategy, which can be replaced by breadth-first strategy. The former search strategy reaches distant child nodes easily, and may provide good quality upper bounds in shorter computation time, while the latter one may provide better lower bounds by exploring some child nodes early. However, depth first may be computationally expensive as it visits many sub-branches before obtaining the optimal solution. In this context, Fayed and Atiya (2013) proposed a mixed breadth-depth traversing algorithm, which substantially decreases the computation time to reach the optimal solution. Therefore, we may follow a similar approach as for the search strategy.

In addition to the improvement in quality of lower bounds, number of visited nodes needs further attention. As the number of orders in an instance increases, the number of visited nodes increases considerably. In this respect, changing the a mixed search strategy may result in an improvement on the number of

visited nodes. We also plan to test different branching strategies, and compare the performances.

As our computational experiment shows, the pricing problem can be solved in a very short time; yet, CPU required for pricing problem may substantially increase for the instances including more than 100 orders. Therefore, we may apply fast and greedy solution approaches to solve pricing problem as another extension.



Chapter 5

Integrated Three Dimensional Bin Packing Problem and SCD-TT

5.1 Introduction

In chapter 3 and 4, the orders are placed in the vehicle regarding the total volume, weight and loading meter of the orders in a vehicle. The only constraint is that the total amount of orders should be less than or equal to the vehicle capacities. However, the solution achieved with this assumption may be infeasible, as the items in a vehicle may not fit in, even if total volume is less than the capacity. We may mitigate the risk of an infeasible solution by reducing the allowed capacity, yet, such an assumption may result in inefficient use of vehicle capacities. Both infeasible solutions and inefficient use of vehicles are undesirable outcomes. Thus, in this part, we propose a feasibility check mechanism, which considers bin packing

assumptions and positioning of each item in a vehicle.

This chapter is organised as follows. In the second section, we will give a brief literature review on the problem assumptions and solution methodology. Then, we will propose a mathematical model in the third section. We will explain the heuristic algorithm in the fourth section, and test the performance of heuristic algorithm on specified two problems, SCD-TTFR and SCD-TTSM, in the fifth section. We will finally conclude the findings and discuss the future works.

5.2 Literature Review

The problem referred in this chapter is a 3D-BPP, which has many variants in the literature, considering different real life assumptions. The role of 3D-BPP in this chapter is to check feasibility of the vehicles obtained in chapter 3 and 4. Therefore, the 3D-BPP will solve only one vehicle at a time and check if the items in the vehicle can fit regarding the capacities. Such a variant of 3D-BPP is referred as Container Loading Problem CLP. We will also include 5 real life assumptions; which are stability constraints, orientation, weight distribution, loading sequence and stacking constraints.

- **Stability Constraints** Stability or vertical load balance constraints ensure the safety of loaded items, and restrain items to fall down or tilt over each other. Therefore, the items should be balanced from top or bottom surface (Bischoff and Ratcliff, 1995b) in a way that the center of gravity of each item is on top of another item or vehicle floor (Lin et al., 2006). Such a constraint also increases the efficient use of available space in a container (Bischoff, 1991). The most common way to satisfy the constraint is to limit

the unsupported bottom surface of an item on vehicle floor or top of another item. While, this limit may be a fraction of the item size (Junqueira et al., 2012), it may also be fully restricted so that all items' bottom surfaces are 100% on top of another item (Bortfeldt and Wäscher, 2013). In the mathematical model we will use the latter assumption; yet, in the heuristic algorithm we do not allow the items' bottom surface to be free of the ground, not even partially.

- **Orientation Constraints** An item may be orientated in a container in at most 6 different ways. Recent review of Bortfeldt and Wäscher (2013) listed the studies allowing different type of orientations; such as allowing only one orientation of vertical or horizontal (Martello et al., 2000), allowing only vertical and restricting horizontal orientation (Hemminki et al., 1998) and vice versa (Bischoff and Ratcliff, 1995b), and allowing both orientations (Wang et al., 2008). In our setting we will assume that each axis of each box may stand vertically or not, resulting with 2, 4 or 6 possible different orientations. This is the approach used in the data set which is mostly used in the bin packing literature.
- **Weight Distribution Constraints** For a stable and safe loading, the weight distribution in a vehicle should be balanced. Therefore, center of gravity of the total weight should be close to the midpoint of the vehicle (Davies and Bischoff, 1999). We will consider a similar assumption for the problem with respect to the real life practice, in which the weight distribution is allocated regarding the load on axles. Respectively, we assume that the vehicle has three compartments, of which the second one is the biggest one, and has the largest weight capacity. Thus, the first and the third ones cannot accommodate extremely heavy goods.

- **Loading Sequence Constraints** In the first part of the thesis, the items of each order are grouped and planned to be loaded on the vehicle together. There are several studies in the literature discussing the same assumption (Bischoff and Ratcliff, 1995b), (Junqueira et al., 2011), (Ceschia and Schaerf, 2013), and they all address to the problem in a similar way. There are 2 real life considerations in this manner; firstly, the orders delivered through a transshipment terminal may be mixed and grouped together to benefit from use of space as those items will be sorted in the cross-dock before delivery to the final destination. Secondly, the orders should be loaded on the vehicle regarding the unloading sequence, therefore last order loaded on the vehicle should be the first one to be offloaded. This last-in-first-out (LIFO) constraint ensures that the order to be unloaded is at the back on the ground or on top of another order. However, when all items are allowed to be on top, the solution may have horizontal layers of items for a single customer, which may result in additional handling during unloading. Therefore, we will use a distance parameter (δ_{ik}) defining the maximum depth an order can be loaded on top of the preceding order.

The heuristics for CLP can be categorized under placement (construction) and improvement heuristics (Zhao et al., 2016). Placement heuristics, whether if it uses predetermined or dynamic ordering, decides on the arrangement of orders in a container. Improvement heuristics mostly use placement heuristics while constructing the initial solution or moving to another neighbourhood. Wall building and layer building are the most common approaches in placement heuristics. Wall building, which was originally proposed by George and Robinson (1980), initially selects the first box of each wall, then places the same type of box for the rest of the wall column by column, until there is no more space left in the wall or there is no more left of the selected box type. The length of box identifies the length of

the wall, and the box selection criteria changes in different stages of wall creation procedure. Based on George and Robinson (1980), Bischoff and Marriott (1990) tested 14 heuristic approaches with different ranking and filling methods, and finalize that there is no significant difference among tested approaches. Bischoff and Ratcliff (1995a) and Gehring et al. (1990) also tested different ranking approaches as well.

Layer building approach, which is relatively less studied in the literature, firstly creates base layer, then places new layers on top of base layer horizontally. The main problem in this approach is that the boxes may be placed unstable in the container. Loh and Nee (1992), Lodi et al. (2002) and Ratcliff and Bischoff (1998) follow a similar building strategy.

Improvement heuristics mainly aim at looking for a better solution than the solution found by a placement heuristic providing fast and reasonably quality solutions. In this sense, many metaheuristic approaches have been adopted; such as, genetic algorithm (Hemminki, 1994; Gehring and Bortfeldt, 1997; Wu et al., 2010), and tabu search algorithm (Bortfeldt et al., 2003; Bortfeldt and Gehring, 1998; Liu et al., 2011). Moura and Oliveira (2005) proposed a new algorithm based on greedy randomized adaptive search procedure (GRASP) for improvement of container loading problem. Parreño et al. (2008) also proposed a GRASP algorithm based on maximal space in the container, and achieve good quality solutions in short computing times and can improve them if longer times are available.

Christensen and Rousøe (2009) defined the length of a wall by tree search, and place items in the wall with a greedy algorithm, in order to cope with loading sequence and load bearing strength constraints. Similarly, Pisinger (2002) proposed a heuristic based on wall building approach, combined with tree-search heuristic, which selects the best set of layer depths and strip widths, and finds the optimal

solution by solving a Knapsack Problem (KP), once the layer and strip is selected.

5.3 Mathematical Model

In this part, we propose a mathematical model to CLP based on Junqueira et al. (2011) and Junqueira et al. (2012) with stability, loading sequence and stacking assumptions.

Indices and Sets

K Set of orders
 I Set of box types

Parameters

b_{ik} Number of boxes for order k of box type i
 P_i Weight of box type i
 L, W, H Dimensions of the vehicle
 X, Y, Z Possible positions along axes L, W and H
 l_i, w_i, h_i Dimensions of the box type i
 δ_{ik} The distance allowed for order k having box type i between boxes of consecutive destinations
 σ_i Load bearing strength of box type i
 M Very big number

Decision Variables

a_{ikxyz} $\begin{cases} 1 & \text{if a box of type } i \text{ from destination } k \text{ is placed with its front-left-} \\ & \text{bottom corner at position } (x, y, z) \\ 0 & \text{otherwise} \end{cases}$
 L_k The necessary length on axis x to load all boxes of order k

$$\text{Maximize } 0 \quad (5.1)$$

$$\text{Subject to } \sum_{i \in I} \sum_{k \in K} \sum_{\substack{x \in X; \\ x' - l_i + 1 \leq x \leq x'}} \sum_{\substack{y \in Y; \\ y' - w_i + 1 \leq y \leq y'}} \sum_{\substack{z \in Z; \\ z' - h_i + 1 \leq z \leq z'}} a_{ikxyz} \leq 1 \quad \forall x' \in X, y' \in Y, z' \in Z \quad (5.2)$$

$$\sum_{x \in X} \sum_{y \in Y} \sum_{z \in Z} a_{ikxyz} \geq b_{ik} \quad i \in I, k \in K \quad (5.3)$$

$$\sum_{\substack{j \in I; \\ z - h_j \geq 0}} \sum_{\substack{k' \in K; \\ k' \leq k}} \sum_{\substack{x \in X; \\ x \leq x' + l_i - 1}} \sum_{\substack{y \in Y; \\ y \leq y' + w_i - 1}} \bar{L}_{ij} \cdot \bar{W}_{ij} \cdot a_{jk'xy(z' - h_j)} \geq l_i \cdot w_i \cdot a_{ikx'y'z'} \quad \forall i \in I, k \in K, x' \in X, y' \in Y, z' \in Z \quad (5.4)$$

$$(x + l_i) \cdot a_{ikxyz} \leq L_k \quad \forall i \in I, k \in K, x \in X, y \in Y, z \in Z \quad (5.5)$$

$$L_{k-1} - \delta_{ik} \leq x \cdot a_{ikxyz} + (1 - a_{ikxyz}) \cdot M \quad \forall i \in I, k \in K : k \geq 2, x \in X, y \in Y, z \in Z \quad (5.6)$$

$$L_{k-1} \leq L_k \quad \forall k \in K, k \geq 2 \quad (5.7)$$

$$\sum_{i \in I} \sum_{k \in K} \sum_{\substack{x \in X; \\ x' - l_i + 1 \leq x \leq x'}} \sum_{\substack{y \in Y; \\ y' - w_i + 1 \leq y \leq y'}} \sum_{\substack{z \in Z; \\ z' + 1 \leq z \leq H - h_i}} \frac{P_i}{l_i \cdot w_i} a_{ikxyz} \leq \sum_{i \in I} \sum_{k \in K} \sum_{\substack{x \in X; \\ x' - l_i + 1 \leq x \leq x'}} \sum_{\substack{y \in Y; \\ y' - w_i + 1 \leq y \leq y'}} \sum_{\substack{z \in Z; \\ z' - h_i + 1 \leq z \leq z'}} \sigma_i \cdot a_{ikxyz} \quad \forall x' \in X, y' \in Y, z' \in Z \quad (5.8)$$

$$a_{ikxyz} \in \{0, 1\} \quad \forall x \in X, y \in Y, z \in Z, i \in I, k \in K \quad (5.9)$$

$$L_k \geq 0 \quad \forall k \in K \quad (5.10)$$

For constraint set (3), note that;

$$\bar{L}_{ij} = \min\{x' + l_i, x + l_j\} - \max\{x', x\}$$

$$\bar{W}_{ij} = \min\{y' + w_i, y + w_j\} - \max\{y', y\}.$$

The problem defined here is a feasibility problem, thus the objective function is constant and zero. Constraint set (5.2) avoids items to occupy the same space. Constraint set (5.3) ensures that all items are placed in the vehicle. Constraint set (5.4) satisfy unloading sequence and stability of the items. Constraint set (5.5) defines the distance of each order group to axis X (or the wall length of order k). Constraint set (5.6) is for the cases, where the items of an order is loaded on top of a consecutive delivery, and ensures that the inwards distance does not exceed allowed arm length to prevent difficulties in unloading. Constraint set (5.7) ensures that prior deliveries are close to the back of the vehicle, and satisfy LIFO requirement. Constraint set (5.8) satisfy the required amount of weight on top of the items regarding the load bearing strength for each type of item. Constraint sets (5.9) and (5.10) define the integrality constraints.

In the mathematical model, weight distribution and orientation assumptions are not included due to computational complexity. Yet, we may include those assumptions based on the mathematical model proposed by Chen et al. (1995). We coded model in GAMS solver, and tested it on randomly generated instances. Most of the instances are either solved in a short time, or infeasible. Additionally, model generation take quite a long time, as there are excessive number of decision variables. For instance, there are 20,000 decision variables for a vehicle with $L = 50$, $W = 20$ and $H = 20$. Thus, we can only solve the model on very small instances, and we cannot obtain a feasible solution in a reasonable computation time.

5.4 Heuristic Approach

In this section, we propose a heuristic algorithm (CLP) based on the wall building algorithm initially proposed by George and Robinson (1980), which also allows all type of orientations. The wall building algorithm creates walls along the length direction of the vehicle, and consecutively fills the walls. In our algorithm, we included the real life practices, which are orientation, stability, weight distribution and loading sequence assumptions. Stacking assumption is not included due to the necessity of a short running time for the algorithm.

Firstly the orders are sorted accordingly with the LIFO approach; thus the order to be delivered lastly is placed in the vehicle first. Secondly, the maximal bottom area of each box type is determined by considering each possible orientation and the boxes are sorted in decreasing order of maximal bottom area. Then the box type from top of the list is selected as the next item to be placed. The length of the recently placed box defines the length of the wall, and the boxes with the

same or similar size are placed consecutively to create the wall. Each placed box creates two empty spaces; one of which is on top and the other is next to the recently placed box. During the wall building, firstly the empty space on top is checked, whether if the next item to be placed fits. If not, the empty space next to the recently placed box is checked. The empty spaces, which cannot allocate any boxes, are assigned to the set of rejected spaces, and checked if it can be merged with the previously created rejected spaces. In such way, we create larger empty spaces.

By controlling the starting point of walls and empty spaces, we ensure that the real life constraints are satisfied. Orientation is satisfied by the algorithm itself while selecting the next box to be placed. Additionally, the stability constraint is satisfied during the placement by not allowing the box to be unsupported at any percentage from the bottom surface. As for the weight distribution constraint, we define the maximum weight that each compartment can take. When the maximum weight of a compartment is achieved, the current wall under construction is cut and the algorithm starts from a new wall starting from the next compartment. Additionally, the empty spaces in that compartment is cut up to the next compartment in order to prevent any new placements. With respect to the loading sequence, we restrict the wall building algorithm in such a way that the last order to be delivered is assigned to the first wall until all boxes of that order is placed. Then, remaining orders are placed to the last possible wall, so that the latter deliveries are placed at the back of prior deliveries. Although each box of an order needs to be loaded together, the order of previous delivery can be loaded to the empty space on top of or next to the latter delivery. For such cases, the only necessity is the ease of unloading operation. Therefore, we added a limit of arms length, which ensures that each box is within a reachable limit, thus the delivery can be performed easily. Respectively, when the algorithm starts to place a new

order, we rearrange the starting point of the empty spaces, and delete the empty spaces before the arms length limit.

Algorithm 5.1 CLP Algorithm

Initialization: Sort box types in order of farthest destination and decreasing maximal bottom area.

Main Step: Until no boxes or no space left, repeat the following steps;

Step 1: *Wall building*

(1a) Choose next box type to start filling the wall. The length of the wall is the length of the box in x direction of its chosen orientation.

(1b) Fill wall until no more space left or no boxes left of the chosen type.

(1c) Generate empty spaces to be filled in **Step 2**.

(1d) If weight limit is reached for the current compartment cut rejected and empty spaces from the beginning point of (with respect to x axis) next compartment. Set next wall's beginning point to next compartment.

(1e) If there is no more box left of the chosen type choose the next type of box.

(1f) If there is no more box left of the current destination move to the set of boxes of the next destination and cut rejected and empty spaces according to the arm's length parameter. Set next wall's beginning point to next compartment.

(1g) Go to **Step (2)**..

Step 2: *Space filling:* Until no empty space left, repeat the following steps

(2a) Choose next box type to start filling the space. It is the first box fitting to the space in the ordered set of boxes. If no box type fits in the space add space to rejected spaces.

(2b) Fill space until no more space left or no boxes left of the chosen type.

(2c) Generate empty spaces.

(2d) If weight limit is reached for the current compartment cut rejected and empty spaces from the beginning point of (with respect to x axis) next compartment. Set next wall's beginning point to next compartment.

(2e) If there is no more box left of the chosen type move to the next type of box.

(2f) If there is no more box left of the current destination move to the set of boxes of the next destination and cut rejected and empty spaces according to the arm's length parameter. Set next wall's beginning point to next compartment.

(2g) Join empty and rejected spaces to generate larger spaces.

5.5 Computational Experiments

In this section, we describe and detail the experiments on the aforementioned CLP algorithm. Initially, we test the performance of the algorithm on vehicle utilization by using the instances of Bischoff and Ratcliff (1995a), and compare the utilization and CPU performances of our algorithm with Moura and Oliveira

(2005). Then, we embed the CLP algorithm into the solution methodologies proposed in Chapter 3 and Chapter 4.

5.5.1 Preliminary Experiment

As a preliminary experiment, we tested the utilization rate of the proposed algorithm. In this sense, we coded the CLP algorithm in C environment and conducted the experiments using Intel Core i5-3230M with 2.6GHz and 4 GB RAM. The data set used in the experiments are well known test instances from Bischoff and Ratcliff (1995a). The instance set is decomposed of seven different levels with various box types. At each level, there are 100 instances and increasing number of box types. Such that, there are 3 different box types in *level BR1*, while *level BR7* has 20 different box types. Additionally, the number of items for each box type is more than 50 on the average. The total volume of an instance is almost as of the volume capacity of a container. Therefore, it is unlikely to obtain feasible solution for all instances, in which all boxes can be inserted. Yet, the experiments suggest a utilization rate for the solution methodology. In this respect, we compare the utilization and computation time performance of our CLP algorithm with the performances of the algorithm proposed by Moura and Oliveira (2005).

As shown in Table 5.1, experiments suggest that the utilization rate of our algorithm is 82% on the average, which is relative worse than those of Moura and Oliveira (2005). Yet, our algorithm outperforms Moura and Oliveira (2005) in terms of computation time. Considering that the CLP algorithm will be recalled excessive number of times when it is embedded to the solution methodologies of Chapter 3 and 4, computation time performance is more appreciated.

Table 5.1: Performances of CLP and Moura and Oliveira (2005)

	Utilization rates (%)	
	CLP	Moura and Oliveira (2005)
BR1	80.6	89.1
BR2	81.7	90.4
BR3	82.6	90.9
BR4	82.2	90.4
BR5	82.3	89.7
BR6	81.7	89.7
Ave. Utilization	81.9	90.0
Ave. CPU (sec)	0.3	172.1

5.5.2 SCD-TTFR with Loading Constraints

In this section, we conduct the computational experiments on the randomly generated data used in Chapter 3, including the dimensions of the boxes defined for each order. As for the LIFO requirement, we assign a delivery priority sequence for each order and transshipment terminal with respect to the distance of respective destination to origin. Additionally, we randomly define orientation restrictions for each order.

As for the acknowledged capacity of vehicles, we consider two capacities; (i) actual and (ii) accepted (utilized) vehicle capacities. For (i) actual vehicle capacity, we comply with the actual allowable weight limit and actual dimensions of a vehicle. For (ii) accepted (utilized) vehicle capacity, we are inspired from the real-life applications; freight forwarders' manner of making dispatching plans by considering less capacity than the actual capacity of the vehicles, especially for the volume. In this way, freight forwarders guarantee that the orders planned for the vehicle can fit. In this sense, we apply actual vehicle capacities in CLP algorithm as it can utilize 82% on the average. During VNS algorithm, we also check if any

possible change is feasible in terms of capacities, yet this time we use the accepted (utilized) vehicle capacities in order to guarantee the feasibility. To this end, we applied accepted capacity as 80% of the actual capacity just for the volume by adjusting only the height of the vehicle and keeping the other dimensions and weight at their allowed maximum.

As a reminder for the reader, proposed VNS algorithm for SCD-TTFR iteratively searches the increasing neighborhoods of the incumbent solution, and moves to a new solution if any improvement is achieved. As for the interaction of VNS and CLP algorithms, we call CLP if any improvement is observed after local search. CLP checks if this new solution is feasible in terms of container loading constraints, and returns 1 to VNS algorithm if the new solution is feasible, or 0 if it is infeasible. If the returned value is 1, then VNS continues by accepting the new solution as the incumbent one. If the returned value is 0, the VNS algorithm rejects the new solution and continues with the incumbent solution.

In order to decrease the number of times that CLP is called, we only check the feasibility of vehicles which are altered during the *shaking procedure*. For instance, if VNS applies a *move* operation, and moves one order from a vehicle to another one, then only two vehicles have substantial changes. Assuming that the remaining vehicles are previously checked and confirmed in terms of loading constraints, we do not check their feasibility, and call CLP only for the altered two vehicles. In this way, we attain savings in computation time.

We initially analyze the vehicle utilization rates of VNS with and without CLP in Table 5.2 including the number of vehicles, the utilization rates (in %) considering the volume (V), weight (W) and length (L) capacity of the vehicles. Paired sample t-test suggests that the number of vehicles in the obtained solution increases

Table 5.2: Utilization Rates of VNS without and with CLP

I	E	#	VNS without CLP				VNS with CLP		
			# opt	# of veh	Gap (%)	CPU (sec)	# opt	# of veh	CPU (sec)
10	0	30	30	7.2	0.0	5.3	28	7.2	5.3
	1	30	29	5.2	0.0	5.0	26	5.3	5.0
20	0	30	29	10.4	0.0	16.5	28	10.4	16.5
	1	30	25	7.1	0.4	5.5	13	7.3	5.5
30	0	30	27	14.0	0.0	31.8	25	14.1	31.8
	1	30	1	10.7	4.0	12.8	1	10.9	12.8
50	0	30	12	20.6	1.4	65.1	7	20.7	65.1
	1	30	0	17.2	6.6	41.6	0	17.5	41.6
100	0	30	0	36.2	9.0	306.3	0	36.4	306.3
	1	30	0	32.5	6.7	299.7	0	32.9	299.7
Grand Total		300	153	16.1	2.8	79.0	128	16.3	79.0

significantly, when CLP is embedded to VNS algorithm. Additionally, the average utilization increases parallel to the increase in number of orders, and there is a negative correlation between the number of vehicles and utilization rates as expected. The elasticity of orders for departure also has a positive effect on utilization rate. In other words, if there is a possibility to await an order for another day, there may be more vehicles to accommodate this order. In this sense, both larger instances and instances with elastic departure dates have more possible movements. Therefore, the utilization rate increases if there are more alternative dispatching plans for both algorithms; hence, less number of vehicles are needed to depart all orders.

Inspired by the above findings, we examine the utilization rate of rejected vehicles and number of times that CLP rejected during VNS. Table 5.3 shows the rejection rate and utilization of rejected vehicles in terms of volume, weight and length. The rejection rate indicates the proportion of total number of times CLP returned infeasible to the total number of times CLP is called. As the possibility to move

an order to another vehicle increases (either because of increasing number of orders or elasticity for departure), the number of times improvements, thus CLP is called, increases.

Table 5.3: Utilization Rate of Rejected Vehicles by CLP

		Utilization rate of Rejected Vehicles (%)									
		Rejection	Volume			Weight			Length		
I	E	Rate (%)	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max
10	0	14.7	80.1	88.3	95.2	40.0	55.7	63.3	77.6	83.1	94.3
	1	15.3	67.2	90.1	99.7	22.7	45.8	59.2	64.4	84.9	96.0
20	0	6.9	87.5	95.4	99.6	27.0	48.6	91.0	80.9	90.7	98.3
	1	13.5	78.6	94.3	99.9	30.4	63.5	95.9	74.9	90.1	98.8
30	0	8.0	59.4	91.8	99.4	26.4	60.4	90.4	57.2	85.1	95.6
	1	9.7	59.1	93.0	99.7	26.3	62.2	96.3	58.4	87.8	99.3
50	0	10.6	81.6	93.1	99.7	18.0	65.0	91.2	74.5	88.1	98.8
	1	15.4	84.4	94.4	99.8	31.2	63.1	94.7	77.9	88.9	98.6
100	0	15.6	77.0	92.7	99.4	33.7	65.4	90.8	71.0	87.9	98.0
	1	15.3	88.1	96.5	99.9	23.9	63.1	90.6	82.1	91.8	99.9
Average		12.7	59.1	93.6	99.9	18.0	61.8	96.3	57.2	88.5	99.9

When the utilization rate of rejected vehicles is concerned, the major source of rejection seems to be the volume and length capacity, while weight capacity doesn't have a significant effect. We assume that such an outcome is specific to the characteristics of generated instances, and can not be generalized. Considering the utilization rates, we suggest that the vehicles with capacity over 90% will most likely be rejected by CLP algorithm.

Table 5.4 shows the performances of both algorithm including the number of instances solved to optimality, average gap and CPU time. The number of instances that are optimally solved decreases when CLP is embedded to VNS. In addition to increase in suboptimal solutions, the gap increases 1% on the average, and the highest increase in gap is observed in the larger instances. On behalf of

Table 5.4: CPU and Quality Performances of VNS without and with CLP

I	E	#	VNS without CLP			VNS with CLP		
			# opt	Gap (%)	CPU (sec)	# opt	Gap (%)	CPU (sec)
10	0	30	30	0.0	5.3	28	0.6	9.9
	1	30	29	0.0	5.0	26	0.8	9.6
20	0	30	29	0.0	16.5	28	0.1	23.4
	1	30	25	0.4	5.5	13	2.4	6.7
30	0	30	27	0.0	31.8	25	0.5	33.9
	1	30	1	4.0	12.8	1	5.1	14.8
50	0	30	12	1.4	65.1	7	1.7	66.7
	1	30	0	6.6	41.6	0	8.0	46.6
100	0	30	0	9.0	306.3	0	9.7	252.8
	1	30	0	6.7	299.7	0	8.3	309.0
Average		300	153	2.8	79.0	128	3.7	77.3

computation time performance, average CPU does not have a significant increase, yet, paired sample t-test suggests that CPU time significantly increases for those instances that are solved to optimality.

5.5.3 SCD-TTSM with Bin Packing

In this section, we conduct the experiments on Branch-and-Price B&P algorithm with CLP with the instances used in Chapter 4. As for the necessary parameters of CLP algorithm, we use the orientation restrictions defined in the previous section. Additionally, we used the distance to the destination of orders to identify the delivery sequence. With the same assumption of Section 5.5.2, CLP algorithm uses actual vehicle capacity, and B&P algorithm accepts utilized vehicle capacity.

Figure 5.1 shows the CLP embedded B&P algorithm on a given node. As *column generation procedure* generates a new column, we call CLP algorithm and check if

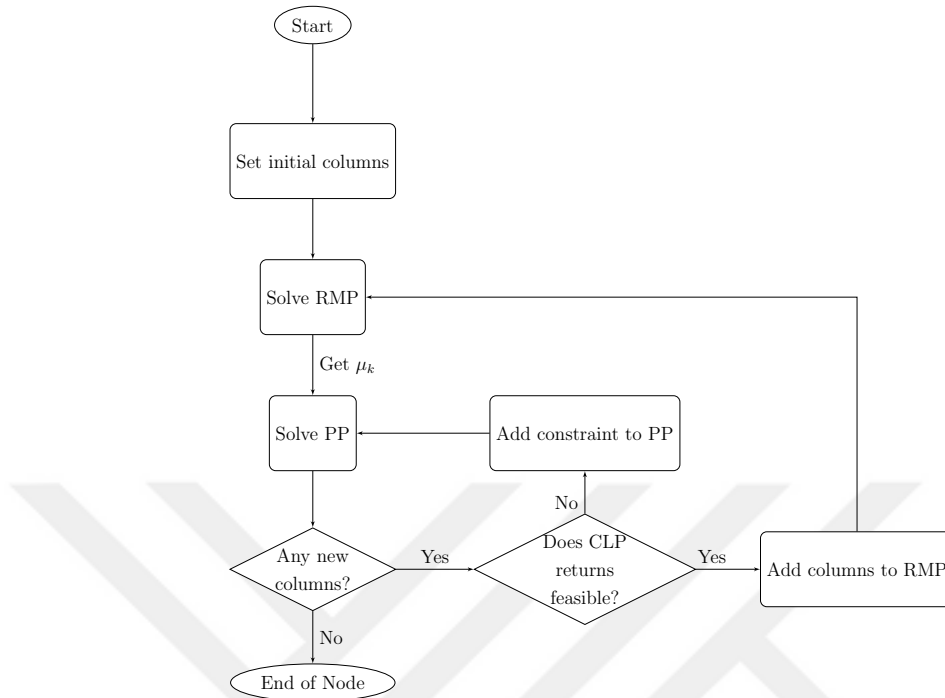


Figure 5.1: Column Generation Procedure with CLP Algorithm on a Given Node

the generated column is feasible in terms of loading constraints. If CLP algorithm returns “1”, then the recently generated column is feasible. Thus, we added the recently generated column to restricted master problem. If the returned value from CLP is “0”, then the recently added column is infeasible when loading constraints are concerned. In this case, we add a constraint to the pricing problem stating that the sum of x variables (which indicates if the order is loaded on the vehicle) should be less than the number of orders in that vehicle. Consequently, such a column with the orders of rejected vehicle cannot be generated at the further iterations, due to the recently added constraint. Additionally, we do not add the indicated infeasible column to restricted master problem, hence all orders in the rejected vehicle are no longer available to be loaded on the same vehicle.

In order to measure the effect of CLP algorithm accurately, we conduct compu-

tational experiments on small scale instances with $I = \{10, 20, 30\}$, which can be solved optimally within the determined time limit. For the instances that hit the specified time limit, we may not be able to estimate the effect on the computation time, number of added columns and vehicles in the optimal solution. Additionally, we may not compare the gap of sub-optimal instances, as the feasible regions of two algorithms are not the same.

Table 5.5: Number of added columns and CPU of B&P with and without CLP

I	B	B&P without CLP				B&P with CLP			
		#opt	# Cols	# Nodes	CPU (sec)	#opt	# Cols	# Nodes	CPU (sec)
10	10	30	13.9	0.5	0.1	30	13.5	0.5	0.2
	70	30	24.9	7.3	0.5	30	21.9	4.3	0.3
20	10	30	42.1	8.2	2.0	30	38.9	4.7	1.2
	70	30	137.3	110.8	11.2	30	117.8	142.2	8.4
30	10	30	79.0	84.7	20.6	30	70.7	43.9	15.1
	70	24	509.0	1542.7	297.5	29	290.4	372.1	69.1
Average		174	121.5	249.2	47.0	179	85.4	85.0	13.9

Table 5.5 shows the number of instances optimally solved, number of columns generated, number of visited nodes and computation time (in seconds). Surprisingly, paired t-test results suggests that B&P with CLP outperforms B&P in all reported indicators. If CLP returns “0”, thus the column generated is infeasible in terms of loading constraints, then column generation algorithm tries to generate a new column, which unlikely includes the orders with the highest dual prices from the restricted master problem. Eventually, the pricing problem may not be able to find a column with a negative reduced cost. In this sense, the column generation procedure is able to generate less columns in B&P with CLP. Deriving from the branching strategy, B&P visit less nodes, as there are less possible combinations of orders in a vehicle, due to loading constraints. With all these factors combined together, the CPU time required to solve the problem to optimality decreases, and the number of solutions solved optimally increases.

Table 5.6 displays the effect of CLP Algorithm on the deviation of cost from the value achieved by B&P without CLP and number of vehicles for the obtained solution. Additionally, the rejection rate indicates the percent of rejected columns to total number of times that pricing problem is solved. Although shown rejection rate is high, it may be misleading, especially for small scale instances. For example, B&P produces only one column for an instance with orders having dispersed release dates. When CLP rejects the only one generated column, the rejection rate will be 100%. However, if we examine the overall sums of rejected columns and number of times pricing problem is called, CLP rejected approximately 2.5% of all columns generated during all instances experimented.

Table 5.6: Effect of CLP on Number of Vehicles and Cost in the Optimal Solution

I	B	Rejection	Deviation	# of Vehicles	
		Rate (%)	(%)	B&P	B&P-CLP
10	10	36.8	5.9	7.2	7.6
	70	21.9	4.2	5.3	5.5
20	10	20.7	5.3	10.6	11.1
	70	19.1	5.5	7.2	7.6
30	10	11.6	4.7	14.2	14.8
	70	17.7	3.0	10.8	11.1
Average		20.2	4.7	9.3	9.7

As a result of rejected vehicles, the number of vehicles required to depart all orders increases. Thus, B&P with CLP departs more vehicles than B&P algorithm, and paired t-test result suggests that such an increase is significant. Respectively, the B&P with CLP obtains greater values for the same instance than B&P algorithm, and the deviation from the optimal value of B&P is approximately 5%.

Table 5.7 shows the utilization rate of B&P without and with CLP (in percent)

in terms of volume, weight and length. For all three examined dimensions, the utilization rate decreases approximately 2% on the average when the CLP algorithm is embedded to B&P. The utilization rate is less than expected, yet the experiments are conducted with instances having upto 30 orders, thus we assume that utilization would increase as the number of orders in an instance increase.

Table 5.7: Utilization Rate of B&P without and with CLP (%)

I	B	B&P without CLP			B&P with CLP		
		V	W	L	V	W	L
10	10	50.4	18.3	48.5	49.0	17.7	47.1
	70	68.2	24.7	65.7	67.0	24.1	64.6
20	10	55.2	26.2	53.4	53.2	25.1	51.4
	70	80.6	38.4	77.9	77.4	36.7	74.8
30	10	65.7	27.8	62.9	62.9	26.7	60.3
	70	86.2	36.7	82.6	84.1	36.0	80.6
Average		67.1	28.4	64.6	65.0	27.4	62.5

Table 5.8 shows the utilization rate of rejected vehicles in terms of volume, weight and length. Similar to the findings of Section 5.5.2, major source of rejection is likely to be volume or length capacity of vehicles as weight utilization of rejected vehicles is relatively low. Yet, it is specific to the characteristics of the instances as it is stated in the previous section. CLP performs better when there are more orders in an instance, or orders can await for departure.

Table 5.8: Utilization Rate of Rejected Vehiles by CLP Algorithm (%)

I	B	Volume			Weight			Length		
		Min	Ave	Max	Min	Ave	Max	Min	Ave	Max
10	10	1.8	42.0	85.5	3.1	18.1	63.3	1.8	40.4	77.9
	70	1.8	62.8	95.2	3.1	25.4	59.9	1.8	59.8	93.5
20	10	5.7	47.8	89.2	6.9	22.5	57.0	5.9	46.2	85.9
	70	39.3	81.1	95.8	17.7	51.5	74.8	35.8	77.9	94.6
30	10	29.7	67.2	95.1	7.9	35.8	72.1	29.3	63.7	88.4
	70	64.5	88.2	96.4	36.4	53.2	71.6	62.6	84.0	91.8
Average		1.8	66.2	96.4	3.1	35.3	74.8	1.8	63.3	94.6

5.6 Conclusion and Future works

In this chapter we develop a 3D-BPP Algorithm that controls if the generated vehicles are feasible in terms of real-life loading constraints, which is also known as CLP algorithm. The developed algorithm is based on the wall building algorithm proposed by George and Robinson (1980). Then we integrate the developed algorithm to both solution methodologies in Chapter 3 and Chapter 4.

In order to test the time and utilization performance of CLP algorithm, we tested the algorithm on the well known bin-packing instances of Bischoff and Ratcliff (1995a). The experiment findings suggest that the utilization rate is approximately 82%, yet our algorithm can produce solution in a very short computation time, 0.3 seconds on the average. If we consider that the CLP algorithm is called several times when it is embedded to both algorithms, computation time performance takes on critical importance. Utilization rate of the test runs is relatively lower than the ones reported in the literature. Yet it would be sufficient enough to create feasible solutions as we assume that the industry utilizes the vehicle

capacities around 80%, and accepts utilized vehicle capacity as such.

While incorporating the CLP algorithm to the solution methodology in Chapter 3, namely VNS, we call CLP algorithm each time we observed an improvement after local search procedure. If all of the altered vehicles are feasible in terms of loading constraints, then the algorithm moves to the new solution. If not, the algorithm rejects the incumbent solution and continues with the rejected one. For B&P, we call CLP algorithm each time pricing problem generates a new column. If CLP indicates that recently generated column is feasible, then it is added to restricted master problem. If the new column is infeasible, then the algorithm adds a constraint, which disables the orders in the generated column to be on the same vehicle all together.

When we examine the results of computational experiments, the embedded CLP algorithm increases the computation time to solve instances to optimality for VNS, while decreasing computation time for B&P. Such diverse effect on computation time of CLP originates from the different structures of VNS and B&P. Firstly, we do not have a tabu list for the vehicles that are detected infeasible by CLP algorithm. Additionally, VNS continues to search solution space until the defined iteration limit is achieved, and without a tabu list, some infeasible vehicles may be generated recurrently and duplicate calls of CLP may be possible. As a consequence of redundant calls of CLP and the obligatory iteration limit, computation time increases. On the other hand, B&P continues to hold a record of infeasible vehicles by keeping the constraints added to pricing problem. As the number of possible combinations of orders decrease with the constraints disabling infeasible vehicles to be generated again, so does the number of nodes to be visited. Considering that B&P stops when all nodes are visited, and there are less nodes to visit when CLP is embedded; hence, computation time decreases.

When we examine the change in number of vehicles and utilization rate after the CLP algorithm is embedded, the results of experiments suggests that CLP impaired both performances. Due to rejected vehicles, utilization rate decreases, and the number of vehicles in the obtained solution increases. As a consequence, solution quality decreases for both algorithms. For VNS with CLP, the number of optimal solutions decrease and the average increase in gap is around 1%. For B&P with CLP, the average deviation from the optimal solution of B&P without CLP is the approximately 5%.

Although utilization rate of CLP is sufficient enough to meet real-life expectations, deriving from the above findings, CLP still leads to inefficient use of vehicles, and thus poor solution quality. Therefore, we assume two different approaches for further research directions. Firstly, both algorithms may be tested on instances having small boxes in order to find the major source of low utilization rate. For instance, weight utilization is relatively lower than the volume and length utilization, yet we assume that such a result is specific to our instances. However, we cannot foresee if volume and length utilization rates are also low because of instance characteristics. Secondly, CLP algorithm may be improved to increase the utilization rate. As another further research, a tabu list for VNS algorithm, which keeps the infeasible vehicles may also be incorporated to decrease the rejection rate and reduce effect of CLP on computation time.

Chapter 6

Conclusion and Future Research

In this chapter, we will summarize the main focus of this thesis, applied methodology and findings and conclude with the future research objectives.

In this thesis, we aim at providing new solution methodologies for a real-life problem of freight forwarders providing long haul freight transportation and LTL service. The main focus is to consolidate LTL orders with time-windows. The orders may be delivered either to their destinations directly or by using transshipment terminals. The route of a vehicle is defined by the farthest destination in the vehicle, and cost is associated with that farthest destination. If contracted vehicles are used, then the predefined routes, which are indicated in the annual contracts, determines the cost of vehicle by considering the farthest destination. If spot market vehicles are used, then the cost is subject to the distance to the farthest destination and an operation cost. In either way, the main objective of freight forwarders is to minimize the total cost of used vehicles and the cost using transshipment terminal. Eventually, such a real life problem is different from the ones in the literature in two ways; (i) cost structure and (ii) delivery structure. In this context, we propose two different problems, which have similar assumptions

of loading constraints and delivery structures, yet having different cost structures.

The first problem having fixed and predefined routes, respectively fixed costs for those routes, is commonly observed in annual contracts between freight forwarders and vehicle owners. We call this problem as Shipment Consolidation and Dispatching with Fixed Routes (SCD-TTFR). We initially develop a mathematical model and two lower bound algorithms, then propose a Variable Neighborhood Search (VNS) Algorithm for such a problem.

We added 3 types of constraints to the proposed mathematical model in order to enhance the performance, which are symmetry breaking constraint, defining an upper bound for M and direct relation between order assignment to vehicle and vehicle departure. Computational experiments show that all three enhancements provides better solutions. We also conduct a computational experiment to indicate the effect of number of routes, and tested the enhanced mathematical model with different number of routes. The results show that a larger route set with more predefined routes brings about an increase in computation time. When we compare VNS algorithm with the mathematical model, VNS outperforms mathematical model in terms of computation time, and VNS provides approximately the same solution quality as of the mathematical model. Both number of orders in an instance and elasticity of orders to await for departure affects the performance of VNS algorithm on solution quality and computation time.

As for the further research directions for Chapter 3, we may improve the quality of lower bound by applying another algorithm. Additionally, the performance of VNS can be enhanced by increasing the iteration limit and including more operations to shaking procedure.

The second problem is mainly faced when there are no contracts between freight

forwarder and vehicle owner, and the cost of a vehicle is determined by the spot market prices. Therefore, the cost of a vehicle is defined by the farthest destination in the vehicle and excessive deviations along the route to that farthest destination is not allowed. We call the second problem as Shipment Consolidation and Dispatching with Spot Market Prices (SCD-TTSM). We develop a mathematical model for the problem, then applied Dantzig-Wolfe Decomposition approach and a Branch-and-Price (B&P) algorithm, in which a column generation procedure based on the decomposed problem is called iteratively.

The computational experiments on the solution methodology of Chapter 4 show that B&P algorithm provides better lower bound than original formulation (OF), and outperforms OF in terms of both computation time and solution quality. The experiment results show that the performance of algorithm changes under different problem characteristics. The number of visited nodes after root node and number of generated columns substantially increases as the the number of instances in an order increases. Similarly, the algorithm visits more nodes and produces more columns when there is an elasticity to await orders. As there are more possible combinations of orders for larger instances or instances having elastic orders, such an increase on number of generated columns and visited nodes is expected. On the other hand, the algorithm can find the optimal solution at a very early stage of iterations, yet it continues to add columns, which are redundant. Such an outcome shows that B&P can find near optimal solutions in a short computation time.

As the number of instances increases, the quality of lower bounds increase, and the number of nodes to be visited increases substantially. As a future study, we can improve the lower bound by changing the search strategy to depth-first or mixed breadth-depth traversing algorithm, as it is proposed by Fayed and Atiya

(2013). Additionally, in order to enhance the upper bound performance for large scale instances, we may apply fast and greedy solution approaches to solve pricing problem as another extension.

In Chapter 5, we develop a mathematical model and a heuristic algorithm, which is based the wall building algorithm of George and Robinson (1980), and includes the real life constraints, such as orientation, stability, weight distribution, loading sequence and stacking assumptions. Basically the heuristic algorithm checks if those constraints are satisfied in the generated solution, thus it is a heuristic for a commonly known problem; Container Loading Problem (CLP). We integrated the CLP Algorithm to both algorithms proposed for the two problems.

Initially, we tested the performance of our heuristic algorithm on the test instances of Bischoff and Ratcliff (1995a), and results show that the utilization rate (approximately 82 %) and computation time (0.3 seconds on the average) is acceptable enough to meet the utilization requirement of industry, and the requirement of a quick response in order to integrate the heuristic to both algorithms. Afterwards, we tested the performance of CLP integrated VNS and B&P algorithms on the randomly generated instances of Chapter 3. We examine that utilization rate and the number of vehicles are impaired after CLP is embedded to VNS and B&P. As a result, both solution quality decreases, hence for VNS with CLP, the number of optimal solutions decrease and the average increase in gap is around 1%. For B&P with CLP, the average deviation from the optimal solution of B&P without CLP is the approximately 5%. As for the computation time performance, the embedded CLP algorithm increases the computation time to solve instances to optimality for VNS, while decreasing computation time for B&P.

As for the future works of Chapter 5, we may test the CLP embedded algorithms with small boxes in order to see if the major source of low utilization rate is the

characteristics of boxes. Additionally, we may improve CLP algorithm to increase the utilization rate. As another further research, we may also incorporate a tabu list for VNS algorithm, which keeps the infeasible vehicles. In this way, we expect to decrease the rejection rate and reduce effect of CLP on computation time.

In this thesis, we focus on two problems separately, yet it is possible to take the problem together under some circumstances. For instance, when there is high season, it may be a good idea to use contracted vehicles as the prices indicated in the annual contracts will most likely be relatively lower than the spot market prices. The other way around, when it is low season, spot market prices will be lower than the annual contracted prices, hence vehicles of spot market will be favorable. In this sense the freight forwarder should decide the type of vehicle to be used, either contracted vehicles, or vehicles of spot market. In this case, we assume that both types of problems can be managed under B&P algorithm, by defining two types of columns, each representing a problem type. For SCD-TTSM, the transition to a problem with types of columns would be straightforward, yet decomposition of SCD-TTFR requires further attention.

In addition to the second type of pricing problem, the pricing problem may be formulated as Prize Collecting Open Traveling Salesman Problem (PC O-TSP). In this way, more accurate routes may be generated, yet the computational complexity, thus computation time required to solve pricing problem to optimality increases.

The tactical level decisions in the stated problem may be another research direction. Especially the contracted transshipment terminals plays a significant role in consolidation decisions in terms of location and prices defined in the contracts. Considering the number of contracted transshipment terminals and their locations, both will affect the routing decisions, whereas the prices of the contracted

terminals will influence the delivery terms of orders (either direct delivery or delivery from a transshipment terminal). In this sense, enhancements on both tactical decisions may benefit freight forwarders in the long run.

In conclusion, the analytical solution methodologies that we propose throughout the thesis produces good quality solutions in a very short computation time. Additionally, we tested proposed solution methodologies on a real-life instance, and the obtained solutions provide approximately 10% savings for both algorithms, and are applicable in real life. In this sense, proposed solution methodology may benefit international freight forwarders in three perspectives; (i) cost savings, (ii) efficient use of human resources, and (iii) time utilization.

Bibliography

- Abdelwahab, W. M. and Sargious, M. (1990). Freight rate structure and optimal shipment size in freight. *Logistics and Transportation Review*, 26(3):271.
- Andersen, J., Christiansen, M., Crainic, T. G., and Grønhaug, R. (2011). Branch and price for service network design with asset management constraints. *Transportation Science*, 45(1):33–49.
- Attanasio, A., Fuduli, A., Ghiani, G., and Triki, C. (2007). Integrated shipment dispatching and packing problems: a case study. *Journal of Mathematical Modelling and Algorithms*, 6(1):77–85.
- Azi, N., Gendreau, M., and Potvin, J.-Y. (2010). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202(3):756–763.
- Barcos, L., Rodríguez, V., Álvarez, M. J., and Robusté, F. (2010). Routing design for less-than-truckload motor carriers using ant colony optimization. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):367–383.
- Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329.

- Barnhart, C. and Schneur, R. R. (1996). Air network design for express shipment service. *Operations Research*, 44(6):852–863.
- Baykasoglu, A. and Kaplanoglu, V. (2011). A multi-agent approach to load consolidation in transportation. *Advances in Engineering Software*, 42(7):477–490.
- Berger, J., Barkaoui, M., and Braysy, O. (2003). A route-directed hybrid genetic approach for the vehicle routing problem with time windows. *Information Systems and Operational Research*, 41(2):179–194.
- Beuthe, M. and Kreuzberger, E. (2008). Consolidation and trans-shipment. In *Handbook of Logistics and Supply-Chain Management*, pages 239–252. Emerald Group Publishing Limited.
- Bischoff, E. (2006). Three-dimensional packing of items with limited load bearing strength. *European Journal of Operational Research*, 168(3):952–966.
- Bischoff, E. and Ratcliff, M. (1995a). Issues in the development of approaches to container loading. *Omega*, 23(4):377 – 390.
- Bischoff, E. E. (1991). Stability aspects of pallet loading. *Operations-Research-Spektrum*, 13(4):189–197.
- Bischoff, E. E. and Marriott, M. D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research*, 44(2):267 – 276.
- Bischoff, E. E. and Ratcliff, M. (1995b). Issues in the development of approaches to container loading. *Omega*, 23(4):377–390.
- Bookbinder, J. H. and Higginson, J. K. (2002). Probabilistic modeling of freight consolidation by private carriage. *Transportation Research Part E: Logistics and Transportation Review*, 38(5):305–318.

- Bortfeldt, A. and Gehring, H. (1998). *Applying Tabu Search to Container Loading Problems*, pages 533–538. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Bortfeldt, A., Gehring, H., and Mack, D. (2003). A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5):641 – 662. Parallel computing in logistics.
- Bortfeldt, A. and Wäscher, G. (2013). Constraints in container loading—a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20.
- Brennan, J. J. (1981). Models and analysis of temporal consolidation. *Dissertation Abstracts International Part B: Science and Engineering*[DISS. ABST. INT. PT. B- SCI. & ENG.], 42(1):1981.
- Campbell, J. F. (1990). Freight consolidation and routing with transportation economies of scale. *Transportation Research Part B: Methodological*, 24(5):345–361.
- Ceschia, S. and Schaerf, A. (2013). Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, 19(2):275–294.
- Çetinkaya, S. (2005). Coordination of inventory and shipment consolidation decisions: A review of premises, models, and justification. In *Applications of supply chain management and e-commerce research*, pages 3–51. Springer.
- Çetinkaya, S. and Lee, C.-Y. (2000). Stock replenishment and shipment scheduling for vendor-managed inventory systems. *Management Science*, 46(2):217–232.
- Çetinkaya, S., Mutlu, F., and Wei, B. (2014). On the service performance of alternative shipment consolidation policies. *Operations Research Letters*, 42(1):41–47.

- Chen, C., Lee, S.-M., and Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76.
- Christensen, S. G. and Rousøe, D. M. (2009). Container loading with multi-drop constraints. *International Transactions in Operational Research*, 16(6):727–743.
- Closs, D. J. and Cook, R. L. (1987). Multi-stage transportation consolidation analysis using dynamic simulation. *International Journal of Physical Distribution & Materials Management*, 17(3):28–45.
- Coffman Jr, E. G., Garey, M. R., and Johnson, D. S. (1996). Approximation algorithms for bin packing: a survey. In *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co.
- Crainic, T. G. (1984). A comparison of two methods for tactical planning in rail freight transportation. In *Operational Research '84: Proceedings*.
- Crainic, T. G. (2000). Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288.
- Crainic, T. G. (2003). *Long-haul freight transportation*. Springer.
- Crainic, T. G. and Laporte, G. (1997). Planning models for freight transportation. *European Journal of Operational Research*, 97(3):409–438.
- Crainic, T. G. and Rousseau, J.-M. (1986). 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.
- Cunha, C. B. and Silva, M. R. (2007). A genetic algorithm for the problem of configuring a hub-and-spoke network for a ltl trucking company in brazil. *European Journal of Operational Research*, 179(3):747–758.

- Daganzo, C. F. (1988). Shipment composition enhancement at a consolidation center. *Transportation Research Part B: Methodological*, 22(2):103–124.
- Danna, E. and Pape, C. (2005). Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In *Column Generation*, pages 99–129. Springer.
- Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.
- Davies, A. P. and Bischoff, E. E. (1999). Weight distribution considerations in container loading. *European Journal of Operational Research*, 114(3):509–527.
- Desrosiers, J., Dumas, Y., and Soumis, F. (1986). A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6(3-4):301–325.
- Desrosiers, J. and Lübbecke, M. E. (2005). A primer in column generation. In *Column Generation*, pages 1–32. Springer.
- Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22.
- Erera, A., Hewitt, M., Savelsbergh, M., and Zhang, Y. (2013). Improved load plan design through integer programming based local search. *Transportation Science*, 47(3):412–427.
- Estrada, M. and Robusté, F. (2009). Long-haul shipment optimization for less-than-truckload carriers. *Transportation Research Record: Journal of the Transportation Research Board*, (2091):12–20.
- Fayed, H. A. and Atiya, A. F. (2013). A mixed breadth-depth first strategy

- for the branch and bound tree of euclidean k-center problems. *Computational Optimization and Applications*, 54(3):675–703.
- Fischetti, M., Toth, P., and Vigo, D. (1994). A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42(5):846–859.
- Fu, Z., Eglese, R., and Li, L. Y. (2005). A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society*, 56(3):267–274.
- Galbreth, M. R., Hill, J. A., and Handley, S. (2008). An investigation of the value of cross-docking for supply chain management. *Journal of Business Logistics*, 29(1):225–239.
- Gehring, H. and Bortfeldt, A. (1997). A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 4(5-6):401–418.
- Gehring, H., Menschner, K., and Meyer, M. (1990). A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, 44(2):277 – 288.
- Gendreau, M., Iori, M., Laporte, G., and Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350.
- George, J. and Robinson, D. (1980). A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147 – 156.
- Ghiani, G., Laporte, G., and Musmanno, R. (2004). *Introduction to Logistics Systems Planning and Control*. John Wiley & Sons.

- Guastaroba, G., Speranza, M. G., and Vigo, D. (2016). Intermediate facilities in freight transportation planning: a survey. *Transportation Science*, 50(3):763–789.
- Gümüş, M. and Bookbinder, J. H. (2004). Cross-docking and its implications in location-distribution systems. *Journal of Business Logistics*, 25(2):199–228.
- Gupta, Y. P. and Bagchi, P. K. (1987). Inbound freight consolidation under just-in-time procuremen. *Journal of Business Logistics*, 8(2):74.
- Hall, R. W. (1987a). Consolidation strategy: inventory, vehicles and terminals. *Journal of Business Logistics*, 8(2):57.
- Hall, R. W. (1987b). Direct versus terminal freight routing on a network with concave costs. *Transportation Research Part B: Methodological*, 21(4):287–298.
- Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.
- Harks, T., König, F. G., Matuschke, J., Richter, A. T., and Schulz, J. (2014). An integrated approach to tactical transportation planning in logistics networks. *Transportation Science*, 50(2):439–460.
- Hemminki, J. (1994). Container loading with variable strategies in each layer. Technical report, University of Turku, Institute of Applied Mathematics.
- Hemminki, J., Leipala, T., and Nevalainen, O. (1998). On-line packing with boxes of different sizes. *International Journal of Production Research*, 36(8):2225–2245.
- Higginson, J. K. and Bookbinder, J. H. (1994). Policy recommendations for a shipment-consolidation program. *Journal of Business Logistics*, 15(1):86–113.

- Higginson, J. K. and Bookbinder, J. H. (1995). Markovian decision processes in shipment consolidation. *Transportation Science*, 29(3):242–255.
- Higginson, J. K. and Bookbinder, J. H. (2005). Distribution centres in supply chain operations. In *Logistics Systems: Design and Optimization*, pages 67–91. Springer.
- Irnich, S. (2002). *Netzwerk-Design für zweistufige Transportsysteme und ein Branch-and-Price-Verfahren für das gemischte Direkt-und Hubflugproblem*. PhD thesis, Bibliothek der RWTH Aachen.
- Jarrah, A. I., Johnson, E., and Neubert, L. C. (2009). Large-scale, less-than-truckload service network design. *Operations Research*, 57(3):609–625.
- Junqueira, L., Morabito, R., and Yamashita, D. S. (2011). Mip-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research*, 199(1):51–75.
- Junqueira, L., Morabito, R., and Yamashita, D. S. (2012). Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85.
- Khebbache-Hadji, S., Prins, C., Yalaoui, A., and Reghioui, M. (2013). Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. *Central European Journal of Operations Research*, 21(2):307–336.
- Kim, D., Barnhart, C., Ware, K., and Reinhardt, G. (1999). Multimodal express package delivery: A service network design application. *Transportation Science*, 33(4):391–407.
- Koca, E. and Yıldırım, E. A. (2012). A hierarchical solution approach for a

- multicommodity distribution problem under a special cost structure. *Computers & Operations Research*, 39(11):2612–2624.
- Kuo, Y. (2010). Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, 59(1):157–165.
- Lee, Y. H., Jung, J. W., and Lee, K. M. (2006). Vehicle routing scheduling for cross-docking in the supply chain. *Computers & Industrial Engineering*, 51(2):247–256.
- Lin, J.-L., Chang, C.-H., and Yang, J.-Y. (2006). A study of optimal system for multiple-constraint multiple-container packing problems. In *Advances in Applied Artificial Intelligence*, pages 1200–1210. Springer.
- Liu, J., Yue, Y., Dong, Z., Maple, C., and Keech, M. (2011). A novel hybrid tabu search approach to container loading. *Computers & Operations Research*, 38(4):797 – 807.
- Liu, Z., Meng, Q., Wang, S., and Sun, Z. (2014). Global intermodal liner shipping network design. *Transportation Research Part E: Logistics and Transportation Review*, 61:28–39.
- Lodi, A., Martello, S., and Vigo, D. (2002). Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410 – 420.
- Loh, T. and Nee, A. (1992). A packing algorithm for hexahedral boxes. In *Proceedings of the Conference of Industrial Automation, Singapore*, volume 115126, pages 115–126.
- Lübbecke, M. E. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6):1007–1023.

- Marin, A. and Salmerón, J. (1996). Tactical design of rail freight networks. part i: Exact and heuristic methods. *European Journal of Operational Research*, 90(1):26–44.
- Martello, S., Pisinger, D., and Vigo, D. (2000). The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267.
- Min, H. (1996). Consolidation terminal location-allocation and consolidated routing problems. *Journal of Business Logistics*, 17(2):235.
- Min, H. and Cooper, M. (1990). A comparative review of analytical studies on freight consolidation and backhauling. *Logistics and Transportation Review*, 26(2):149.
- Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.
- Moura, A. and Oliveira, J. F. (2005). A grasp approach to the container-loading problem. *IEEE Intelligent Systems*, 20(4):50–57.
- Parreño, F., Alvarez-Valdes, R., Tamarit, J. M., and Oliveira, J. F. (2008). A maximal-space algorithm for the container loading problem. *INFORMS Journal on Computing*, 20(3):412–422.
- Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research*, 141(2):382 – 392.
- Powell, W. B. (1986). A local improvement heuristic for the design of less-than-truckload motor carrier networks. *Transportation Science*, 20(4):246–257.
- Powell, W. B. and Sheffi, Y. (1983). The load planning problem of motor carriers: Problem description and a proposed solution approach. *Transportation Research Part A: General*, 17(6):471–480.

- Powell, W. B. and Sheffi, Y. (1989). Design and implementation of an interactive optimization system for network design in the motor carrier industry. *Operations Research*, 37(1):12–29.
- Ratcliff, M. S. W. and Bischoff, E. E. (1998). Allowing for weight considerations in container loading. *Operations-Research-Spektrum*, 20(1):65–71.
- Santos, F. A., da Cunha, A. S., and Mateus, G. R. (2013). Branch-and-price algorithms for the two-echelon capacitated vehicle routing problem. *Optimization Letters*, pages 1–11.
- Sariklis, D. and Powell, S. (2000). A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, pages 564–573.
- Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154.
- Sheffi, Y. (1986). Carrier shipper interactions in the transportation market: An analytical framework. *Journal of Business Logistics*, 7(1).
- Solomon, M. M. (1984). Vehicle routing and scheduling with time window constraints: Models and algorithms. Technical report, No. 84-17364 UMI.
- Tokcaer, S., Özpeynirci, Ö., Demir, M. H., and Çelik, İ. (2016). Shipment consolidation and dispatching problem at Ekol Logistics.
- Vanderbeck, F. (2000). On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128.
- Wang, Z., Li, K. W., and Levy, J. K. (2008). A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. *European Journal of Operational Research*, 191(1):86–99.

Wieberneit, N. (2008). Service network design for freight transportation: a review. *OR Spectrum*, 30(1):77–112.

Wu, Y., Li, W., Goh, M., and de Souza, R. (2010). Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research*, 202(2):347 – 355.

Zhao, X., Bennell, J. A., Bektaş, T., and Dowsland, K. (2016). A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 23(1-2):287–320.

CURRICULUM VITAE

Sinem Tokcaer was born in 1983, Izmir, and completed high school education in Buca Anatolian High School. In 2005, she received her Bachelor's Degree in European Union from University of Bahcesehir, and afterwards, studied Logistics and Maritime transportation in Dokuz Eylul University. In the meanwhile, she worked at logistics service providers in İzmir. In October 2012, she enrolled to Ph.D in Business Administration with a major of Logistics Management at Izmir University of Economics, where she also began to work as research assistant at the department of Logistics Management.