Three-dimensional Capacitated Vehicle Routing Problem

With Loading Constraints

Batoul Mahvash Mohammadi

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfilment of the Requirements
For the Degree of
Doctor of Philosophy (Industrial Engineering) at
Concordia University
Montreal, Quebec, Canada

December 2014

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Batoul Mahvash Mohammadi

Entitled: Three-Dimensional capacitated vehicle routing problems with loading constraints

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Industrial Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________ Chair
Dr. T. Fevens

______________________ External Examiner
Dr. M. Verma

______________________ External to Program
Dr. Y. Zeng

______________________ Examiner
Dr. A. Akgunduz

______________________ Examiner
Dr. M. Chen

______________________ Thesis Co-Supervisor
Dr. A. Awasthi

______________________ Thesis Co-Supervisor
Dr. S. Chauhan

Approved by: ______________________
Dr. A. Dolatabadi, Graduate Program Director

____
December 8, 2014 Dr. A. Asif, Dean
Faculty of Engineering and Computer Science

ABSTRACT

Three-dimensional capacitated vehicle routing problem with loading constraints

Batoul Mahvash Mohammadi,Ph.D.

Concordia University, 2014

City logistics planning involves organizing the movement of goods in urban areas carried out by logistics operators. The loading and routing of goods are critical components of these operations. Efficient utilization of vehicle space and limiting number of empty vehicle movements can strongly impact the nuisances created by goods delivery vehicles in urban areas. We consider an integrated problem of routing and loading known as the three-dimensional loading capacitated vehicle routing problem (3L-CVRP). 3L-CVRP consists of finding feasible routes with the minimum total travel cost while satisfying customers' demands expressed in terms of cuboid and weighted items. Practical constraints related to connectivity, stability, fragility, and LIFO are considered as parts of the problem. We address the problem in two stages. Firstly, we address the three-dimensional (3D) loading problem followed by 3L-CVRP.

The main objective of a 3D loading problem without routing aspect is finding the best way of packing 3D items into vehicles or containers to increase the loading factor with the purpose of minimizing empty vehicle movements. We present the general linear programming model to the pure three-dimensional vehicle loading problem and solve it by CPLEX. To deal with large-sized instances, Column Generation (CG) technique is applied. The designed method in this work outperforms the best existing techniques in the literature.

The 3DVLP with allocation and capacity constraints, called 3DVLP-AC, is also considered. For the 3DVLP-AC, CPLEX could handle moderate-sized instances with up to 40 customers. To deal with large-sized instances, a Tabu Search (TS) heuristic algorithm is developed. There are no solution methods or lower bounds (LBs) for the 3DVLP-AC existent in the literature by which to evaluate the TS results. Therefore, we evaluate our TS with the CPLEX results for small instances.

3L-CVRP is addressed by using CG technique. To generate new columns, the pricing problem that is part of CG is solved by using two approaches: 1-by means of shortest path problem with resource constraints (ESPPRC) and loading problem, and 2-a heuristic pricing method (HP). CG using HP with a simple scheme can attain solutions competitive with the efficient TS algorithms described in the literature.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Prof. Anjali Awasthi and co-advisor Prof. Satyaveer S Chauhan, for the continuous support during my research, for their motivation, enthusiasm, and knowledge.

I would like to express my gratitude to the examining committees for the time spent in reading this document. I benefited a lot from their constructive comments, questions, and suggestions.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter1:
Introduction

1.1 Background

The negative environmental impacts of urban freight, such as pollution and high levels of traffic congestion, are steadily increasing in cities around the world. Freight movements are essential to economic and social activities because they are a major source of employment and link producers to customers in the supply chain. However, they are also the source of a great proportion of certain pollutants as compared with interurban freight, due to higher fuel consumption per unit of distance travelled. More and more cities are experiencing high numbers of freight movements. Accordingly, it has become an urgent challenge to manage urban freight mobility in an efficient manner in terms of economic, social, and environmental costs.

It should be also noted in particular the City Logistics concept has been developed through considering the effects of freight transportation within cities. Benjelloun et al. [1] define City Logistics concept as a new process for managing freight transportation and alleviating harmful effects of increasing freight vehicles, while not penalizing essential factors of freight movements to most economic and social activities. Taniguchi et al. [2] define City Logistics as “the process for totally optimising the logistics and transport activities by private companies in urban areas while considering the traffic environment, the traffic congestion and energy consumption within the framework of a market economy."

Aforementioned, it has become an urgent challenge to manage urban freight mobility in an efficient manner for a number of environmental, economic and social reasons summarized as follows:

- The contribution that an efficient urban freight transport system makes to the competitiveness of industry in an urban area;
- Existing well-organized transport systems prevent economic impacts like efficiency and resource waste;
- Costs of commodities are deeply affected by the freight transport costs and also total cost of freight transports has a strong influence on the efficiency of the economy;
- An efficient improvement in living conditions in cities is resulted by reducing empty vehicle operating within cities and controlling number and dimensions of freight vehicles; and
- Reducing environmental impacts such as pollution, noise, high levels of traffic congestion and high energy consumption raised by the urban freight movements.

An efficient mobility for freights can be resulted by developing and implementing policies and strategies in City Logistics. One way to achieve efficient mobility for freight systems is by developing and implementing well-organized policies and strategies. Governments and companies can both engage in changing urban freight systems by means of defining and introducing efficient measures (Crainic et al. [3], Anderson et al. [4]). Governments can introduce policies and measures that can encourage companies to adopt sustainable practices in operating their freight systems. City access control, parking regulations, road pricing, time windows, and vehicle restrictions are some instances of such policies. There are also operational measures that can be taken by city logistics operators, such as planning collaboratively for goods distribution or developing new goods distribution strategies to conform to municipal regulations. Under the influence of municipal regulations, companies may have to devise new goods distribution planning strategies, such as reorganizing freight transport operations, changing supply chain organization, using more efficient technologies, employing more efficient methods for loading and routing goods, improving vehicles' fuel efficiency, and consolidating shipments.

The three-dimensional loading and routing problems are increasingly important issues in supply chain and distribution systems, with strong implications for freight transport costs. Clearly, the costs of commodities are deeply affected by freight transport costs, and the total cost of freight transports has a powerful impact on the efficiency of the economy.

1.2 Objective

The purpose of this dissertation is as follows:

- Addressing the general mixed-integer linear programming model to the three-dimensional vehicle loading problem (3DVLP) that deals with the way of loading cuboid items in one or more vehicles such that empty vehicle space is minimized in purpose of minimizing vehicle movements;
- Developing an efficient heuristic method for 3DVLP; and

- Developing an efficient heuristic method for an integrated problem of vehicle routing and 3DVLP (called 3L-CVRP) with additional constraints such as stability, fragility, and LIFO. The main objectives involved in this integrated problem are partitioning customers' demands into groups based on loading constraints, and determining the sequence in which customers are visited in order to minimize total cost.

1.3 Thesis Organization

The thesis is divided into 6 chapters. The first chapter gives a general introduction to the topic and identifies the objectives of the work. Chapter 2 reviews the literature on loading problems, practical constraints in loading, exact and heuristic methods to solve the loading problems, and the integrated problem of loading and routing. In Chapter 3 we present in detail the three-dimensional vehicle loading problem and formulate it as a mixed integer programming model. The integrated problem of routing and loading is also covered in this chapter. Chapter 4 describes our proposed approaches to solve both the loading problem and the integrated problem of routing and loading. Numerical applications and results are presented and discussed in Chapter 5. Finally, conclusions and suggestions for further investigation are included in Chapter 6.

Chapter 2:
Literature Review

In this chapter, we present a detailed review of the literature on integrated vehicle loading and routing problems using three broad categories: loading problems, loading problems with practicality constraints, and capacitated vehicle routing problems with three-dimensional loading constraints.

2.1 Loading Problems

The loading problem involves loading a given set of items into bins or compartments with the objective of minimizing empty space of bins. The loading problem is referred to as cutting and packing problem in the literature (Bortfeldt and Wäscher [5]). In the literature there is a range of names for the packing problem, depending on the assumptions used to address it: cutting stock or trim loss problem; bin or strip packing problem; vehicle, pallet or container loading problem; nesting problem; knapsack problem; and so on. Dyckhoff [6] clarifies logical structure and characteristics that need to be considered in the various kinds of packing problems. The significance characteristics are as follows:

- Dimensionality;
- Kind of assignment;

- Assortment of large objects; and
- Assortment of small items.

The ‘dimensionality’ characteristic specifies whether items are one-dimensional, two-dimensional, three-dimensional or even four-dimensional. An instance of a four-dimensional case is one in which cuboid items must be stored in a container for a fixed time period. The ‘kind of assignment’ characteristic determines whether all items and a selection of objects, or a selection of items and all objects, are to be investigated. The ‘assortment of large objects’ characteristic considers the properties of objects (containers or vehicles), that is, whether they have identical or different figures. Finally, the ‘assortment of small items’ characteristic determines the category of items, that is, if there are a few items of various figures or several items of different figures, etc. Detailed information on each characteristic can be found in Dyckhoff [6]. Wäscher et al. [7] presented an improved typology based on Dyckhoff’s original ideas. They introduce new categorization criteria different from those of Dyckhoff and suggest new system names.

Pisinger [8] classifies loading problems into three types, namely bin packing, strip packing, and knapsack problems. For each of these types, a detailed explanation of the objective function and restrictions is presented in the following sections.

2.1.1 Bin Packing Problem

According to the typology suggested by Wäscher et al. [7], bin packing problems can be further classified into three types: single bin packing problem (SBPP), multiple bin packing problem (MBPP), and residual bin packing problem (RBPP).

In SBPP a set of different items must be assigned to a set of identical bins, in MBPP to a set of weakly heterogeneous bins, and in RBPP to a set of strongly heterogeneous bins, while in all cases minimizing the number of bins used. Each of the problems can be considered as one-dimensional, two dimensional or three-dimensional based on the properties of items and bins. Since we deal with SBPP in this work, a detailed literature on different types of this problem is presented in the following.

One-Dimensional Bin Packing Problem (1BPP)

This is the classic bin packing problem, in which a set of different items of given weight is to be packed into a minimum number of bins of identical capacity. The total weight of the packed items in a bin should not exceed the bin's capacity. The reader is referred to Scholl et al. [9], Schwerin and Wäscher [10] and Martello and Toth [11] for more details.

Best fit decreasing (BFD) and first fit decreasing (FFD) are well-known heuristic methods for solving this problem (Gilmore and Gomory [12]). The set of selected bins is initially empty in both methods. Then, items are sorted in non-increasing order of weight. FFD assigns each item to the first bin that can receive it, while BFD assigns each item to the bin with the minimum residual capacity. If all selected bins have insufficient residual capacity to accommodate an item, a new bin is selected.

Two-Dimensional Bin Packing Problem (2BPP)

Given an unlimited number of large identical rectangular bins, 2BPP is concerned with packing a given set of distinct rectangular items into the minimum number of bins without overlapping. An item is specified by a height (or length) and a width. 2BPP has applications in the cutting of rectangular glass or wood pieces from a large piece of material.

Gilmore and Gomory [13] were the first to model 2BPP by extending their 1BPP approach to it. Fekete and Schepers [14] propose different modeling approaches using a graph-theoretical characterization. A number of exact algorithms and lower bounds for 2BPP can be found in Pisinger and Sigurd [15], Martello and Vigo [16], Boschetti [17], and Caprara and Monaci [18]. Exact approaches for solving 2BPP usually employ a branch-and-bound technique (B&B).

Greedy heuristics for 2BPP commonly employ the concept of levels. The first level is the bottom of the bin on which the items' bases are located. The next level is the horizontal line drawn parallel to the top of the highest item located on the previous level, and so on. In the majority of greedy algorithms for 2BPP, given a list of items sorted by non-increasing order of their height, items are iteratively placed in levels in accord to the following rules (Coffman et al. [19]):

- **Next fit decreasing height (NFDH)**: Each item is packed in the current fitting level starting from the left. If the item cannot be packed on the current level then the level is closed and a new current level is created;

- **First fit decreasing height (FFDH)**: Each item is packed on the first existing fitting level in the bin starting from the bottom left. If there is no level in which the item fits, a new level is created;
- **Best fit decreasing height (BFDH)**: Each item is packed on the fitting level with the minimum residual horizontal space. If there is no fitting level available, a new one is created; and
- **Floor–ceiling:** Items are packed from left to right with their bottom edges on the level floor, and also from right to left with their top edges touching the level ceiling, i.e., the horizontal line drawn on the top of the tallest item packed on the floor (Lodi et al.[20]).

Puchinger and Raidl [21] describe a three-stage two-dimensional bin packing problem to deal with real-world applications such as glass, paper, or steel cutting. Lodi et al. [20] propose a tabu search algorithm to solve 2BPP, having tested it on 500 instances generated randomly with a maximum of 100 items. These authors provide a survey of the mathematical models, lower bounds, and greedy methods relevant to 2BPP and discuss recent heuristic and metaheuristic methods and exact approaches.

Three-Dimensional Bin Packing Problem (3BPP)

This problem consists of packing a set of distinct rectangular items (boxes) orthogonally into a minimum number of large identical rectangular containers (or vehicles) without overlapping. An item is specified by a length, a width and a height.

Chen et al. [22] describe an integer programming formulation for 3BPP with weight distribution and orientation constraints. The numerical experiments are limited to a single small-sized instance with six boxes solved to optimality through the LINGO solver.

Materllo et al. [23] were the first to design an exact two-level B&B algorithm for 3BPP. In this algorithm, boxes are assigned to bins in the first level of the search. A second level B&B method is employed in each node of the first level search tree to verify the feasibility of packing items using the concept of corner points (CPs). CPs indicate the points where a new item can be accommodated within the residual space of the container in a given partial packing. The authors also propose two heuristic algorithms, H1 and H2, which are executed at each root node of the search tree. Algorithm H1 builds a number of layers based on the depth of the bins and combines the layers into bins by solving a one-dimensional bin packing problem defined on the depths of the layers. Algorithm H2 minimizes the bins empty space by heuristically filling each one as full as possible. B&B algorithm was tested on instances with up to 60 boxes. Numerical results show that instances with up to 20 boxes can be solved to optimality within a reasonable time limit.

Den Boef et al. [24] indicate that the exact algorithm proposed by Martello et al. [23] might fail to give optimal solutions to 3BPP, since it cannot generate all feasible orthogonal packing patterns. Therefore, Martello et al. [25] have extended their approach in [23] by combining the original enumerative method with a new constraint-based programming approach. Using this algorithm, moderate-sized instances can be solved to optimality for the general 3BPP as well as for robot-packable variants of the problem.

Faroe et al. [26] propose a guided local search (GLS) heuristic method for 3BBP. They first obtain an upper bound on the number of bins through a greedy heuristic method. GLS iteratively tries to decrease this upper number by searching for a feasible packing of the boxes, and stops when a given time limit is reached or the current solution is equal to the given lower bound. GLS has been tested on instances of up 200 boxes. The authors indicate that GLS gives solutions equal to or better than the exact algorithm suggested by Martello et al. [23].

Lodi et al. [27] propose a heuristic method based on layers, called height first area second (HA) . The base of a bin is defined as the first layer; the height of the tallest item available in the first layer is the second layer, and so on. HA selects the best solutions through the following two methods:

1. Items are partitioned into clusters according to non-increasing order of their height. Then the layers resulting from each cluster are packed into bins through a 1BPP solution; and
2. Items are sorted by non-increasing order of their base area and re-allocated to the current layers. Then the layers are packed into bins via a 1BPP solution.

Authors also describe a tabu search method in which the neighborhood is evaluated using HA. The combination of TS and HA gives satisfactory solutions when compared with the exact and heuristic methods suggested by Martello et al. [23]. The average solution of TS is exactly the same as GLS [26] in one third of the instances. In some cases, TS is better that GLS, and vice versa. As the authors point out, the TS and GLS methods complement each other.

To better utilize the bin's volume, a new concept called extreme points (EPs), an extension of CPs, was proposed by Crainic et al. [28]. Using EPs, the authors present a new heuristic algorithm based on the first fit decreasing and the best fit decreasing. Crainic et al. [29] also propose a two-level tabu search called TS^{2}PACK for 3BPP. In TS^{2}PACK, the number of bins is optimized in the first level, and the accommodation of items within bins is optimized in the second. An extension of the interval graph representation of packing (see Fekete and Schepers [30]) is used within the algorithm. The *k*-chain-moves procedure is also applied to maximize the neighborhood and to improve the quality of the solutions. Computational results on benchmark problem instances show that the proposed approach gives better results than other methods mentioned before.

A greedy randomized adaptive search procedure (GRASP) for 3BPP is proposed by Parreño et al. [31]. The GRASP algorithm iteratively combines a constructive procedure and an improvement phase in order to obtain a solution. In the improvement phase, various moves in a variable neighborhood descent (VND) structure are used to improve the solution obtained by the constructive procedure. GRASP/VND algorithm gives solutions equal to or better than Crainic et al. [29]. Table 2-1 gives an overview of the significant methods in the literature for 3BPP.

Table 2-1: An overview of all efficient methods for 3BPP

Methodology / Work	Exact Algorithm	Heuristic Method
Martello et al. [23], [25]	Branch-and-Bound(B&B)	H1, H2
Faroe et al. [26]		Guided Local Search (GLS)
Lodi et al. [27]		HA, Tabu Search(TS)
Crainic et al. [28]		Extreme point best fit decreasing method (C-EPBFD)
Crainic et al. [29]		Tabu Search (TS^{2}PACK)
Parreño et al. [31]		GRASP/VND
Mahvash et al. [32] (our work)	Column Generation(CG)	

2.1.2 Strip Packing Problem

According to the typology of Wäscher et al. [7], the strip packing problem can be considered as an open dimension problem (ODP) in which a set of items is to be packed into a single container with one or more variable dimensions. There are two variants of this problem, the two-dimensional and the three-dimensional strip packing problems (2SPP and 3SPP respectively). 2SPP consists of packing a given set of different rectangles into one open-ended rectangular container of given width and hypothetically infinite height (i.e., a strip) in such a way as to minimize the overall height. All rectangles should be packed into the container without overlapping. This problem has industrial applications such as paper and cloth cutting. 3SPP is concerned with packing cuboid items into a container with infinite height (strip) and fixed width and length.

Exact methods for solving 2SPP are based on a B&B approach (Martello et al. [33]), while heuristic methods are based on bottom-left (BL) (Baker et al. [34]), and best-fit (BF) (Burke et al. [35]). In BL heuristics, rectangular items are sorted based on their areas and placed in a container as left as possible. In the heuristics based on BF, free places from the bottom-most and left-most are considered first, and the best-fitting rectangular is selected for each place. Metaheuristic approaches for the 2SPP are TS, Simulated Annealing (Dowsland [36]) and Genetic algorithms (Jakobs [37]). The reader is referred to Riff et al. [38] for a survey of 2SPP and to Bortfeldt and Mack [39] ,Wu et al. [40], and Allen et al. [41] for the heuristic algorithms for 3SPP.

2.1.3 Knapsack Packing Problem

In the knapsack problem, there is a set of different items, each associated with a profit value. The objective in this problem is to pack a subset of items with maximal total profit into a single container. If the profit of an item is expressed in terms of volume, then the problem objective is to minimize the container's empty space. Egeblad and Pisinger [42] propose an exact method and heuristics for the two- and three-dimensional types of the problem. Gehring and Bortfeldt [43] present a genetic algorithm for loading strongly heterogeneous sets of cartons into a single container. We refer the reader to Moura and Oliveira [44], Parreño et al. [45], Fanslau and Bortfeldt [46], and Gonçalves and Resende [47] for more details on methods for addressing the knapsack packing problem.

2.2 Practical Constraints in the Loading Problem

There are a number of practical constraints involved in the loading problem under real-world circumstances. For instance, goods may need to be packed based on the sequence of their delivery to the customers; on this basis it is impossible to pack the first parcel for delivery at the bottom of the vehicle. Bortfeld and Wäscher [5] classified all the significant practical constraints identified in the literature that are commonly encountered in the real-world container loading problem. We briefly outline them in the following sections.

Container Weight Capacity Constraint

As the name implies, this constraint requires that the total weight of items loaded into a container should not exceed its maximum designed weight capacity. This constraint is normally invoked in cases when items are too heavy.

Balancing Constraints

The weight of packed items in a loading pattern should be spread across the floor of the container as evenly as possible in order to satisfy balancing constraints (see Mathur [48]). Balancing constraints reduces the risk of items shifting when the container is moved. These constraints demand that the center of gravity of a loading pattern be an optimal point close to the geometrical mid-point of the container, and no more than a specified distance away from it. Balancing constraints arise in several practical applications such as aircraft and space cargo loading.

Orientation Constraints

In general, a box can be loaded into a container with any one of six orientations. A vertical orientation of a box is defined by choosing a particular dimension as the height. Vertical orientation constraints allow only orthogonal loading patterns, and thus prevent damage to the goods while guaranteeing the stability of the loading pattern (Bortfeldt and Wäscher [5]).
Orientation constraints may allow just a single orientation for a box, which means that the box cannot be rotated at all. For some purposes, upside-down rotations are not allowed, and boxes can be rotated only by 90 degrees on the horizontal plane. In this case, each box is loaded with a designated surface on the bottom. Conversely, there are applications in which no restrictions apply to the orientation of the boxes, in which case boxes can be rotated both vertically and horizontality (Lodi et al. [49]).

Load-Bearing Constraints

Load-bearing or stacking constraints are restrictions to the placement of items on top of each other. The maximum weight placed on a box and the maximum number of boxes stacked on the top of each other can be limited by such constraints. The fragility constraint, as a subset of this category, prevents non-fragile items being placed on top of fragile items. These constraints avoid damaging boxes and protect the contents of the load.

Allocation Constraints

In some applications, items are partitioned into subsets based on their properties. A subset can be the set of items designated for the same customer or destination. Allocation or connectivity constraints require that items of a particular subset should be packed into the same container (see Eley [50]). Allocation constraints may also be applied as separation constraints, preventing different classes of items from being packed with each other in the same container. For example, food and perfumery items should not be packed together in a single container.

Cargo Positioning Constraints

Positioning constraints restrict the location of items relative to each other within the container. In the last-in-first-out (LIFO), sequential loading or rear loading policies, items should be packed into a container according to their delivery sequence. For example, the first item for delivery cannot be packed on the bottom of the container. The goal is that items should be unloaded easily and without moving other items (Gendreau et al. [51]).

Stability Constraints

Unstable loads may result in damage to items and unsafe conditions in the loading and unloading operations. Vertical stability constraints reduce the risk of items falling down onto the container floor or onto the tops of other items. Such constraints demand that the bottom faces of items be supported by either the top surfaces of other boxes or by the container floor (Junqueira et al. [52]).

2.3 Capacitated Vehicle Routing Problem with Loading Constraints

The Capacitated Vehicle Routing Problem (CVRP) was first introduced by Dantzig and Ramser [53] in 1959. Given a fleet of vehicles located in the central depot and customers with specific demands, CVRP consists of finding the set of vehicle routes with the minimum total cost. In this problem, each customer should be visited by exactly one vehicle and each vehicle should perform at most one route. The total cost is usually defined as vehicle operating costs plus additional costs imposed along the routes. The total demand of customers to be served by a given vehicle should not exceed that vehicle's capacity.

Toth and Vigo [54] describe four different variants of CVRP, as follows:

- In distance-constrained CVRP, traveling time between each pair of customers is considered, and the total traveling time of each vehicle is limited by an upper bound;
- In CVRP with time windows, each customer's service is to take place within a given time interval or window. If a vehicle arrives before the beginning of the time window, then it must wait until the window begins before starting its service;
- CVRP with backhauls posits two types of customers, namely line-haul and backhaul customers. The line-haul customer receives a given quantity of products from the depot, while the vehicle picks up a given quantity of products from the backhaul customer. This problem can be divided into four sub-groups (see Parragh et al. [55]); and

- In CVRP with pickup and delivery, for each customer there exist both products to be delivered and products to be picked up. Moreover, in this problem goods are transported between customers. For each customer, the delivery is implemented before the pickup. This problem can be divided into two sub-groups (see Parragh et al. [55]).

To decrease operational costs in the supply chain, it is needed to solve CVRP using effective approaches. Toth and Vigo [54] identify the following as the most widely-used exact algorithms for solving CVRP and its variants: branch-and-bound, branch-and-cut [56], branch-and-price [57], branch-and-cut-and-price [58], and set-covering-based.

Some heuristic methods applied to CVRP are the nearest neighbor algorithm, insertion algorithms, and tour improvement procedures. The structures of most of the heuristics developed for VRP are detailed in Laporte et al. [59] and Cordeau et al. [60]. Metaheuristics successfully applied to VRP include TS, simulated annealing (SA), genetic algorithm (GA), memetic algorithm (MA), ant colony optimization (ACO), and variable neighborhood search (VNS). The reader is referred to Füllerer [61] for more details.

Excellent results for CVRP have recently been provided by Fukasawa et al. [62] (branch-and-cut-and-price), Baldacci and Mingozzi [63], and Baldacci et al. [64] (set partitioning formulation). Baldacci et al. [64] have developed a new exact algorithm for CVRP based on set partitioning formulation, with additional cuts corresponding to capacity and clique inequalities. Fukasawa et al. [62] indicate that the best exact algorithms for CVRP are based on either branch-and-cut or lagrangian relaxation/column generation. They have developed an algorithm combining these two approaches which can solve to optimality all instances from the literature with up to 135 vertices. We refer interested reader to books by Toth and Vigo [54] and Golden et al. [65], as well as to the recent surveys by Laporte [66] and Baldacci et al. [67].

Most vehicle routing problems consider only those capacity constraints which require that the total weight of products transported by a vehicle should not exceed the vehicle's capacity. In real-world distribution systems, the shape of the products is also a key factor. Therefore, vehicle routing problems must also take into account loading constraints that identify feasible placements of products within vehicles based on their shapes and properties.

In the literature, vehicle routing problems with loading constraints are classified into two types, two-dimensional and three-dimensional loading capacitated vehicle routing problems (2L-CVRP and 3L-CVRP respectively).

2L-CVRP

In 2L-CVRP, the customer's demands are expressed in terms of distinct rectangular items. Items cannot be stacked on top of each other due to their fragility, weight, or large dimensions. One of the applications of this problem is the distribution of kitchen appliances such as refrigerators.

Iori et al. [68] were the first to propose an exact approach to solve 2L-CVRP. In their approach, routing costs are minimized by a branch-and-cut algorithm, and loading aspects are iteratively checked by a branch-and-bound algorithm. The authors tested their approach on benchmark instances derived from the classical CVRP test problems involving up to 35 customers and more than 100 items. Gendreau et al. [51] solve larger 2L-CVRP instances with up to 255 customers and 786 items by employing a TS method, where customers are relocated through a generalized insertion procedure (GENI) [69]. A two-dimensional strip packing problem is also applied in order to guarantee the optimal placement of the load components. Interested reader may refer to Fuellerer et al. [70], Zachariadis et al. [71] and Duhamel et al. [72] for other methods of solving 2L-CVRP.

3L-CVRP

In 3L-CVRP, the demands of customers are expressed in terms of cuboid and weighted items. The aim of the problem is to find feasible routes with the minimum total travel cost while satisfying customers' demands and practical loading constraints. 3L-CVRP, as one of the rich routing problems, draws a great deal of attention in supply chain management. In particular, this problem has significance for applications that deal with many large items and in which the loading requirement is not trivial. Some examples include distribution of household appliances, , and mechanical components.

There is no exact algorithm for 3L-CVRP in the literature. Most researchers simply extend metaheuristic algorithms from 2L-CVRP to apply to 3L-CVRP. As mentioned by Vidal et al. [73], most existing techniques for 3L-CVRP are based on TS combined with efficient packing heuristics. Gendreau et al. [51] were the first to employ TS to tackle routing aspects of the problem and applied two packing heuristics to handle loading constraints. They evaluated their algorithm based on vehicle routing instances adapted from the literature as well as on new real-world instances. Tarantilis et al. [74] describe a hybrid metaheuristic methodology called GTS that combines the approaches of TS and guided local search (GLS). They show that GLS improves the solution attained by TS within a variable neighborhood search. The loading characteristics are determined by employing a collection of packing heuristics. GTS improves average solution of TS of Gendreau et al. [51] by 3.54%.

Zhu et al. [75] apply a two-phase tabu search algorithm. They obtain feasible solutions in the first phase and then optimize the traveling cost in the second phase. The loading aspects are handled by two sequence-based loading heuristic methods called deepest-bottom-left-fill (DBLF) and maximum touching area (MTA). To explore new solutions in the TS, the authors employ five neighborhood operators. Wisniewski et al. [76] also solve the routing aspect of 3L-CVRP by means of a TS and a first-improvement local search. To deal with loading constraints, they employ a randomized bottom-left-based algorithm using multiple permutations of the bottom-left heuristic. Bortfeldt [77], Tao and Wang [78], and Wang et al. [79] also solve 3L-CVRP by means of a TS.

The following works employ other heuristic methods to deal with 3L-CVRP.

Fuellerer et al. [80] solve 3L-CVRP by means of a highly efficient ant colony optimization (ACO) algorithm, adapted of the savings-based ants [81]. They handle routing aspects by an ant-based procedure. To deal with loading components, the ACO employs and iteratively invokes the local search and packing heuristics used in Gendreau et al. [51]. The ACO outperforms TS of Gendreau et al. [51] in 26 out of 27 instances and GTS of Tarantilis et al. [74] in 23 out of 27 instances.

Ruan et al. [82] describe a metaheuristic called HA that integrates a honey bees mating optimization (HBMO) [83] with six types of loading heuristics [74]. The proposed method improves the average solution of TS of Gendreau et al. [51] by 7% and the average solution of ACO of Fuellerer et al. [80] by 0.62%.

Lacomme et al. [84] consider 3L-CVRP without fragility, stability or LIFO constraints. They solve the routing component of the problem by a GRASP×ELS [85] hybrid algorithm and the loading part by a two-step procedure. In the loading part, (*x*, *y*) positioning items are first obtained by relaxing z-constraints. Then a feasible loading is found by computing the compatible z-coordinate. Note that GRASP×ELS is a combination of the greedy randomized adaptive search procedure (GRASP) and the evolutionary local search (ELS). GRASP [86] is a multi-start local search metaheuristic and ELS an extension of the iterated local search. GRASP×ELS outperforms the TS in Gendreau et al. [51], ACO in Fuellerer et al. [70], and TS in Bortfeldt [77]. We refer the reader to Wang et al. [87] and Iori and Martello [88] for a survey of 3L-CVRP.

Research gaps in 3L-CVRP

Table 2-2 provides an overview of all methods proposed in the literature for 3L-CVRP. The average costs and average times for each method are reported when 3L-CVRP deals with all loading constraints (including 3D, fragility, stability and LIFO) and with 3D loading constraints alone. A comparison of average costs reveals that the TS method of Wisniewski et al. [76] outperforms other works when all constraints are considered. For the 3D loading case alone, the method of Zhu et al. [75] gives the best average cost. Bold values in the table indicate the best average costs for all loading constraints and for 3D loading constraints alone. The symbol "-" in the table indicates that the relevant work contains no report for this entry.

Table 2-2: An overview of solution methods for 3L-CVRP

Work	Constraints	Routing part	Loading part	All Constraints		3D-Loading	
				AvgCost	**AvgTime**	**AvgCost**	**AvgTime**
Gendreau et al. [51]	LIFO Stability Fragility	Tabu search	Tabu search+ direct packing	1042.26	2058.9	876.31	1567.4
Tarantilis et al. [74]	LIFO Stability Fragility	Tabu search +Guided local search (GLS)	Heuristic direct packing	997.18	1471	876.39	-
Fuellerer et al. [80]	LIFO Stability Fragility	Ant colony optimization (ACO)	Heuristic direct packing	966.66	1746.51	856.67	689.3
Tao and Wang [78]	LIFO Stability Fragility	Tabu Search	Heuristic direct packing	953.09	945	848.27	468.3
Wang et al. [79]	LIFO Stability Fragility	Tabu Search	Heuristic direct packing	956.10	820	-	-
Zhu et al. [75]	LIFO Stability Fragility	Tabu Search	Heuristic direct packing	962.08	1419	**846.44**	-
Bortfeldt [77]	LIFO Stability Fragility	Tabu search	Tree search +direct packing	958.74	219.40	864.53	810.30
Wisniewski et al. [76]	LIFO Stability Fragility	Tabu search + a first-improvement local search	Heuristic direct packing	**948.74**	930.40	-	-
Ruan et al. [82]	LIFO Stability Fragility	Honey Bee Mating Optimization	Heuristic Direct packing	960.10	754.21	858.07	-
Lacomme et al. [84]	Just 3DLoading	GRASP_ELS	2-step procedure based on a relaxation of the 3DPP	-	-	847.04	1004.89

Table 2-3 gives the percentage differences in the average solution cost for each method compared to the best methods for all constraints and for 3D loading alone. These gaps have been evaluated as (*average cost–best average cost*)/*best average cost*. Aforementioned, TS method of Wisniewski et al. [76] is best method when all constraints are considered. For the 3D loading case alone, the method of Zhu et al. [75] give the best average cost.

Table 2-3: Percentage gaps between each method and best method for 3L-CVRP

Work	All constraints	3D loading
	Gap %	*Gap%*
Gendreau et al. [51]	9.85	3.52
Tarantilis et al. [74]	5.10	3.53
Fuellerer et al. [80]	1.88	1.20
Tao and Wang [78]	0.45	0.21
Wang et al. [79]	0.77	-
Zhu et al. [75]	1.40	**0.0**
Bortfeldt [77]	1.05	2.13
Wisniewski et al. [76]	**0.0**	-
Ruan et al. [82]	1.19	1.37
Lacomme et al. [84]	-	0.07

Chapter 3:
Problem Statement

3.1 Assumptions

The assumptions associated with the customers, the items, and the vehicles in our three-dimensional vehicle loading problem and the vehicle routing problem with loading constraints are:

- **Customer:** A customer has a deterministic demand expressed in terms of a set of weighted and cuboid items. In general, the capacity required for a single customer does not exceed the capacity of vehicle in terms of volume and weight. For the routing case, the location of the customers on the road network and the cost of traveling between each pair of customers are given;
- **Item:** An item is a cuboid box whose length, width and height are given. An item may be defined as fragile or non-fragile according to the type of goods packed in it. Each item is considered as an independent unit in the problems described below; and
- **Vehicle:** The type of a vehicle is specified according to its capacity, expressed as the maximum weight or volume of the items that may be loaded into the vehicle. The length, width, and height of the vehicle are known. A vehicle has a rectangular loading surface accessed only from one side.

3.2 Three-Dimensional Vehicle Loading Problem (3DVLP)

The 3DVLP consists of packing items orthogonally into a minimum number of vehicles. Each item has a fixed orientation; that is, it cannot be rotated around any of the axes. Each vehicle has only a volume capacity.

The modeling parameters and variables used in the mathematical formulation of 3VDLP are:

I,	The set of all items;
$(l_i, w_i, h_i\ ),$	Parameters representing the length, width and height of item $i \epsilon I$;
$(L, W, H),$	Length, width and height of vehicle; and
$(x_i, y_i, z_i),$	Continuous variables representing the position of item *i* within the vehicle, that are, coordinates of back-left bottom (BLB) corner of item in the vehicle.

Without loss of generality, it is assumed that the length of vehicle falls along the *X*-axis and its width along the *Y*-axis. The BLB corner of the vehicle is fixed at the origin. Figure 3-1 gives a geometric illustration of a vehicle with item *i* packed into the vehicle.

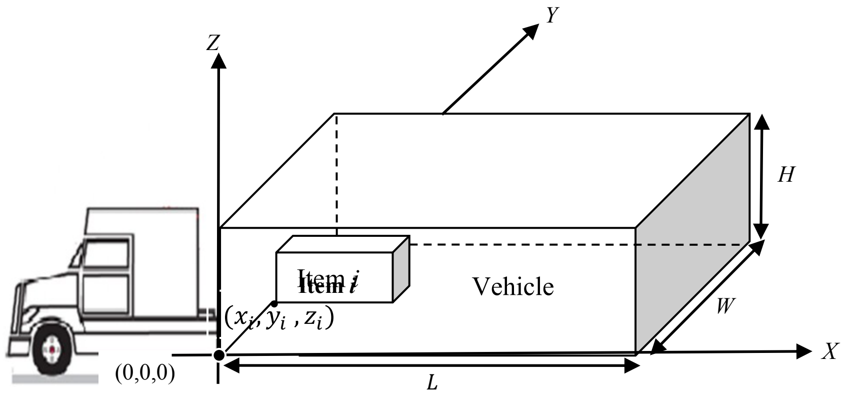


Figure 3-1: Geometric presentation of a vehicle with item i packed in it

In order to indicate the position of two items i and i' relative to each other, we first define the binary variables $f_{ii'}^1, f_{ii'}^2, f_{ii'}^3, f_{ii'}^4, f_{ii'}^5$ and $f_{ii'}^6$. The value of $f_{ii'}^1$ is equal to one if item i is behind item i' and zero otherwise. The value of $f_{ii'}^2$ is equal to one if item i is at the front of item i' and zero otherwise. The value of $f_{ii'}^3$ is equal to one if item i is in the left side of item i' and zero otherwise. The value of $f_{ii'}^4$ is equal to one if item i is in the right side of item i' and zero otherwise. The value of $f_{ii'}^5$ is equal to one if item i is below item i' and zero otherwise. The value of $f_{ii'}^6$ is equal to 1 if item i is above item i' and zero otherwise.

Let us define nV as total number of vehicles needed to pack all items. Suppose that n_i is the vehicle's number containing item i. We define binary variable $s_{ii'}$ equal to one if and only if $n_i = n_{i'}$ and zero otherwise. M is an arbitrary large number.

A partial linear mixed integer programming model for 3DVLP that is an extension of Chen et al. [22] is:

$$Min\ nV \tag{3.1}$$

Subject to:

$$x_i + l_i \le L \qquad \forall\, i \in I \tag{3.2}$$

$$y_i + w_i \le W \qquad \forall\, i \in I \tag{3.3}$$

$$z_i + h_i \le H \qquad \forall\, i \in I \tag{3.4}$$

$$x_i + l_i \le x_{i'} + \left(1 - f_{ii'}^1\right)M \qquad \forall\, i, i' \in I, i < i' \tag{3.5}$$

$$x_{i'} + l_{i'} \le x_i + (1 - f_{ii'}^2)M \qquad \forall\, i, i' \in I, i < i' \tag{3.6}$$

$$y_i + w_i \le y_{i'} + \left(1 - f_{ii'}^3\right)M \qquad \forall\, i, i' \in I, i < i' \tag{3.7}$$

$$y_{i'} + w_{i'} \le y_i + (1 - f_{ii'}^4)M \qquad \forall\, i, i' \in I, i < i' \tag{3.8}$$

$$z_i + h_i \le z_{i'} + \left(1 - f_{ii'}^5\right)M \qquad \forall\, i, i' \in I, i < i' \tag{3.9}$$

$$z_{i'} + h_{i'} \le z_i + (1 - f_{ii'}^6)M \qquad \forall\, i, i' \in I, i < i' \tag{3.10}$$

$$f_{ii'}^1 + f_{ii'}^2 + f_{ii'}^3 + f_{ii'}^4 + f_{ii'}^5 + f_{ii'}^6 \ge s_{ii'} \qquad \forall\, i, i' \in I, i < i' \tag{3.11}$$

$$1 \le n_i \le nV \qquad \forall\, i \in I \tag{3.12}$$

$$x_i, y_i, z_i \ge 0 \tag{3.13}$$

$$f_{ii'}^1, f_{ii'}^2, f_{ii'}^3, f_{ii'}^4, f_{ii'}^5, f_{ii'}^6, s_{ii'} \in \{0,1\} \qquad \forall\, i, i' \in I \tag{3.14}$$

$$x_i, y_i, z_i\,, n_i, nV \in \mathbb{Z} \qquad \forall\, i\ \in I \tag{3.15}$$

Objective function (3.1) gives the minimum number of vehicles needed to pack all items. Constraints (3.2) to (3.4) ensure that an item that is orthogonally packed into a vehicle lies within the vehicle's geometric boundaries. According to constraints (3.5) to (3.10), items cannot overlap with each other. Constraint (3.11) demands that an overlapping condition should be checked for only those items packed in the same vehicle.

Model contains $8|I|+\frac{7}{2}|I|(|I|$-1) constraints and $4|I|+1+\frac{7}{2}|I|(|I|$-1) variables. It should be noted that all variables are integer variables in the model.

3DVLP with Different Item's Orientations

If an item can be rotated in all directions then it may be located into the vehicle in six orientations as described below (see Figure 3-2):

- Orientation-1: Item's length is parallel to the *X*-axis and its width is parallel to the *Y*-axis;
- Orientation-2: Item's length is parallel to the *Y*-axis and its width is parallel to the *X*-axis;
- Orientation-3: Item's length is parallel to the *Z*-axis and its width is parallel to the *Y*-axis;
- Orientation-4: Item's length is parallel to the *Z*-axis and its width is parallel to the *X*-axis;
- Orientation-5: Item's length is parallel to the *X*-axis and its width is parallel to the *Z*-axis; and
- Orientation-6: Item's length is parallel to the *Y*-axis and its width is parallel to the *Z*-axis.

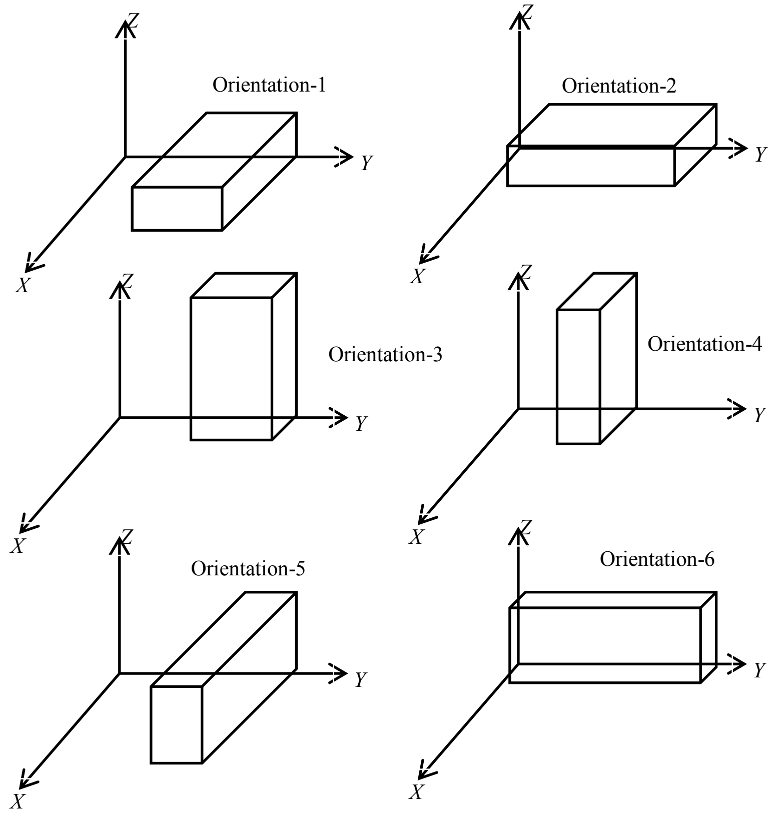


Figure 3-2: Item's Orientations

For each item $i \in I$, we define binary variables $o_{ik}(k = 1, \dots, 6)$ that indicate different orientations of item i in a mathematical formulation of 3DVLP. The value of o_{ik} is equal to one if item i is placed into a vehicle in accordance with the orientation-k and zero otherwise. Considering orientations variables, constraints 3-2 to 3-10 in the formulation of 3DVLP are replaced by the following constraints:

$$x_i + l_i(o_{i1} + o_{i5}) + w_i(o_{i2} + o_{i4}) + h_i(o_{i3} + o_{i6}) \leq L \qquad \forall\, i, i' \in I, i < i' \qquad (3.16)$$
$$y_i + l_i(o_{i2} + o_{i6}) + w_i(o_{i1} + o_{i3}) + h_i(o_{i4} + o_{i5}) \leq W \qquad \forall\, i, i' \in I, i < i' \qquad (3.17)$$
$$z_i + l_i(o_{i3} + o_{i4}) + w_i(o_{i5} + o_{i6}) + h_i(o_{i1} + o_{i2}) \leq H \qquad \forall\, i, i' \in I, i < i' \qquad (3.18)$$

$$x_i + l_i(o_{i1} + o_{i5}) + w_i(o_{i2} + o_{i4}) + h_i(o_{i3} + o_{i6}) \leq x_{i'} + \left(1 - f_{ii'}^1\right)M \qquad \forall\, i, i' \in I, i < i' \qquad (3.19)$$

$$x_{i'} + l_{i'}(o_{i'1} + o_{i'5}) + w_{i'}(o_{i'2} + o_{i'4}) + h_{i'}(o_{i'3} + o_{i'6}) \leq x_i + (1 - f_{ii'}^2)M \qquad \forall\, i, i' \in I, i < i' \qquad (3.20)$$

$$y_i + l_i(o_{i2} + o_{i6}) + w_i(o_{i1} + o_{i3}) + h_i(o_{i4} + o_{i5}) \leq y_{i'} + \left(1 - f_{ii'}^3\right)M \qquad \forall\, i, i' \in I, i < i' \qquad (3.21)$$

$$y_{i'} + l_{i'}(o_{i'2} + o_{i'6}) + w_{i'}(o_{i'1} + o_{i'3}) + h_{i'}(o_{i'4} + o_{i'5}) \leq y_i + (1 - f_{ii'}^4)M \qquad \forall\, i, i' \in I, i < i' \qquad (3.22)$$

$$z_i + l_i(o_{i3} + o_{i4}) + b_i(o_{i5} + o_{i6}) + h_i(o_{i1} + o_{i2}) \leq z_{i'} + \left(1 - f_{ii'}^5\right)M \qquad \forall\, i, i' \in I, i < i' \qquad (3.23)$$

$$z_{i'} + l_{i'}(o_{i'3} + o_{i'4}) + w_{i'}(o_{i'5} + o_{i'6}) + h_{i'}(o_{i'1} + o_{i'2}) \leq z_i + (1 - f_{ii'}^6)M \qquad \forall\, i, i' \in I, i < i' \qquad (3.24)$$

$$o_{i1} + o_{i2} + o_{i3} + o_{i4} + o_{i5} + o_{i6} = 1 \qquad \forall i \in I \qquad (3.25)$$

Note that constraints 3-25 ensure that each item is placed into the vehicle with just one of the six orientations.

3.3 3DVLP with Allocation and Capacity Constraints

In some loading applications, items are partitioned into subsets according to their properties. Suppose that a subset here is the set of those items related to the same customer or destination, and each vehicle has a certain weight capacity.

The 3DVLP with allocation and capacity constraints (3DVLP-AC) is concerned with minimizing the total empty space of vehicles used to pack all items, while taking into account the following restrictions:

- Items of a particular subset (a customer) should be packed into the same vehicle (allocation constraints);
- The total capacity of items placed into a vehicle should not exceed the capacity of that vehicle in terms of weight and volume (capacity constraints);
- Items should be packed into the vehicles with a fixed vertical orientation without overlapping.

The modeling parameters and variables used in the mathematical formulation of 3VDLP-AC are:

n,	Total number of items;
m,	Total number of vehicles available in the depot;
c,	Total number of customers;
I_k,	The set of items related to the customer k;
d_k,	The capacity of customer *k* in terms of weight;
D,	The capacity of vehicle in terms of weight;
u_{ij},	A binary variable equal to 1 if item *i* is packed into vehicle *j* and 0 otherwise; and
v_j,	A binary variable equal to 1 if vehicle *j* is used and 0 otherwise.

There are only two orientations that can be used to place an item into a vehicle and maintain a fixed vertical orientation. For each item i, the binary variable o_i is defined as indicating the orientation of the item. The value of o_i is equal to one if the length of item *i* is parallel to the *X*-axis and its width is parallel to the *Y*-axis. Conversely, the value of o_i is equal to zero if width of item *i* is parallel to the *X*-axis and its length is parallel to the *Y*-axis.

Mathematical Formulation of 3DVLP-AC

$$Min\ (L.W.H\sum_{j=1}^{m} v_j - \sum_{i=1}^{n} l_i.w_i.h_i) \qquad (3\text{-}26)$$

Subject to :

$$x_i + (l_i - w_i)o_i + w_i \le L \qquad \forall i \in \{1 \dots n\} \qquad (3\text{-}27)$$

$$y_i + l_i + (w_i - l_i)o_i \le W \qquad \forall\, i \in \{1 \dots n\} \qquad (3\text{-}28)$$

$$z_i + h_i \le H \qquad \forall\, i \in \{1 \dots n\} \qquad (3\text{-}29)$$

$$x_i + (l_i - w_i)o_i + w_i \le x_{i'} + \left(1 - f_{ii'}^{1}\right)M \qquad \forall\, i, i' \in \{1 \dots n\}, i < i' \qquad (3\text{-}30)$$

$$x_{i'} + (l_{i'} - w_{i'})o_{i'} + w_{i'} \le x_i + (1 - f_{ii'}^{2})M \qquad \forall\, i, i' \in \{1 \dots n\}, i < i' \qquad (3\text{-}31)$$

$$y_i + l_i + (w_i - l_i)o_i \le y_{i'} + \left(1 - f_{ii'}^{3}\right)M \qquad \forall\, i, i' \in \{1 \dots n\}, i < i' \qquad (3\text{-}32)$$

$$y_{i'} + l_{i'} + (w_{i'} - l_{i'})o_{i'} \le y_i + (1 - f_{ii'}^{4})M \qquad \forall\, i, i' \in \{1 \dots n\}, i < i' \qquad (3\text{-}33)$$

$$z_i + h_i \le z_{i'} + \left(1 - f_{ii'}^{5}\right)M \qquad \forall\, i, i' \in \{1 \dots n\}, i < i' \qquad (3\text{-}34)$$

$$z_{i'} + h_{i'} \le z_i + (1 - f_{ii'}^{6})M \qquad \forall\, i, i' \in \{1 \dots n\}, i < i' \qquad (3\text{-}35)$$

$$f_{ii'}^{1} + f_{ii'}^{2} + f_{ii'}^{3} + f_{ii'}^{4} + f_{ii'}^{5} + f_{ii'}^{6} \ge u_{ij} + u_{i'j} - 1 \qquad \forall\, i, i' \in \{1 \dots n\}, i < i', \quad \forall j \in \{1 \dots m\} \qquad (3\text{-}36)$$

$$\sum_{k=1}^{c} \frac{d_k}{|I_k|} \sum_{i \in I_k} u_{ij} \le D\, v_j \qquad \forall\, j \in \{1 \dots m\} \qquad (3\text{-}37)$$

$$\sum_{j=1}^{m} u_{ij} = 1 \qquad \forall\, i \in \{1 \dots n\} \qquad (3\text{-}38)$$

$$u_{ij} = u_{i'j} \qquad \forall\, i, i' \in I_k, i < i', \forall k \in \{1 \dots c\}, \forall j \in \{1 \dots m\} \qquad (3\text{-}39)$$

$$u_{ij}, v_j, f_{ii'}^{1}, f_{ii'}^{2}, f_{ii'}^{3}, f_{ii'}^{4}, f_{ii'}^{5}, f_{ii'}^{6}, o_i \in \{0,1\} \qquad \forall\, i, i' \in \{1 \dots n\}, \forall j \in \{1 \dots m\} \qquad (3\text{-}40)$$

$$x_i, y_i, z_i \in \mathbb{Z}^{+} \qquad \forall\, i \in \{1 \dots n\} \qquad (3\text{-}41)$$

Objective function (3-26) minimizes the empty space of vehicles needed to pack all items. Constraints (3.27) to (3.29) ensure that an item that is orthogonally packed into a vehicle lies within the vehicle's geometric boundaries. Constraints (3-30) to (3-36) ensure that all items packed into a vehicle do not overlap each other. Constraints (3-37) ensure that the total capacity of customer's items serviced by a specific vehicle does not exceed the capacity of that vehicle in terms of weight. Constraints (3-38) guarantee that each item is packed into exactly one vehicle. All items associated with a customer should be packed into the same vehicle. This condition is ensured by constraints (3-39).

Model contains $4n + 3n(n-1) + \frac{m}{4}n(n-1) + m + \frac{m}{2}\sum_{k=1}^{c}|I_k|(|I_k|-1)$ constraints and $4n+m+\frac{7}{2}n(n-1)$ variables. It should be noted that the computational time for solving a model depends on the number of variables and constraints in the model and also optimization software used. All variables in the model are integer variables.

3.4 The Vehicle Routing Problem with Three-dimensional Loading Constraints

The integrated problem of vehicle routing with three-dimensional loading, known as the three-dimensional loading capacitated vehicle routing problem (3L-CVRP), was first described by Gendreau et al. [51]**.** Let us consider a complete graph $G = (V, E)$ where $V = \{0,1,\ldots,n\}$ is the vertex set and $E = \{(i,j): i,j \in V, i \neq j\}$ is the set of edges. Vertex 0 corresponds to the central depot and the other vertices correspond to the customers. A cost c_{ij} is associated with each edge (i, j) that represents traveling cost from vertex i to vertex j. Since $c_{ij} = c_{ji}$ for each $(i,j) \in E$, edges can be designated by a single index e. For each $S \subseteq V$, $\delta(S)$ is the set of edges with only one endpoint in S and $E(S)$ is the set of edges with both endpoints in S. The notation $\delta(i)$ is used rather than $\delta(\{i\})$ for simplicity.

Assume that a fleet of v identical vehicles is available in the central depot. Each vehicle has a weight capacity D and a three-dimensional loading space of length L, width W, and height H. The demand of each customer i (i = 1,…,n) is expressed in terms of a set of cuboid items CI_i with a total volume s_i and a total weight capacity d_i. It is assumed that $s_i \leq L.W.H$ and $d_i \leq D$. Each item $I_{ik} \in CI_i$ (k = 1, 2 . . . m_i) is characterized by the length l_{ik}, width w_{ik}, height h_{ik}, and fragility status f_{ik} ($f_{ik} = 1$ for fragile items and 0 for non-fragile ones).

The aim of the 3L-CVRP is to identify a set of vehicle routes with the minimum total traveling cost while satisfying the following constraints:

- The number of routes selected in the solution must be less than or equal to the number of vehicles available in the depot;
- Each route must start and end at the depot;
- Each customer must be served by exactly one vehicle and visited only once;
- The total weight capacity and volume of the items placed in a vehicle must not exceed the weight capacity and volume of that vehicle; and
- All items must be orthogonally packed into the vehicles without overlapping, while satisfying stability, fragility, and last in first out (LIFO) constraints defined below.

Fragility constraint: Only fragile items can be stacked on other fragile items; any items can be stacked on the non-fragile items.

Stability constraint: Each item that is not packed directly on the vehicle floor should be stable in the vehicle and supported by a sufficient surface comprising other items. The supporting surface of an item should be at least 75% of the base area of the item. Figure 3-3 gives examples of stable and unstable situations.

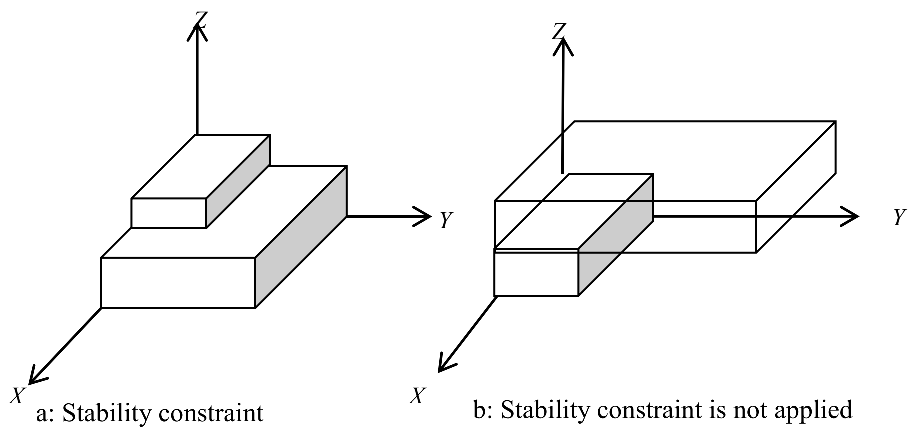


Figure 3-3: Situation of stability constraint for two cases

LIFO constraint: All items packed into a vehicle should be directly unloaded through a sequence of straight movements parallel to the length of vehicle without repositioning other items. When a customer is visited, its items should not be blocked by or stacked under items that will be delivered to customers later on the route. We have formulated the LIFO constraint for a single vehicle loading problem in which some dimensions of the vehicle are unknown (see Appendix A for more details).

The LIFO constraint mentioned here is suitable for cases where a forklift is used to unload the packed items. In this case, items are first elevated before being moved toward the rear of the vehicle. Tarantilis et al. [74] introduced a new version of the 3L-CVRP, called the capacitated vehicle routing problem with manual 3-D loading constraints (M3L-CVRP), where items are not essentially elevated before unloading. Figure 3-4 shows the LIFO policy for the 3L-CVRP and the M3L-CVRP. In this figure, the items specified by the dashed line should be unloaded before the other items.

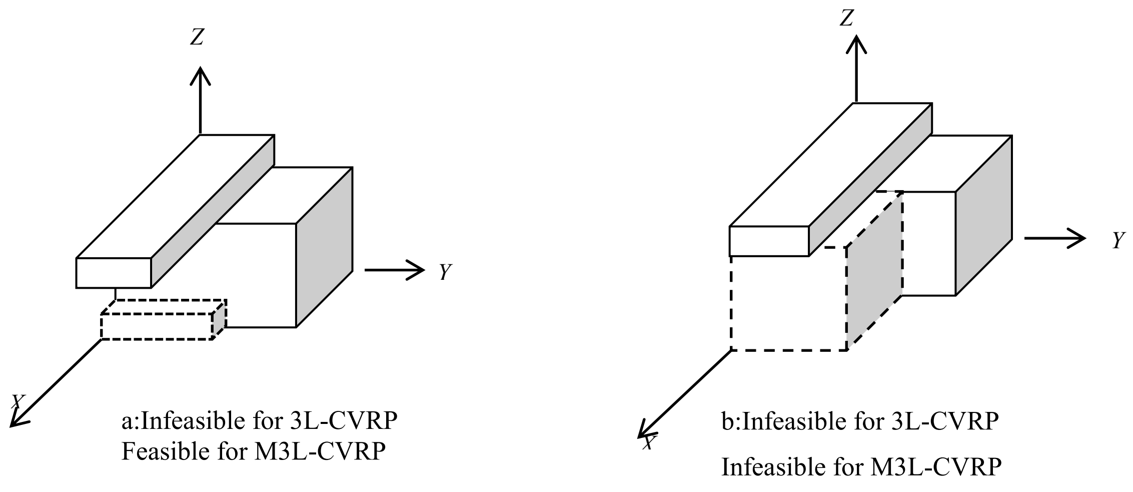


Figure 3-4: LIFO policy for the 3L-CVRP and the M3L-CVRP

Linear Programming Model

We present an integer linear programming formulation for 3L-CVRP adapted from 2L-CVRP formulation of Iori et al. [68]. For each $e \notin \delta(0)$, integer variable x_e is equal to one if edge e belongs to the optimal solution and zero otherwise. If $e \in \delta(0)$, then $x_e \in \{0,1,2\}$. Let us define σ as an order of visit of customers in S. $\sum(S)$ is the set of all sequences σ such that (S, σ) is a feasible route in 3L-CVRP. We define *r(S)* as the minimum number of vehicles needed to serve customers in set *S*.

$$min \sum_{e \in E} c_e x_e \tag{3-42}$$

Subject to:

$$\sum_{e \in \delta(i)} x_e = 2 \qquad \forall\, i \in V \setminus \{0\} \tag{3-43}$$

$$\sum_{e \in \delta(0)} x_e \leq 2K \tag{3-44}$$

$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \qquad \forall\, S \subseteq V \setminus \{0\}, S \neq \emptyset \tag{3-45}$$

$$\sum_{e \in E(S,\sigma)} x_e \leq |S| - 1 \qquad \forall\, (S,\sigma) | \sigma \notin \sum(S) \tag{3-46}$$

$$x_e \in \{0,1\} \qquad \forall\, e \notin \delta(0) \tag{3-47}$$

$$x_e \in \{0,1,2\} \qquad \forall\, e \in \delta(0) \tag{3-48}$$

$$K \in \{1,2,\dots,v\} \tag{3-49}$$

The degree constraints (3-43) ensure that exactly one arc enters and one arc leaves each customer vertex. Constraint (3-44) demands that the number of routes selected in the solution should be less than the number of vehicles available in the depot. This constraint also guarantees the degree requirements for the depot vertex. The connectivity and feasibility of the solution routes in terms of weight capacity and loading are ensured by constraints (3-45). We define *r(S)* as the minimum number of vehicles needed to pack all items associated with the customers in set *S*. *r(S)* can be obtained by solving an associated 3DVLP with allocation, capacity, stability and fragility constraints. Non-feasible customers' sequences for *S*, especially the sequences for which LIFO conditions are not satisfied, are eliminated by constraints (3-46).

Figure 3-5 gives an example of 3L-CVRP with four customers. A set of cuboid items as demand is associated with each customer. All customers' items are non-fragile except item 6. Two vehicles are used to serve customers. The order of visiting customers by each vehicle and items demanded by each customer are depicted in the figure.

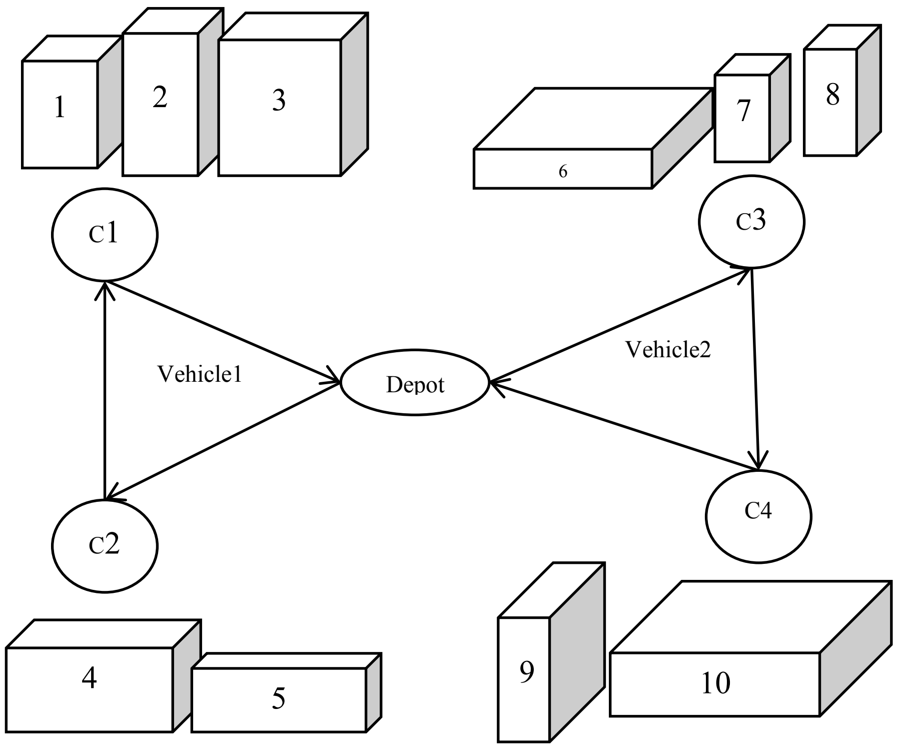


Figure 3-5: An example of 3L-CVRP with four customers and ten items

The loading pattern associated with each vehicle is specified in Figure 3-6. A loading pattern is given in accord with the fragility, stability and LIFO constraints. Since item 6 in vehicle 2 is a fragile item, no non-fragile items are stacked on the top of this item. It is seen form the figure that LIFO policies are satisfied for both loading patterns. For instance, in vehicle 1, items 4 and 5 related to customer 2 can be unloaded without removing other items. In vehicle 2, items 6, 7 and 8 related to customer 3 can be unloaded without removing other items available in the vehicle.

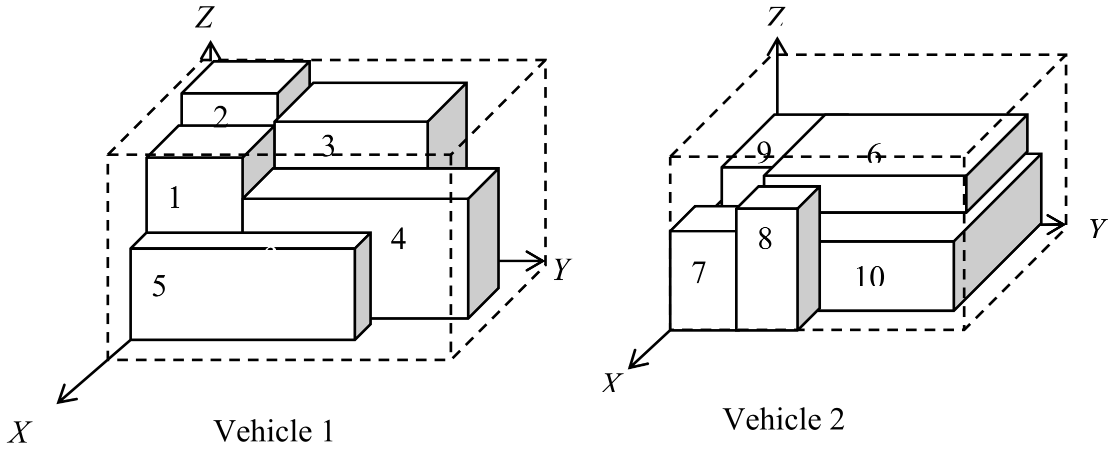


Figure 3-6: Feasible loading pattern for each vehicle

3.5 Conclusion

Mathematical models are very useful for clarifying problems and evaluating the quality of solutions realized from the heuristic algorithms. In this chapter, we presented the linear programming (LP) models for the 3DVLP, 3DVLP-AC, and 3L-CVRP. Mathematical formulations for certain constraints, such as stability and fragility, were not presented because of the complexity involved.

Chapter 4:
Solution Methods

Two solution methods, namely the column generation (CG) and tabu search (TS), are used for the problems introduced in the last chapter. Before discussing these methods in more details, however, we first introduce and develop an efficient heuristic method based on the concept of extreme points, which are used to check the feasibility of a given pattern in terms of loading. This heuristic method is called within the CG technique and TS method to check the loading feasibility. Figure 4-1 gives an overview of the solution methods used in this work.

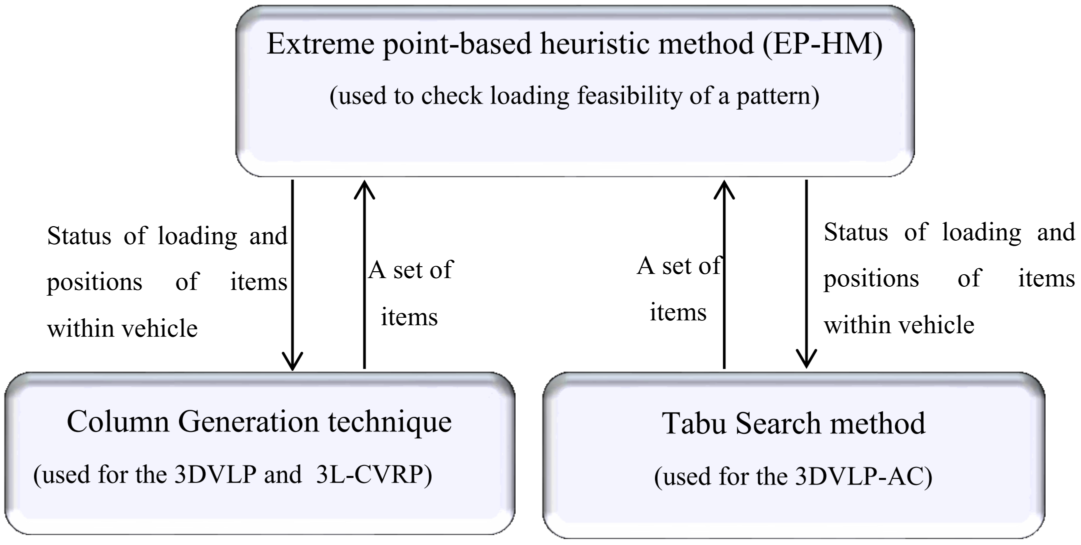

Extreme point-based heuristic method (EP-HM)
(used to check loading feasibility of a pattern)
Status of loading and positions of items within vehicle
A set of items
A set of items
Status of loading and positions of items within vehicle
Column Generation technique
(used for the 3DVLP and 3L-CVRP)
Tabu Search method
(used for the 3DVLP-AC)

Figure 4-1: An overview of solution approaches

4.1 Extreme Point-Based Heuristic Method

When using a heuristic method to check the loading feasibility of a sequence of items, its performance strongly depends on the definition of the physical points at which items are placed within the vehicle. The best definition for such points is based on the concept of extreme points (EPs) [28]. We detail this concept and develop an extreme point–based heuristic method (EP-HM), in which items are sequentially positioned in a vehicle based on EPs.

4.1.1 EPs Definition

Placing an item into a vehicle sub-divides that vehicle's space into new volumes. All possible empty volumes, where new items may be placed, are specified by the EPs. Figure 4-2 shows all EPs for two different loading patterns, each of which includes two items.

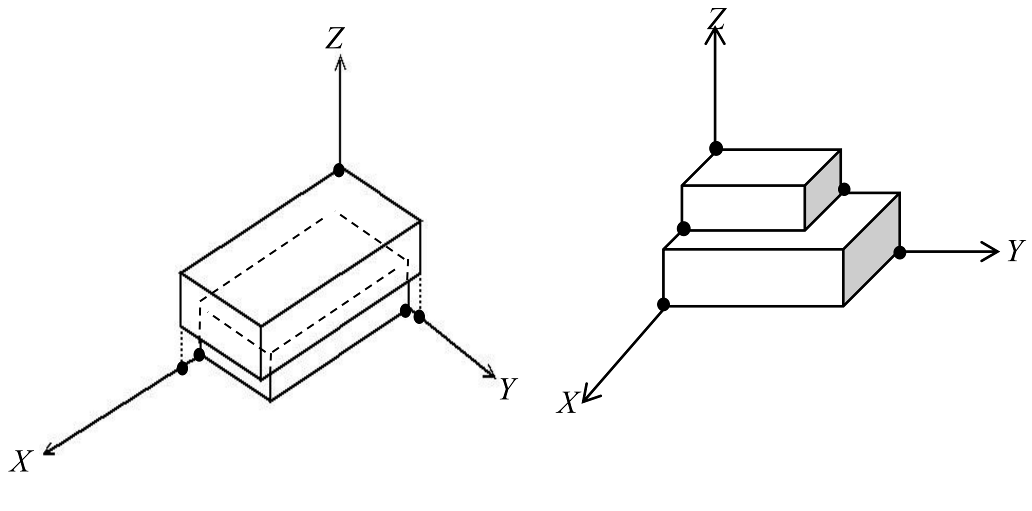


Figure 4-2: Extreme points (black circles) for two loading patterns, each including two items

The process of generating EPs comprises the following steps [28]:

- Placing the first item with dimensions $l_1 \times w_1 \times h_1$ into the empty vehicle at position (0, 0, 0) generates three EPs at points $(l_1, 0{,}0)$, $(0, w_1, 0)$, and $(0{,}0, h_1)$; and
- Adding item i with dimensions $l_i \times w_i \times h_i$ into a vehicle at position (x_i, y_i, z_i) yields the following new EPs (see Figure 4-3):
    - Projections of point $(x_i + l_i, y_i, z_i)$ along Y and Z axes;
    - Projections of point $(x_i, y_i + w_i, z_i)$ along X and Z axes; and
    - Projections of point $(x_i, y_i, z_i + h_i)$ along X and Y axes.

In each direction, if there are some items between item i and the wall of vehicle, then the associated projection points to the nearest item to item i.

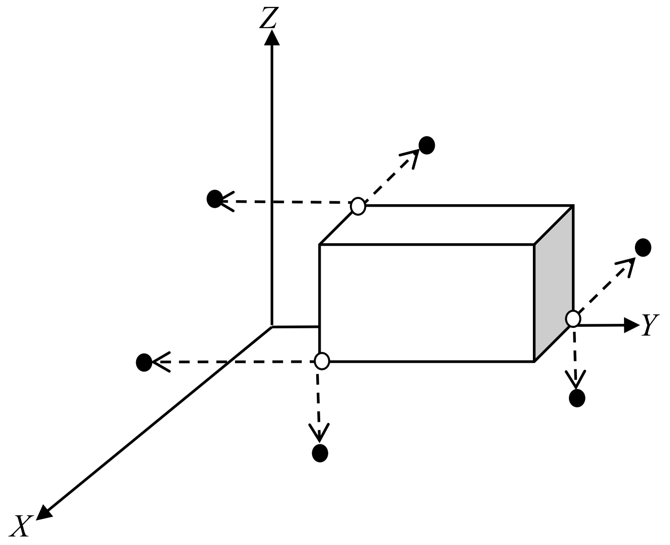


Figure 4-3: All possible EPs (black circles) generated by placing an item into a vehicle

Residual Space of EPs

The empty space around an EP is called its residual space (RS). The distance between an EP and the side of the vehicle along each axis is the RS of the EP on that axis. The RS of all EPs should be updated with each new item added to the loading pattern. The dashed cuboids in Figure 4-4 show the RS of an EP (specified by the black circle) before and after packing item *k*.

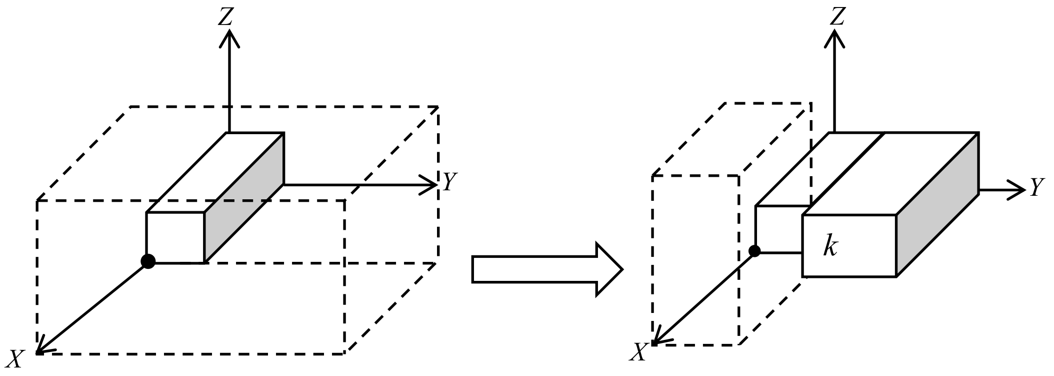


Figure 4-4: RS of an EP before and after packing item *k*

Introducing New EPs

Let us consider a loading pattern of two items, as shown in Figure 4-5. All EPs generated by the aforementioned process for such a pattern are illustrated by the black circles. To add a new item *k* with dimensions $l \times w \times h$ into the current loading, all EPs must first be scanned. In this instance, no EPs are appropriate for item *k*, because some dimensions of item *k* exceed the vehicle's geometric boundaries when item *k* is placed on each EP. The only position where item

k can be accommodated is the point specified by the white circle in Figure 4-5—although this point is not in the set of EPs. Therefore, we modify the process used to generate EPs.

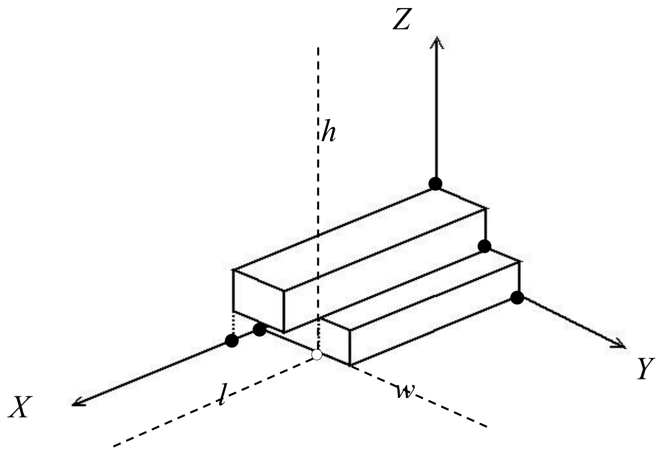


Figure 4-5: A loading pattern with all EPs specified by black circles

The process of generating new EPs comprises the following steps. Suppose that an item i with dimensions $l_i \times w_i \times h_i$ is placed into a vehicle in position (x_i, y_i, z_i). Let p_i be the projection of point $(x_i + l_i, y_i + w_i, z_i)$ along Z axis (see Figure 4-6). In such direction, if there are some items between item i and floor of vehicle, the associated projection points to the nearest item to item i. The projections of point p_i along X and Y axes are new EPs as specified by black circles in Figure 4-6. If there are some items between point p_i and the wall of vehicle in these directions, then the associated projections are on the nearest item to item i.

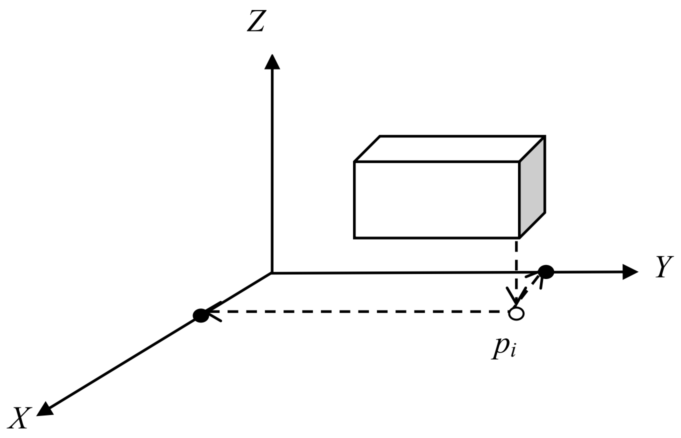


Figure 4-6: New EPs (black circles) after placing an item into vehicle

4.1.2 Items Sorting Rules

Different sorting rules can be tested for checking loading feasibility of a sequence of items when the order of placing items into a vehicle is not essential. Some items sorting rules are [28]:

- ***Volume-Height Rule***: Items are sorted according to non-increasing order of their volume; the items with similar volume are sorted according to non-increasing order of their height;
- ***Height-Volume Rule***: Items are sorted according to non-increasing order of their height; items with similar height are sorted according to non-increasing order of their volume;
- ***Area-Height Rule***: Items are sorted according to non-increasing order of their base plane; items with similar base plane are sorted according to non-increasing order of their height;
- ***Height-Area Rule****:* Items are sorted according to non-increasing order of their height; items with similar height are sorted according to non-increasing order of their base plane;

- ***Clustered Area-Height Rule*** [28]: For an integer value $\alpha \in [1{,}100]$, a set of intervals $A_{k,\alpha} = [\frac{(k-1)LW}{100}\alpha, \frac{kLW}{100}\alpha]$ is defined based on the vehicle's area $L \times W$. Items whose base areas are in the same interval are assigned into same group. Groups are ordered according to k and items of a group are sorted according to decreasing order of their height; and
- ***Clustered Height-Area Rule*** [28]: For an integer value $\alpha \in [1{,}100]$, a set of intervals $H_{k,\alpha} = [\frac{(k-1)H}{100}\alpha, \frac{kH}{100}\alpha]$ is defined based on the vehicle's height H. Items whose heights are in the same interval are assigned into same group. Groups are ordered according to k and items of a group are sorted according to decreasing order of their base area.

4.1.3 Pseudo code of EP-HM

Given a vehicle with dimensions $L{\times}W{\times}H$ and a list of sorted items called *SI*, Algorithm 4-1 is utilized to insert items of *SI* into a single vehicle using the concept of EPs.

Let *EPs* be the set of all EPs. First, *EPs* is initialized with the point (0,0,0). *RSs*, the set of all RSs, is initialized with (*L,W,H*) which is the RS of EP (0,0,0) along the *X*, *Y*, and *Z* axes, respectively. Each packed item and its position within the vehicle are saved in a set labeled *PI*.

For each item $i \in SI$, the *PutItemInVehicle* procedure (see Algorithm 4-2) scans *EPs* from the beginning to the end to identify an EP, *ep*, to determine where to place item *i* within the vehicle. An item can be placed on an EP if placement of its left-back-bottom corner on the EP does not overlap with the other items available in the vehicle. When LIFO, stability and fragility constraints are specified, such constraints need to be taken into account in order to select an appropriate EP. If there is more than one available EP, then the EP with the lowest *Z*, *Y*, *X* coordinates, in order, is selected.

Given the positions of the items already loaded into the vehicle, new item *i* and *ep*, the *Update-EPs* procedure generates new EPs and inserts them into the *EPs* (see Algorithm 4-3). In Algorithm 4-3, the *projection* procedure returns the value *true* if the projection of the point *k* related to item *i* lies on the side of item *j* along the required direction. The RS of the EPs is updated by the *UpdateResidualSpace* procedure (see Algorithm 4-4). Items are added to the vehicle as long as space is available in Algorithm 4-1. Once an item cannot be placed in the vehicle, then the algorithm stops. It should be noted that Algorithm 4-3 and 4-4 are same as Algorithms 1 and 2 in Crainic et al.[28].

Algorithm 4-1: EP-HM Procedure

```
Input: SI, (L,W,H )
Initialize vehicle v with dimensions $L \times W \times H$
EPs={(0, 0, 0)}
RSs={(L,W,H)}
PI= ∅
for each item $i \in SI$ do
        if PutItemInVehicle(i,RSs,EPs) then
                Update-EPs(PI,i,ep,EPs)
                UpdateResidualSpace(RSs,i,EPs)
                Remove item i from SI and add it to PI
        end if
        else
                PI= null and exit
        end else
```

```
14  end for
15  return $PI$
```

Algorithm 4-2: Putting Item into Vehicle Procedure

```
    $PutItemInVehicle(i,RSs,EPs)$
1   $ep=null$
2   for each $EP \in EPs$ do
3       if ($l_i \leq RS_{EP-X}$ and $w_i \leq RS_{EP-Y}$ and $h_i \leq RS_{EP-Z}$ and $x_{EP+}l_i \leq L$ and $y_{EP+}w_i \leq W$ and $z_{EP+}h_i \leq H$) then
4           $ep=EP$ and exit
5       end if
6   end for
7   return $ep$
```

Algorithm 4-3: EP Update Procedure

```
    $Update\text{-}EPs(PI,i, ep, EPs),ep=(x_i,y_i,z_i)$
1   $setBound[8]$=[-1,-1,-1,-1,-1,-1,-1,-1]
2   for all $j \in PI$ do
3       $k=(x_i ,y_i+w_i ,z_i)$
4       if $Projection(k,j,alongX\text{-}axis)$ and $x_j+l_j>setBound[1]$ then
5           $newEps[1]=(x_j+l_j,y_i+w_i,z_i)$
6           $setBound[1]=x_j+l_j$
7       end if
8       if $Projection(k,j,alongZ\text{-}axis)$ and $z_j+h_j >setBound[2]$ then
9           $newEps[2]=(x_i,y_i+w_i,z_j+h_j)$
10          $setBound[2]=z_j+h_j$
11      end if
12      $k=(x_i+l_i,yi,z_i)$
```

```
        **if** *Projection(k,j,alongY-axis)* and $y_j+w_j>setBound[3]$ **then**
            $newEps[3]=(x_i+l_i, y_j+w_j, z_i)$
            $setBound[3]=y_j+w_j$
        **end if**
        **if** *Projection(k,j,alongZ-axis)* and $z_j+h_j>setBound[4]$ **then**
            $newEps[4]=(x_i+l_i, y_i, z_j+h_j)$
            $setBound[4]=z_j+h_j$
        **end if**
        $k=(x_i, y_i, z_i+h_i)$
        **if** *Projection(k,j,alongX-axis)* and $x_j+l_j>setBound[5]$ **then**
            $newEps[5]=(x_j+l_j, y_i, z_i+h_i)$
            $setBound[5]=x_j+l_j$
        **end if**
        **if** *Projection(k,j,alongY-axis)* and $y_j+w_j>setBound[6]$ **then**
            $newEps[6]=(x_i, y_j+w_j, z_i+h_i)$
            $setBound[6]=y_j+w_j$
        **end if**
        $k=(x_i+l_i, y_i+w_i, z_i)$
        **if** *Projection(k,j,alongZ-axis)* and $y_j+w_j>setBound[7]$ **then**
            $newEps[7]=(x_i+l_i, y_i+w_i, z_j+h_j)$
            $setBound[7]=z_j+h_j$
        **end if**
    **end for**
    **for** all $j \in PI$ do
        $k=newEps[7]$, $setBound[7]=-1$
        **if** *Projection(k,j,alongX-axis)* and $x_j+l_j>setBound[7]$ **then**
            $newEps[7]=(x_j+l_j, y_k, z_k)$
            $setBound[7]=x_j+l_j$
        **end if**
```

```
42          if $Projection(k,j,alongZ\text{-}axis)$ and $z_j+h_j > setBound[8]$ then
43              $newEps[8]=(x_k,y_k,z_j+h_j)$
44              $setBound[8]=z_j+h_j$
45          end if
46      end for
47      for all $EP$ in $newEps[]$
48          $EPs=EPs+\{EP\}$
49      end for
50      return $EPs$
```

Algorithm 4-4: RS Update Procedure

```
        $UpdateResidualSpace(RSs,i,EPs)$,
1       for all $EP$ in $EPs$ do
2           if $(z_{EP} \geq z_i)$ and $(z_{EP} < z_i+h_i)$ then
3               if $(x_{EP} \leq x_i)$ and $EP$ is in space between $i$ and $YZ$ plane then
4                   $RS_{EP\text{-}X}=min(RS_{EP\text{-}X}, x_i-x_{EP})$
5               end if
6               if $(y_{EP} \leq y_i)$ and $EP$ is in space between $i$ and $XZ$ plane then
7                   $RS_{EP\text{-}Y}=min(RS_{EP\text{-}Y}, y_i-y_{EP})$
8               end if
9           end if
10          if $(z_{EP} \leq z_i)$ and $EP$ is in space between $i$ and $XY$ plane then
11              $RS_{EP\text{-}Z}=min(RS_{EP\text{-}Z}, z_i-z_{EP})$
12          end if
13      end for
```

Algorithm 4-3 generates 8 extreme points within a fixed time by two for-loops. Therefore this generation phase is $O(2|PI|)$. Suppose that n is the maximum number of items that can be packed into a vehicle, i.e. $|SI|=n$ and $|PI|<=n$. Then the overall complexity of Algorithm 4-3 is $O(n)$. Since both algorithms 4-2 and 4-4 are executed for each *EP* and the maximum number of *EPs* is $8n$, the complexity of each of mentioned algorithms is $O(8n)$. EP-HM algorithm requires $n(n+8n+n)$ operations, therefore the overall time complexity of EP-HM algorithm is $O(n^2)$.

4.2 Column Generation Technique

The column generation (CG) technique has been advocated by several researchers as a very powerful technique for solving a variety of operation research problems to optimality. CG is capable of solving LP models with a large number of variables. The technique was introduced by Ford and Fulkerson [89] to address a multi-commodity network flow problem. Then, Dantzig and Wolfe [90] adapted it to LP problems with a decomposable structure. Furthermore, Gilmore and Gomory [12,13] demonstrated the efficiency of the CG technique as applied to a cutting stock problem. The CG technique has also found applications in relation to the bin packing problem, the generalized assignment problem, the vehicle routing problem, the crew scheduling problem, the coloring, p-median problem, and other integer-constrained problems.

In general, the CG technique starts with a small part of the mathematical model of the problem, specifically, a partial set of the variables in the model. By solving this part, and analyzing the resultant partial solutions, more variables can be discovered and added to the partial model. Then, the expanded model is resolved. This process is repeated step by step to reach an acceptable solution for the entire model.

We deal with three problems in the CG technique, namely the master problem (MP), the restricted master problem (RMP), and the sub-problem. The LP relaxation of the original problem without integrality constraints is called the MP. Meanwhile, the corresponding LP relaxation model of the partial set of variables is called the RMP. The sub-problem is a new problem used to discover new variables.

According to the current dual values, the reduced costs of new variables are considered entries of the objective function in the sub-problem. Dual values are obtained by solving the RMP. Then, the objective function of the sub-problem (reduced costs) is updated with respect to the dual values. If variables with negative reduced cost are available, they are identified by the sub-problem. Such variables, expressed as new columns, are added to the RMP. Then, the RMP is resolved to generate a new set of dual values. This process is repeated until no father variables with a negative reduced cost are discovered. Figure 4-7 demonstrates the interactions between the MP, RMP, and sub-problem.

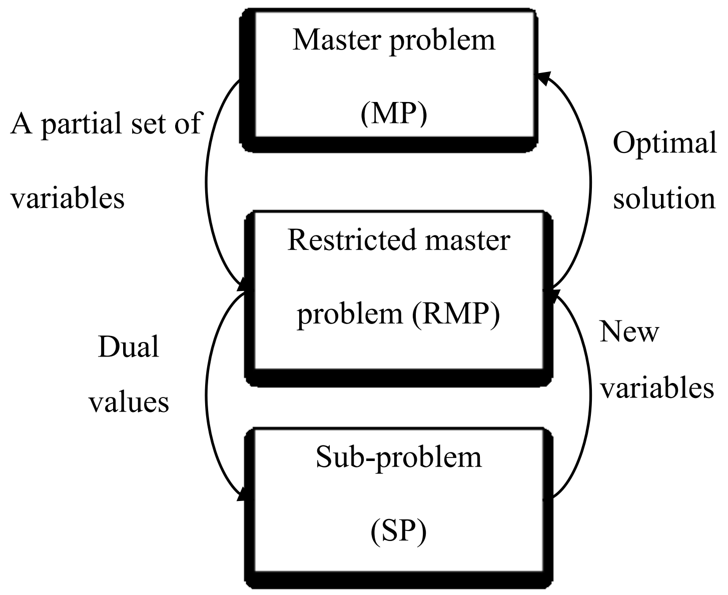


Figure 4-7: The interactions between MP, RMP and SP

Note that we try to obtain an optimal solution to the MP using CG technique. If this solution is integer, then it is an optimal solution to the LP of the original problem.

4.3 Tabu Search (TS) Method

Glover [91] [92] was first to propose a TS for addressing mathematical optimization problems. TS as a memory-based search strategy is an extension of the local search approach. This technique starts with an initial solution and moves iteratively from one solution to another, taken from the admissible neighborhoods of solutions, to reach the optimal one. As pointed out by Gendreau et al. [93], the search space and the neighborhood structure are two key elements of the TS method. It is critical that these be clearly defined before starting the TS. The search space is the space of all possible solutions that are visited during the search. A neighborhood structure discovers adjacent solutions that may be reached from the current solution for each step of the TS. To prevent cycling from one solution to another, a certain set of moves are designated as forbidden moves (called tabus). Tabus are recorded in a list that represents a short-term memory of the search. This list prevents returning to solutions that were previously examined and explores the neighborhood of all solutions that are still unexplored.

Aspiration Criteria

Tabus may exclude some moves with no cycles, but such moves may yield solutions that are better than any others that have been examined. To deal with this difficulty, tabus may be cancelled by applying an aspiration criterion. One of the aspiration criterion commonly used in TS method allows a move that gives a solution with an objective value better than the current best solution, even if that move is on the tabu list.

Stopping Criteria

The stopping criteria commonly used in the TS method are based on:

- Using a fixed number of iterations;
- Using a fixed amount of CPU time;
- Using a fixed number of consecutive iterations without any improvement in the objective function value; and
- Using pre-specified threshold value for the objective function.

TS Algorithm

Algorithm 4-5 presents the TS procedure. Let us define S and *f(S)* as current solution and its objective value. S^* and f^* are the best solution and its objective value respectively. TL is list of tabus. *giveBestN* procedure generates a subset of acceptable solutions in the neighborhood of S in accord with TL and aspiration criteria and then returns best solution from this subset.

Algorithm 4-5: TS Procedure

```
1    Select an initial solution $S$
2    Set $S^* = S$ , $f^* = f(S), TL = \emptyset$
3    **while** stopping criterion is not satisfied **do**
4        $S$=giveBestN(S)
5        **if** $f(S) < f^*$ **then**
6            Set $S^* = S$ , $f^* = f(S)$
7            Update $TL$
8        **end if**
9    **end while**
```

4.4 3DVLP

We propose a column generation–based solution to address the 3DVLP. We first have to express 3DVLP as a set-partitioning formulation (Dantzig-Wolfe decomposition) in order to use the CG technique. The key idea of a set-partitioning formulation for the 3DVLP is the enumeration of all feasible loading patterns in the problem. Let $\mathcal{L}$ be the set of all feasible loading patterns associated with a single vehicle. For each loading pattern $l \in \mathcal{L}$, a binary variable x_l is defined that is equal to one if the loading pattern l appears in the solution and zero otherwise. The binary parameter a_i^l is equal to one if the loading pattern *l* contains item $i \epsilon I$ and zero otherwise.

The set-partitioning formulation of 3DVLP is:

$$Min \sum_{l \in \mathcal{L}} x_l \tag{4.1}$$

Subject to:

$$\sum_{l \in \mathcal{L}} a_i^l x_l = 1 \qquad \forall\, i \in I \tag{4.2}$$

$$x_l \in \{0,1\} \qquad \forall\, l \in \mathcal{L} \tag{4.3}$$

Objective function (4.1) gives the minimum number of vehicles needed to load all items. Constraints (4-2) ensure that each item is packed into only one vehicle.

The set-partitioning model of 3DVLP that is relaxed by removing the integrality constraints on *x* variables is:

$$Min \sum_{l \in \mathcal{L}} x_l \tag{4.4}$$

Subject to:

$$\sum_{l \in \mathcal{L}} a_i^l x_l = 1 \qquad \forall\, i \in I \tag{4.5}$$

$$x_l \geq 0 \qquad \forall\, l \in \mathcal{L} \tag{4.6}$$

Let $\mathcal{L}'$ be a partial set of feasible loading patterns enumerated by obtaining an initial solution to the MP. RMP that is the corresponding model to $\mathcal{L}'$ presents as follow:

$$Min \sum_{l\in\mathcal{L}'} x_l \tag{4.7}$$

Subject to:

$$\sum_{l\in\mathcal{L}'} a_i^l x_l = 1 \qquad \forall\, i \in I \tag{4.8}$$

$$x_l \geq 0 \qquad \forall\, l \in \mathcal{L}' \tag{4.9}$$

Suppose that x_l' , $l \in \mathcal{L}'$ is optimal solution to the RMP and π_i', $i\epsilon\, I$ are dual optimal values associated with constraints (4.8). The question arising here is whether x_l', $l \in \mathcal{L}'$ is also an optimal solution to the MP or equivalently whether π_i', $i\epsilon\, I$ are optimal solution values to the dual of MP. In order to answer this question, we consider dual of MP.

Let π_i , $i\epsilon\, I$ be dual variables associated with constraints (4-5). Dual of MP (MPD) is:

$$Max \sum_{i\in\mathrm{I}} \pi_i \tag{4.10}$$

Subject to:

$$\sum_{i\in I} a_i^l \pi_i \leq 1 \qquad \forall\, l \in \mathcal{L} \tag{4.11}$$

$$\pi_i\ free \qquad \forall\, i \in I \tag{4.12}$$

If dual values π_i', $i \epsilon I$ satisfy constraints (4.11) for every feasible loading pattern $l \in \mathcal{L}$ then π_i', $i \epsilon I$ are optimal solution values to the MPD, thereby x_l' , $l \in \mathcal{L}'$ is optimal solution to the MP. In the other word, we should see if constraint $\sum_{i \in I} a_i^l \pi_i' \leq 1$ is satisfied for each loading pattern $l \in \mathcal{L}$. If this constraint is not satisfied by a loading pattern $l \in \mathcal{L}$ then π_i', $i \epsilon I$ are not optimal solution values to the MPD.

Let us define $c_l^\pi = 1 - \sum_{i \in I} a_i^l \pi_i$ as reduced cost associated with the loading pattern $l \in \mathcal{L}$. The feasible loading patterns with a negative reduced cost can improve the current solution to the RMP. For each iteration of CG technique, we should solve the sub-problem that consists of finding a feasible loading pattern $l \in \mathcal{L}$ with the minimum negative reduced cost. If no feasible loading patterns with the negative reduced cost exist then there are no columns (loading pattern) to be added to RMP. Therefore, the current solution is optimal to the RMP and MP.

4.4.1 Sub-problem (Pricing Problem)

The pricing problem consists of finding a feasible loading pattern of a single vehicle with the minimum negative reduced cost. In fact, pricing problem is a three-dimensional knapsack problem (3DKP) with the profit π_i for each item $i \epsilon I$ with size $l_i \times w_i \times h_i$ and knapsack size $L \times W \times H$.

Let v_i be the binary variable equal to one if item *i* is loaded into the vehicle (knapsack) and zero otherwise. Based on parameters and variables defined in Chapter 3, 3DKP can be formulated as follows:

$$Max \sum_{i \epsilon I} \pi_i v_i \tag{4.13}$$

Subject to:

$$x_i + l_i \le L \qquad \forall\, i \in I \tag{4.14}$$

$$y_i + w_i \le W \qquad \forall\, i \in I \tag{4.15}$$

$$z_i + h_i \le H \qquad \forall\, i \in I \tag{4.16}$$

$$x_i + l_i \le x_{i'} + \left(1 - f_{ii'}^1\right)M \qquad \forall\, i, i' \in I, i < i' \tag{4.17}$$

$$x_{i'} + l_{i'} \le x_i + (1 - f_{ii'}^2)M \qquad \forall\, i, i' \in I, i < i' \tag{4.18}$$

$$y_i + w_i \le y_{i'} + \left(1 - f_{ii'}^3\right)M \qquad \forall\, i, i' \in I, i < i' \tag{4.19}$$

$$y_{i'} + w_{i'} \le y_i + (1 - f_{ii'}^4)M \qquad \forall\, i, i' \in I, i < i' \tag{4.20}$$

$$z_i + h_i \le z_{i'} + \left(1 - f_{ii'}^5\right)M \qquad \forall\, i, i' \in I, i < i' \tag{4.21}$$

$$z_{i'} + h_{i'} \le z_i + (1 - f_{ii'}^6)M \qquad \forall\, i, i' \in I, i < i' \tag{4.22}$$

$$f_{ii'}^1 + f_{ii'}^2 + f_{ii'}^3 + f_{ii'}^4 + f_{ii'}^5 + f_{ii'}^6 \ge v_i + v_{i'} - 1 \qquad \forall\, i, i' \in I, i < i' \tag{4.23}$$

$$f_{ii'}^1, f_{ii'}^2, f_{ii'}^3, f_{ii'}^4, f_{ii'}^5, f_{ii'}^6, v_i \in \{0,1\} \qquad \forall\, i, i' \in I \tag{4.24}$$

$$x_i, y_i, z_i \in \mathbb{Z}^+ \qquad \forall\, i \in I \tag{4.25}$$

3DKP is NP-hard problem in the strong sense. Some solution techniques have been provided by Pisinger and Sigurd [15] for two-dimensional knapsack case, that are not efficient in terms of the computational time. Since pricing problem is called in each iteration of CG technique, we develop heuristic pricing method in order to speed up the entire CG algorithm.

4.4.2 Heuristic Pricing Solution Method

To generate feasible loading patterns with a negative reduced cost, we consider six orders of items, as described in the following.

Order 1: items are sorted by non-increasing order of π_i ;

Order 2: items are sorted by non-increasing order of $\frac{\pi_i}{h_i.w_i.l_i}$;

Order 3: items are sorted by non-increasing order of $\frac{\pi_i}{h_i+w_i+l_i}$;

Order 4: items are sorted by non-increasing order of $\frac{\pi_i}{h_i}$;

Order 5: items are sorted by non-increasing order of $\frac{\pi_i}{w_i}$; and

Order 6: items are sorted by non-increasing order of $\frac{\pi_i}{l_i}$.

According to each order rule, items are inserted into a vehicle using EP-HM algorithm until the residual capacity of the vehicle is sufficient to accommodate one more item. If the reduced cost of such a feasible loading pattern is negative, then it is added to the RMP as a new column.

The heuristic pricing method usually gives promising loading patterns with a negative reduced cost, but not necessarily the minimal. To reduce the total number of iterations and the overall computing time, all valid columns found in each iteration of the CG technique are added to the RMP. If no further columns with negative reduced costs can be found by this procedure, then the CG stops.

Since the time complexity a sorting algorithm for *n* items is *O*(n^2) and also time complexity of EP-HM algorithm is *O*(n^2). Therefore overall time complexity of pricing method is *O*(n^2).

4.5 3DVLP-AC

We use TS method to address 3DVLP-AC. The TS method for the 3DVLP-AC starts with assigning one customer to just one vehicle. To explore a new solution, a neighborhood structure is defined as a customer relocation procedure in which a customer is shifted from the current position to another vehicle. Given a solution, we try to discard a target vehicle, which is not declared tabu and has the minimum value of the filling function [94] defined as follows:

$$\varphi(j) = \frac{\sum_{i \in S_j} v_i}{V} + \frac{\sum_{i \in S_j} d_i}{D} - \frac{|S_j|}{n}$$

Here, S_j is the set of customers currently assigned to a vehicle j, D and V are capacities of a vehicle j in terms of weight and volume respectively, and n is the total number of customers in the problem.

Each customer's item in a target vehicle is relocated into a different vehicle with the maximum value of φ. The feasibility of a loading that results in this move is investigated using the EP-HM algorithm detailed in the last section. If a move is feasible, then its reverse move is kept in the tabu list in order to avoid any cycling. If all customers' items available in a target vehicle can be moved to the other vehicles without overlapping, then the target vehicle is discarded and the current best solution is updated. Otherwise it is kept in the tabu list and a new target vehicle is investigated. The TS method stops when either a given time limit is reached or the number of vehicles in the current solution is equal to a given lower bound. Note that the TS method may stop when there are no feasible moves to explore a neighborhood search after a certain number of iterations. Algorithm 4-6 gives a schema of the TS method for the 3DVLP-AC with the following notation:

BS: Best solution;

LB: Lower bound;

TL: Tabu List;

n: Number of Customers;

TV: Target Vehicle; and

Sv: the set of all vehicles in the current solution.

Algorithm 4-6: TS Method for 3DVLP-AC

```
1   Load each customer into separate vehicle and set $BS=n$
2   $TL = \emptyset$
3   while time limit is not reached or $BS \neq LB$ do
4       Select $TV=\text{argmin}\varphi(j), j \in Sv$, with respect to the $TL$
5       for all customer $c \in TV$ do
6           $TV'=\text{argmax}\varphi(j), j \in Sv$ and $j \neq TV$
7           if EP-HM($c,TV'$) is true and move is not in $TL$ then
8               Add $c$ to $TV'$ and remove $c$ from $TV$
9               Update $TL$
10          end if
11      end for
12      if all customers' items of $TV$ are replaced into other vehicles then
13          $BS=BS-1$
14          Update $TL$
15          $Sv=Sv \setminus TV$
16      end if
17  end while
```

4.6 3L-CVRP

We propose a column generation–based solution to address the 3L-CVRP. We first express 3L-CVRP as set-partitioning formulation (Dantzig-Wolfe decomposition) in order to use CG technique. The key idea of set-partitioning formulation for the 3L-CVRP is to enumerate all feasible routes of the problem. A feasible route is defined as a vehicle trip that starts and ends at the depot, while visiting a subset of customers and satisfying capacity and loading constraints detailed in the problem description chapter.

Let $\mathcal{R}$ be the set of all possible feasible routes. The parameters and variables used in the set-partitioning formulation of 3L-CVRP are:

m,	The number of vehicles available in the depot;
c_{ij},	The traveling cost from customer *i* to customer *j*;
$c_r = \sum_{(i,j)\in r} c_{ij}$,	The cost of feasible route $r \in \mathcal{R}$, that is sum of traveling costs along route r;
a_{ir},	The binary parameter equal to 1 if customer *i* is along feasible route *r* and 0 otherwise; and
x_r,	The binary variable equal to 1 if feasible route *r* is selected in the solution and 0 otherwise.

The set-partitioning formulation of 3L-CVRP is:

$$Min \sum_{r\epsilon\mathcal{R}} c_r x_r \tag{4-26}$$

Subject to:

$$\sum_{r\epsilon\mathcal{R}} a_{ir} x_r = 1 \qquad \forall\, i \in V\backslash\{0\} \tag{4-27}$$

$$\sum_{r\in\mathcal{R}} x_r \leq m \tag{4-28}$$

$$x_r \in \{0,1\} \qquad \forall\, r \in \mathcal{R} \tag{4-29}$$

A set of feasible routes with the minimum total traveling cost is determined in the objective function (4-26). Constraints (4-27) ensure that each customer is covered by exactly one of the feasible routes selected in the solution. Constraint (4-28) ensures that the number of routes selected in the solution does not exceed the number of vehicles.

A linear programming (LP) relaxation of the set-partitioning formulation (called MP) without integrality constraints on *x* variables is:

$$Min \sum_{r \epsilon \mathcal{R}} c_r x_r \tag{4-30}$$

Subject to:

$$\sum_{r \epsilon \mathcal{R}} a_{ir} x_r = 1 \qquad \forall\, i \in \mathrm{V} \backslash \{0\} \tag{4-31}$$

$$\sum_{r \in \mathcal{R}} x_r \leq m \tag{4-32}$$

$$x_r \geq 0 \qquad \forall\, r \in \mathcal{R} \tag{4-33}$$

Let $\mathcal{R}'$ be a partial set of feasible routes enumerated by obtaining an initial solution to the MP, Corresponding model to this set (called RMP) is:

$$Min \sum_{r \epsilon\, \mathcal{R}'} c_r x_r \tag{4.34}$$

Subject to:

$$\sum_{r \epsilon\, \mathcal{R}'} a_{ir} x_r = 1 \qquad \forall\, i \in \mathrm{V} \backslash \{0\} \tag{4-35}$$

$$\sum_{r \in\, \mathcal{R}'} x_{rj} \leq m \tag{4.36}$$

$$x_r \geq 0 \qquad \forall\, r \in\, \mathcal{R}' \tag{4.37}$$

Suppose that $\pi_i',\ i\epsilon\ V\backslash\{0\}$ be dual values associated with the constraints (4-35) and μ'dual value associated with constraint (4-36) form current solution x_r' , $r \in \mathcal{R}'$ to the RMP. In order to recognize whether x_r' , $r \in \mathcal{R}'$ are also optimal values to the MP or equivalently whether $\pi_i',\ i\epsilon\ V\backslash\{0\}$ and μ' are optimal values to the dual of MP, we consider dual of MP.

Let $\pi_i,\ i\epsilon\ V\backslash\{0\}$ and μ be dual variables associated with constraints (4.31) and (4.32) in the MP. Dual of MP (called MPD) is:

$$Max \sum_{i\in V\backslash\{0\}} \pi_i + m\mu \qquad (4.38)$$

Subject to:

$$\sum_{i\in V\backslash\{0\}} a_{ir}\pi_i + \mu \leq c_r \qquad \forall\, r \in \mathcal{R} \qquad (4.39)$$

$$\mu \leq 0 \qquad (4.40)$$

$$\pi_i\, free \qquad \forall\ i\epsilon\ V\backslash\{0\} \qquad (4.41)$$

If dual values (π_i' , μ') satisfy constraints (4.39) then they are optimal solution values to the MPD, thereby x_r' , $\forall\, r \in \mathcal{R}$ are optimal values to the MP. In fact we should see if constraint $\sum_{i\in V\backslash\{0\}} a_{ir}\pi_i' + \mu' \leq c_r$ is satisfied for each route $r \in \mathcal{R}$. We define $\bar{c}_r = c_r - \mu - \sum_{i\in V\backslash\{0\}} a_{ir}\pi_i$ as reduced cost associated with the route $r \in \mathcal{R}$.

CG technique is initialized by assigning each customer to exactly one vehicle. Since constraint (4-36) is not satisfied with such initial columns, an artificial variable *y* is added to constraint (4-36) with a large penalty in the objective function. A basic scheme of CG technique for the 3L-CVRP is given in Algorithm 4-7.

Algorithm 4-7: CG Technique for the 3L-CVRP

```
1  Generate a partial set of routes (columns) by assigning each customer to one vehicle
2  Solve corresponding RMP and obtain dual values
3  Solve sub-problem to identify feasible routes with the negative reduced cost
4  Add feasible routes to RMP as new columns and go to step 2
5  If there are no feasible routes with negative reduced cost then stop, the current
   solution to the RMP is optimal to the MP
```

4.6.1 Sub-problem (Pricing Problem)

The pricing problem for the 3L-CVRP consists of finding a feasible route *r* with the minimum negative reduced cost for a single vehicle. The kind of pricing problem is an elementary shortest path problem with resource and additional constraints (ESPPRC) such as weight capacity, volume, and loading conditions.

As aforementioned, in graph $G = (V, E)$, V is the vertex set containing n customers and central depot $\{0\}$ and $E = \{(i,j): i,j \in V, i \neq j\}$ is the set of edges. With replacing each edge with two arcs, graph G can be represented as directed graph $G^{'}$ with the set of arcs A. Without loss of generality, depot node in G is replaced with two nodes s and e which are nodes of start and end in graph $G^{'}$, respectively. Let $P_r = \{(s,i), \ldots \ldots (k,e)\}$ be the set of arcs (path) associated with the route $r \in \mathcal{R}$. $\bar{c}_r = c_r - \mu - \sum_{i \in V \setminus \{0\}} a_{ir}\pi_i$, reduced cost of route $r$can be reformulated as follows:

$$\bar{c}_r = \sum_{(i,j) \in P_r} \bar{c}_{ij}$$

$$\bar{c}_{ij} = \begin{cases} c_{ij} - \pi_j & \forall (i,j) \in P_r, j \neq e \\ c_{ij} - \mu & \forall (i,j) \in P_r, j = e \end{cases}$$

The reduced cost $\overline{c}_{ij}$, volume s_j and weight d_j are associated with each arc *(i,j)* in graph $G^{'}$ (see Figure 4-8).

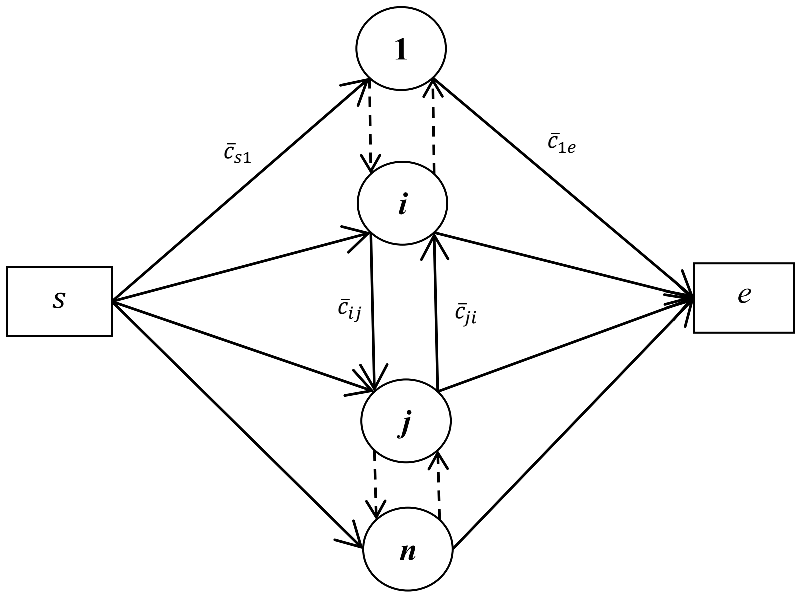


Figure 4-8: Sub-problem network $G^{'}$ for pricing problem in 3L-CVRP

Let us define F as set of all feasible routes satisfying loading constraints. We define binary variable x_{ij} equal to one if arc *(i,j)* is selected in the solution and zero otherwise. The problem of elementary shortest path from point *s* to point *e* while satisfying resource and loading constraints is formulated as follows:

$$min \sum_{(i,j)\in A} \bar{c}_{ij} x_{ij} \tag{4-42}$$

Subject to:

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = 0 \qquad \forall\ i \in V \setminus \{s,e\} \tag{4-43}$$

$$\sum_{(s,j)\in A} x_{sj} = 1 \tag{4-44}$$

$$\sum_{(i,e)\in A} x_{ie} = 1 \tag{4-45}$$

$$\sum_{(i,j)\in A} x_{ij} \leq 1 \qquad \forall\, i \in V \setminus \{s,e\} \tag{4-46}$$

$$\sum_{(i,j)\in A} d_j x_{ij} \leq D \tag{4-47}$$

$$\sum_{(i,j)\in A} s_j x_{ij} \leq SV \tag{4-48}$$

$$\{x_{ij} \,|(i,j) \in A\} \in F \tag{4-49}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{4-50}$$

A path with the minimum reduced cost is selected in the objective function (4-42). Constraints (4-43) to (4-45) are flow conservations. Constraints (4-46) ensure that there exist no cycles in the path selected as solution. Constraints (4-47) and (4-48) ensure that the total weight capacity and volume of arcs selected in the solution do not exceed the weight capacity and volume of vehicle respectively. The loading feasibility of arcs is ensured by constraint (4-49).

Pricing Problem Solution Method

The ESPPRC with the loading constraints is strongly NP-hard. We solve the ESPPRC via a label-correcting algorithm with the dominance rules introduced in Feillet et al. [95] and Desrochers [96], without considering loading constraints. Then, the loading feasibility of a route with a negative reduced cost resulting from the ESPPRS is verified by the EP-HM algorithm. If EP-HM cannot give a feasible loading pattern for such a route, then a saving algorithm is applied in order to obtain a valid column. Figure 4-9 gives an overview of the algorithms employed to obtain a valid column in the pricing problem.

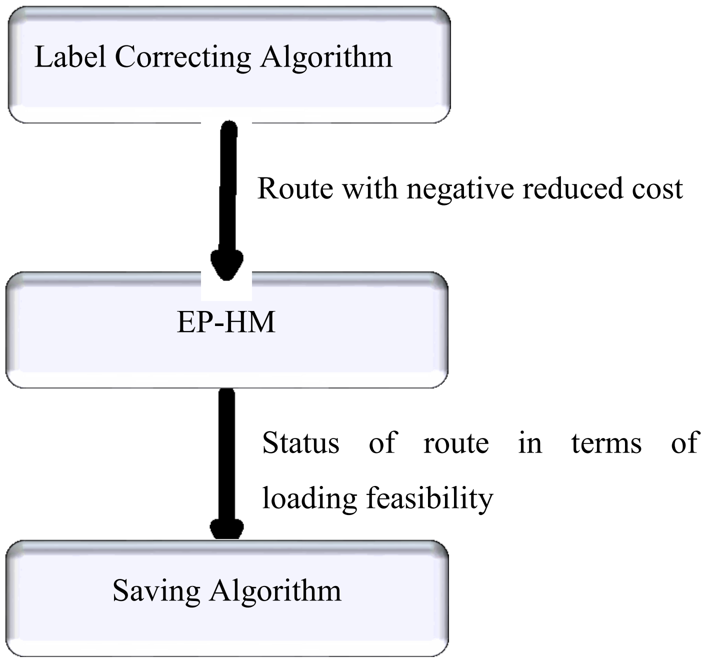

Label Correcting Algorithm
Route with negative reduced cost
EP-HM
Status of route in terms of loading feasibility
Saving Algorithm

Figure 4-9: An overview of the algorithms used in the pricing problem

Label Correcting Algorithm

Label correcting algorithm is an extension of the Ford-Bellman algorithm [97] where a set of labels is associated with each possible partial path. A label indicates the consumption of a resource along the partial path. In the label correcting algorithm, labels are eliminated using dominance rules, and nodes are repeatedly considered while their labels are extended.

Let *R* be the number of resources and $d_{ij}^r \geq 0$ the value of resource *r* along arc *(i,j)*. For each resource *r,* an interval $[a_i^r, b_i^r]$ is associated with each node *i.* The consumption of resource *r* along a path from *s* to *i* should be in interval $[a_i^r, b_i^r]$. Note that triangle constraint is satisfied for each resource value d_{ij}^r. Since here we deal with the volume and weight resources, only two intervals [*0, D*] and [*0, V*] are associated with each node.

Label Definition

According to Feillet et al. [95], label (L_i, C_i) for each path p_{si} and node *i* is defined as follows:

- $L_i = (U_i^1, U_i^2, \dots, U_i^R, n_i, v_i^1, v_i^2, \dots v_i^n)$; and
- C_i = cost of path p_{si}.

Here, $U_i^1, U_i^2, \dots, U_i^R$ are values of *R* resources used along path p_{si}, n_i is the number of unreachable nodes and $(v_i^1, v_i^2, \dots v_i^n)$ is the vector of unreachable nodes. v_i^k is equal to one if node k is unreachable and zero otherwise. A node is considered as unreachable if it is along path p_{si} or it cannot be visited anymore due to the resource constraints.

Dominance Rule

Suppose that P^1 with label (L_i^1, C_i^1) and P^2 with label (L_i^2, C_i^2) are two different paths from *s* to *i*. Path P^1 dominates path P^2 in node *i* if and only if following conditions are satisfied:

- $C_i^1 \leq C_i^2$;
- $n_i^1 \leq n_i^2$;

- $U_i^{1r} \leq U_i^{2r}$ for r=1,2,…,R;
- $v_i^{1k} \leq v_i^{2k}$ for k=1,2,…,n; and
- $(L_i^1, C_i^1) \neq (L_i^2, C_i^2)$.

Only non-dominated paths are investigated to obtain the shortest path in the network.

Pseudo Code of Label Correcting Algorithm

A minimum cost elementary path from *s* to each node is obtained in Algorithm 4-8. Let *L(i)* be the set of all labels on the node *i* and *S(i)* be the set of nodes after node *i*. Define *T* as list of waiting nodes to be treated and E_{ij} as set of extended labels form node *i* to node *j*. *DoExtension* procedure gives extended label resulted from label L_i to node *j*. This procedure returns nothing if an extension is impossible. *NoDo* procedure uses the dominate rules to remove some labels from set *L*.

Algorithm 4-8: Label Correcting Procedure

```
Output: Minimum cost elementary path from point s to each point
1   set $L(s)=\{(0,0)\}$
2   for all $i \in V \setminus \{s\}$ do
3       Set $L(i)=\emptyset$
4   end for
5   Set $T=\{s\}$
6   while $T$ is not empty do
7       Select $i \in T$
8       for all $j \in S(i)$ do
9           Set $E_{ij} = \emptyset$
10          For all $(L_i, C_i) \in L(i)$ do
11              if $v_i^j$=0 then
12                  $E_{ij} = E_{ij} \cup \{DoExtension(L_i, j)\}$
```

```
13              end if
14          end for
15          $L(j)=NoDo(L(j) \cup E_{ij})$
16          if there is change in $L(j)$ then
17              $T=T\cup\{j\}$
18          end if
19      end for
20      $T=T\setminus\{i\}$
21  end while
```

The time complexity of this label correcting algorithms depends strongly on the graph structure, number of nodes and also tightness of resource constraints [95].

Loading Feasibility Check

The label correcting algorithm gives a route with a negative reduced cost without considering its loading feasibility. To check the loading feasibility of such a route, the EP-MH algorithm is examined for a combination of three items sorting rules and different position selecting criteria. For all three sorting rules (called *Rule-1, Rule-2, Rule-3*), customers along the route are sorted by inverse order of visit. For each customer non-fragile items precede fragile ones. Each of the subsets of fragile and non-fragile items is sorted by following criteria for each sorting rule:

- *Rule-1*: Decreasing volume, breaking ties by decreasing height;
- *Rule-2*: Decreasing base area, breaking ties by decreasing height; and
- *Rule-3*: Decreasing height, breaking ties by decreasing base area.

The extreme points are scanned according to the six following position selection criteria:

- *Criterion-1*: Select the EP with the minimum *Y* coordinate value, breaking ties with the *X* coordinate value and then minimum *Z* coordinate value;
- *Criterion-2*: Select the EP with the minimum *X* coordinate value, breaking ties with the minimum *Y* coordinate value and then minimum *Z* coordinate value;
- *Criterion-3*: Select the EP with the minimum *Y* coordinate value, breaking ties with the minimum *Z* coordinate value and then minimum *X* coordinate value [34];
- *Criterion-4*: Select the EP at which the item to be packed, touches the vehicle and other packed items more [20];
- *Criterion-5*: Select the EP with the minimum residual space along *X* and *Y* axes [28]; and
- *Criterion-6*: Select the EP with the minimum residual space [28].

Saving Algorithm

If the EP-HM algorithm cannot give a feasible loading pattern for a route with a negative cost, then, a saving algorithm is used (see Algorithm 4-9). The saving algorithm eliminates some customers from a route which represent a smaller effect on the reduced cost in order to obtain a feasible loading pattern.

Let $(s,..,i,k,j,...e)$ be an elementary path with the negative reduced cost given by the label correcting algorithm. The saving algorithm is based on notation of saving that is:

$$s_k = \bar{c}_{ij} - \bar{c}_{ik} - \bar{c}_{kj}.$$

According to the value of s_k, the least profitable customer along path $(s,..,i,k,j,...e)$ is selected and removed from the route. The loading feasibility of this path is then checked. If a valid loading pattern is not derived, then the next least profitable customer is removed from the path. The saving algorithm stops when a valid loading pattern is achieved or the reduced cost of the path is positive.

Algorithm 4-9: Saving Procedure

```
1   Calculate $s_k$ for each customer $k$ in path $(s, .., i, k, j, ... e)$
2   Select customer $k$ with the minimum value $s_k$ and remove it from the path
3   **if** the reduced cost of path is positive or there is a valid loading for the route **then**
4           exit
5   **else**
6           Go to step1
```

4.6.2 Heuristic Pricing Approach

We have developed a greedy heuristic pricing approach to hasten the generation of valid columns with a negative reduced cost (see Algorithm 4-10). Algorithm 4-10 gives promising feasible routes with negative, but not necessarily minimal reduced cost. For each $i \in V$, L_i is a list of $j \in V, j \neq i$, sorted by increasing order of $\bar{c}_{ij}$. The algorithm starts from L_d, which is a list of customers $j \in V \backslash d$ sorted by increasing order of $\bar{c}_{dj}$. Let k be the first entry in list L_d. Items associated with customer k are inserted into a vehicle by the EP-HM algorithm. Then, the first entry in list L_k is selected. If this entry is a customer whose items cannot be packed into the vehicle according to the loading constraints, then the second entry in L_k is selected. This process is repeated until the selected entry is depot. A feasible loading pattern with a negative reduced cost (called *fLoading*) is added to the set *all-Loadings*. For a feasible loading pattern with a positive reduced cost, the *checkR* procedure removes one or more customers with the lowest value of π from *fLoading* in order to obtain a feasible loading pattern with a negative reduced cost. We apply Algorithm 4-10 for different position criteria, as explained in the previous section.

Suppose that *n* is the total number of items and *m* the total number of customers. Aforementioned the time complexity of EP-HM algorithm is $O(n^2)$, and it is seen that this algorithm is called within heuristic pricing procedure for each customer. Moreover the overall complexity time of all sorting rules in the pricing algorithm is $O(m^3)$. Therefore the overall time complexity of pricing method is $O(m^3+mn^2)$.

Algorithm 4-10: Heuristic Pricing Procedure for the 3L-CVRP

```
Input V:the set of customers and depot; ($\pi,\mu$): the dual values
Output all-Loadings: all feasible loading patterns with the negative reduced costs
1   Obtain $L_i$ for each $i \in V$
2   for each customer $k \in L_d$ do
3       Pack items of customer k by EP-HM algorithm
4       Add customer k to fLoading
5           Set CL=$L_k$
6           for each $j \in CL$ do
7               if j is depot then
8                   Go to step 15
9               end if
10              if items of customer j can be packed then
11                  Add customer j to fLoading
12                  Set CL= $L_j$ and go to step 6
13              end if
14          end for
```

```
        if reduced cost of fLoading is negative then
            Inverse the order of customers in fLoading
            Add fLoading to all-Loadings
        end if
        if reduced cost of fLoading is positive then
        checkR(fLoading)
        end if
    end for
    return all-Loadings
```

4.7 Branching Rules

After terminating the CG technique, if the current solution to the MP is an integer, then it is either a near-optimal or an optimal solution to the set-partitioning model. Otherwise, an optimal integer solution needs to be determined through the branch and bound method embedded in the CG technique; this is called the branch and price (B&P) method. Note that the non-integer optimal solution of the MP is used as a lower bound for the optimal solution of the set-partitioning model.

Some modifications are applied to the sub-problem in order to avoid regenerating columns in the branching. Infeasible columns in accord with the branching constraints are not be generated using these modifications. Two following branching strategies are used in this work.

Depth-First Heuristic (D_FH) Branching Rule

In this branching rule, variables with fractional values that are greater than a given specific value are set to one. If there are no variables with appropriate fractional values, then the variable with the largest fractional value is fixed to one. After fixing the variables, more columns are generated using a heuristic pricing algorithm until there is little or no change in the RMP solution. Repeating this branching strategy leads either to an integer solution or an infeasible one.

Ryan and Foster Branching Rule

This branching rule is a general rule for the set-partitioning problem originally suggested by Ryan and Foster [98]. We apply such branching rule to the 3DVLP.

Given a fractional LP solution to the set-partitioning model, branching is applied on a pair of items selected by one of the following rules:

- Select items *i* and *j* with the maximum volumes from a loading pattern solution that has the high fractional value; and
- Select items *i* and *j* with the maximum volumes from a loading pattern solution whose corresponding value is closest to 0.5.

The pair of items i and j should be assigned to the same vehicle in one branch and to different vehicles on the other branch. The first branch is performed by adding constraint $\sum_{l\in\mathcal{L}} a_i^l a_j^l x_l = 1$ to the mathematical model and the second branch by adding two constraints $\sum_{l\in\mathcal{L}}(1 - a_i^l)a_j^l x_l = 0$ and $\sum_{l\in\mathcal{L}} a_i^l(1 - a_j^l)x_l = 0$ to the model.

All loading patterns that contain just item *i* or *j* are individually eliminated from the RMP on the branch where a pair *i* and *j* should be in the same vehicle. All loading patterns containing both items *i* and *j* are eliminated from the RMP on the branch where a pair of items *i* and *j* should not be in the same vehicle.

4.8 A Numerical Example of the CG Technique

An example of the vehicle routing problem without capacity and loading constraints is presented and solved by means of CG technique to understand better the strategy of this technique. Suppose that there exist two identical vehicles in the depot. Such vehicles should meet a set of three customers. Figure 4-10 shows the road network including customers, depot as well as traveling cost between each pair of customers.

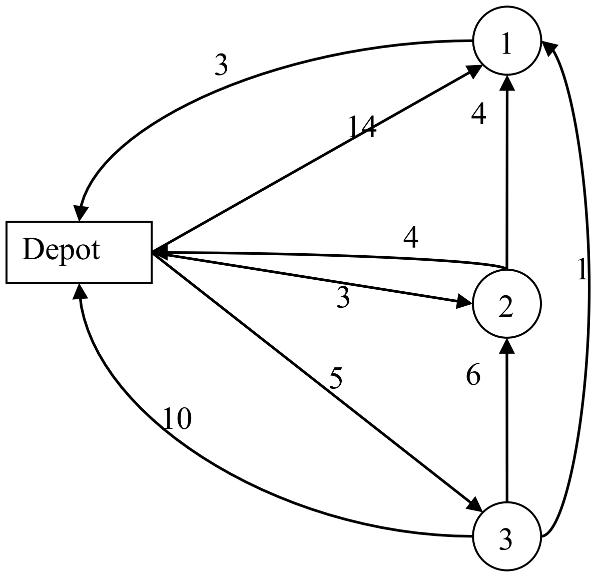


Figure 4-10: Road network of an example of the vehicle routing problem

Set-Partitioning Formulation

Let x_r be binary variable equal to one if route *r* is selected in the solution and zero otherwise. c_r is total cost of route *r*. *R* is the set of all feasible routes. a_{ir} is binary parameter equal to one if customer *i* is visited by route *r* and zero otherwise. The set partitioning formulation is:

$$Min \sum_{r \in R} c_r x_r$$

Subject to:

$$\sum_{r \in R} a_{ir} x_r = 1 \qquad i=1,2,3$$

$$\sum_{r \in R} x_r \leq 2$$

$$x_r \in \{0,1\} \qquad \forall\, r \in R$$

Master Problem

$$Min \sum_{r \in R} c_r x_r$$

Subject to:

$$\sum_{r \in R} a_{ir} x_r = 1 \qquad i=1,2,3$$

$$\sum_{r \in R} x_r \leq 2$$

$$x_r \geq 0 \qquad \forall\, r \in R$$

CG technique starts with $R' = \emptyset$. The corresponding RMP with the artificial variables a_1, a_2, and a_3 and slack variable s is:

$Min(100a_1 + 100a_2 + 100a_3)$

Subject to:

$a_1 = 1$

$a_2 = 1$

$a_3 = 1$

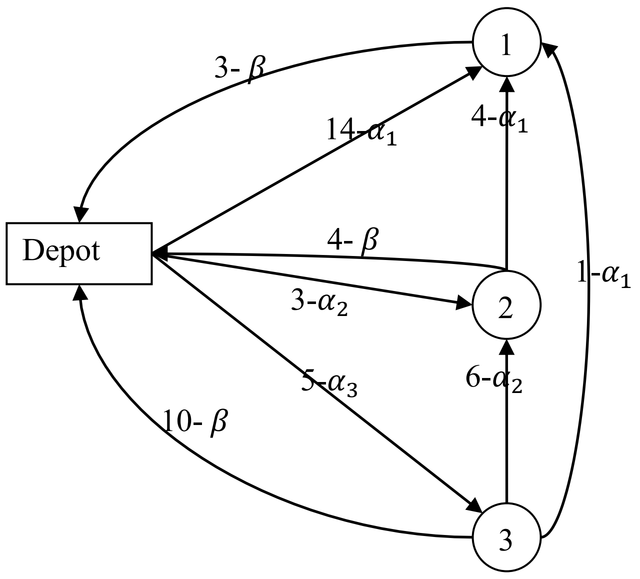
$s = 2$

$a_1, a_2, a_3, s \geq 0$

Solution of RMP

Objective value	Artificial and slack values				Dual values			
	a_1	a_2	a_2	s	α_1	α_2	α_3	β
300	1	1	1	2	100	100	100	0

Reduced costs associated with the arcs are defined with respect to the dual values (see Figure 4-



11).

Figure 4-11: Reduced costs in the network

Let us replace $\alpha_1, \alpha_2, \alpha_3$ with 100 and β with 0. Dynamic programming approach is used to price out the route with the minimum negative reduced cost. The recurrence equation in the dynamic programming approach for the network *N(V,A)* is defined as follows:

$$Q(U,j) = \min_{(i,j)\in A}\{Q(U-\{i\},i) + \bar{c}_{ij}\} \qquad \forall\, j \in V, j \notin U, U \subseteq V$$

$$Q(\{\emptyset\},i) = \bar{c}_{0i} \qquad \forall i \in V$$

$Q(U,j)$ is the minimum reduced cost of a path that starts at the depot and ends at node *j*, while visiting all nodes in the set U only once (node 0 is depot). Values of Q in different states and stages are:

Node	Stage 0	Stage 1	Stage 2	Stage 3
1	$Q(\{\emptyset\},1)$ =-86	$Q(\{2\},1)$ =-193 $Q(\{3\},1)$ =-194	$Q(\{3{,}2\},1)$ =-285	
2	$Q(\{\emptyset\},2)$ =-97	$Q(\{3\},2)$ =-189		
3	$Q(\{\emptyset\},3)$ =-95			
d		$Q(\{1\},d)$ =-83 $Q(\{2\},d)$ =-93 $Q(\{3\},d)$ =-85	$Q(\{2{,}1\},d)$ =-190 $Q(\{3{,}1\},d)$ =-191 $Q(\{3{,}2\},d)$ =-185	$\boldsymbol{Q(\{3,2,1\},d)}$ **=-282**

-282 is minimum negative reduced cost that is related to route (*d*, 3, 2, 1, *d*). Let x_1 be variable associated with this route. x_1 is added to RMP as follows:

$Min(100a_1 + 100a_2 + 100a_3 + 18x_1)$

Subject to:

$a_1 + x_1 = 1$

$a_2 + x_1 = 1$

$a_3 + x_1 = 1$

$s \ + x_1 = 2$

$a_1, a_2, a_3, s, x_1 \geq 0$

Solution of RMP

Objective value	x value	Artificial and slack values				Dual values			
	$x_1.$	a_1	a_2	a_2	s	α_1	α_2	α_3	β
18	1	0	0	0	1	100	100	-182	0

After updating reduced costs in the road network, values of Q are:

Node	Stage 0	Stage 1	Stage 2	Stage 3
1	$Q(\{\emptyset\},1)$ =-86	$Q(\{2\},1)$ =-193 $Q(\{3\},1) = 88$	Q({3,2},1) =-13	
2	$Q(\{\emptyset\},2)$ =-97	$Q(\{3\},2) = 83$		
3	$Q(\{\emptyset\},3) = 187$			
d		$Q(\{1\},d)$ =-83 $Q(\{2\},d)$ =-93 $Q(\{3\},d) = 197$	$\boldsymbol{Q(\{2,1\},d)}$ **=-190** $Q(\{3,1\},d) = 91$ $Q(\{3,2\},d) = 87$	$Q(\{3,2,1\},d)$ =-10

The value of minimum reduced cost is -190 that is related to route (*d, 2, 1, d*). Let x_2 be variable associated with this route. x_2 is added to the RMP as follows:

$Min(18x_1 + 10x_2)$

Subject to:

$x_1 + x_2 = 1$

$x_1 + x_2 = 1$

$x_1 = 1$

$x_1 + x_2 + s = 2$

$s, x_1, x_2 \geq 0$

Solution of RMP

Objective value	x values		Slack value	Dual values			
	$x_1.$	x_2	s	α_1	α_2	α_3	β
18	1	0	1	100	-82	0	0

After updating reduced costs, the route with the minimum negative reduced cost is found as follows:

Node	Stage 0	Stage 1	Stage 2	Stage 3
1	$Q(\{\emptyset\},1)$ =-86	$Q(\{2\},1)$ =-11 $Q(\{3\},1)$ =-94	$Q(\{3,2\},1)$ =-3	
2	$Q(\{\emptyset\},2) = 85$	$Q(\{3\},2) = 93$		
3	$Q(\{\emptyset\},3) = 5$			
d		$Q(\{1\},d)$ =-83 $Q(\{2\},d) = 89$ $Q(\{3\},d) = 15$	$Q(\{2,1\},d)$ =-8 $\boldsymbol{Q(\{3,1\},d)}$ **=-91** $Q(\{3,2\},d) = 97$	$Q(\{3,2,1\},d) = 0$

-91 is minimum reduced cost that is related to route (*d,* 3, 1, *d*). Let x_3 be variable associated with this route. x_3 is added to RMP.

$Min(18x_1 + 10x_2 + 9x_3)$

Subject to:

$x_1 + x_2 + x_3 = 1$

$x_1 + x_2 \qquad = 1$

$x_1 + x_3 \qquad = 1$

$x_1 + x_2 + x_3 + s = 2$

$s, x_1, x_2, x_3 \geq 0$

Objective value	x values			Slack value	Dual values			
	$x_1.$	x_2	x_3	s	α_1	α_2	α_3	β
18	1	0	0	1	1	9	8	0

After updating reduced costs, the values of *Q* are:

Node	Stage 0	Stage 1	Stage 2	Stage 3
1	$Q(\{\emptyset\},1) = 14$	$Q(\{2\},1) = -3$ $Q(\{3\},1) = -3$	$Q(\{3{,}2\},1) = -3$	
2	$Q(\{\emptyset\},2) = -6$	$Q(\{3\},2) = -6$		
3	$Q(\{\emptyset\},3) = -3$			
d		$Q(\{1\},d) = 17$ $\boldsymbol{Q(\{2\},d) = -2}$ $Q(\{3\},d) = 7$	$Q(\{2{,}1\},d) = 0$ $Q(\{3{,}1\},d) = 0$ $\boldsymbol{Q(\{3,2\},d) = -2}$	$Q(\{3{,}2{,}1\},d) = 0$

Since minimum reduced cost (-2) is related to two routes (*d,* 2, *d*) and (*d*, 3, 2, *d*), we select route (*d,* 2, *d*) arbitrary. Let x_4 be variable associated with this route.

$Min(18\mathrm{x}_1 + 10\mathrm{x}_2 + 9\mathrm{x}_3 + 7\mathrm{x}_4)$

Subject to:

$x_1 + x_2 + x_3 = 1$

$x_1 + x_2 + x_4 = 1$

$x_1 + x_3 \qquad = 1$

$x_1 + x_2 + x_3 + x_4 + s = 2$

$s, x_1, x_2, x_3, x_4 \geq 0$

Objective value	x values				Dual values			
	x_1.	x_2	x_3	x_4	α_1	α_2	α_3	β
16	0	0	1	1	3	7	6	0

After updating reduced costs, the values of *Q* are:

Node	Stage 0	Stage 1	Stage 2	Stage 3
1	$Q(\{\emptyset\},1) = 11$	$Q(\{2\},1) = -3$ $Q(\{3\},1) = -3$	$Q(\{3{,}2\},1) = -1$	
2	$Q(\{\emptyset\},2) = -4$	$Q(\{3\},2) = -2$		
3	$Q(\{\emptyset\},3) = -1$			
d		$Q(\{1\},d) = 14$ $Q(\{2\},d) = 0$ $Q(\{3\},d) = 9$	$Q(\{3{,}2\},d) = 2$ $Q(\{3{,}1\},d) = 0$ $Q(\{2{,}1\},d) = 0$	$Q(\{3{,}2{,}1\},d)$ $= 2$

Since there are no routes with the negative marginal cost in the network, the current optimal solution of RMP is optimal to the MP. The value of objective function is 16. One of the vehicles visits customer 3 first and then customer 1 with a total transportation cost equal to 9. Another vehicle visits only customer 2 with the total transportation cost equal to 7.

Chapter 5:
Computational Experiments

In this chapter, we present the computational results for the general 3DVLP, the 3DVLP-AC, and the 3L-CVRP.

The 3DVLP requires the packing of items orthogonally into a minimum number of vehicles. Items have fixed orientation; that is, they cannot be rotated around any of the axes. Meanwhile, the 3DVLP-AC consists of packing items orthogonally with the fixed vertical orientation into a minimum number of vehicles while satisfying allocation and capacity constraints. Finally, the 3L-CVRP consists of finding feasible routes with minimum total traveling cost while satisfying customers' demands expressed in terms of cuboid and weighted items (loading constraints).

All algorithms applied in this work are implemented in visual C++ and run on an Intel Core Duo 2.2 GHz CPU. The ILOG CPLEX 12.2.0 solver is used to solve the linear mathematical programming models.

5.1 3DVLP

5.1.1 Problem Tests

The experiments for the 3DVLP are performed using eight classes of random instances generated by Martello et al. [23]. For classes 1 to 5, the bin size is $W = H = L = 100$ and the dimensions of the items adhere to one of the following five types:

- Type 1: w_i uniformly random in $[1, \frac{1}{2}W]$, h_i uniformly random in $[\frac{2}{3}H, H]$, l_i uniformly random in $[\frac{2}{3}L, L]$;
- Type 2: w_i uniformly random in $[\frac{2}{3}W, W]$, h_i uniformly random in $[1, \frac{1}{2}H]$, l_i uniformly random in $[\frac{2}{3}L, L]$;
- Type 3: w_i uniformly random in $[\frac{2}{3}W, W]$, h_i uniformly random in $[\frac{2}{3}H, H]$, l_i uniformly random in $[1, \frac{1}{2}L]$;
- Type 4: w_i uniformly random in $[\frac{1}{2}W, W]$, h_i uniformly random in $[\frac{1}{2}H, H]$, l_i uniformly random in $[\frac{1}{2}L, L]$;
- Type 5: w_i uniformly random in $[1, \frac{1}{2}W]$, h_i uniformly random in $[1, \frac{1}{2}H]$, l_i uniformly random in $[1, \frac{1}{2}L]$;
- Class 1: type 1 with probability 60%, each of types 2, 3, 4 and 5 with probability 10%;
- Class 2: type 2 with probability 60%, each of types 1, 3,4 and 5 with probability 10%;
- Class 3: type 3 with probability 60%, each of types 1, 2, 4 and 5 with probability 10%;
- Class 4: type 4 with probability 60%, each of types 1, 2, 3 and 5 with probability 10%; and

- Class 5: type 5 with probability 60%, each of types 1, 2, 3 and 4 with probability 10%.

Classes 6 to 8, produced according to instances presented in Berkey and Wang [99], are:

- Class 6: w_i , h_i and l_i uniformly random in [1,10] and $W = H = L = 10$;
- Class 7: w_i , h_i and l_i uniformly random in [1,35] and $W = H = L = 40$; and
- Class 8: w_i , h_i and l_i uniformly random in [1,100] and $W = H = L = 100$.

Such instances can be reproduced by the instance generator available at Pisinger's website (www.diku.dk/~pisinger/codes.html).

5.1.2 Mathematical Formulation Results

The general LP model of 3DVLP is solved using CPLEX for different instances with a number of items (*n*) between 10 and 45. Ten different problem instances based on different random seeds are generated for each class and number of items.

Table 5-1 gives the number of instances, out of 10, that are solved to optimality using CPLEX within the time limit of 3,600 seconds for each class and number of items. The average computation times in which CPLEX can solve instances to optimality are indicated in the Time column for each class and number of items. The total number of instances solved to optimality in each class is reported in the last row. It can be seen from Table 5-1 that, for up to 30 items most class instances are solved to optimality by CPLEX. Since the total number of solved instances in classes 7 and 3 is less than the other classes, these are the most difficult classes solved by CPLEX. In contrast, class 4 represents the easiest one, with 80 solved instances in total.

Table 5-1: CPLEX result for 3DVLP

n	Class1		Class2		Class3		Class4		Class5		Class6		Class7		Class8	
n	Number of instances	Time	Number of instances	Time	Number of instances	Time	Number of instances	Time	Number of instances	Time	Number of instances	Time	Number of instances	Time	Number of instances	Time
10	10	0.95	10	0.90	10	0.70	10	0.85	10	1.02	10	0.86	10	0.99	10	1.00
15	10	2.13	10	1.80	10	1.90	10	1.75	10	2.16	10	1.90	10	2.08	10	2.15
20	10	6.50	10	3.60	10	3.80	10	3.25	10	3.95	10	3.85	10	3.87	10	3.72
25	10	13.81	9	21.33	10	13.88	10	4.46	10	9.74	10	10.78	10	58.35	10	17.14
30	10	14.18	10	31	9	10.88	10	7.9	10	15.57	9	17.3	10	36	9	18.77
35	8	161	7	17	8	294	10	6.664	10	30	9	277	8	76	9	50
40	3	108	2	121.7	6	191	10	9.6	8	75	8	257	4	50	6	91.66
45	5	263	7	1774	0	-	10	11.87	6	85	8	300	1	30	6	333
Total Num	66		65		63		80		74		74		63		70	

5.1.3 CG Results

The CG algorithm for the 3DVLP is tested on a set of 320 instances. For each class and number of items (*n*) between 50 and 200, ten different problem instances based on different random seeds are generated.

The CG technique is initialized by assigning one item to one vehicle. Since the solutions to the RMP for all instances are non-integers after stopping CG, we apply branching to obtain integer solutions. D_FH branching with a limit value of 0.6 leads to good integer solutions for most of instances, but leads to infeasible solutions for a maximum of 4 problem instances out of 10 for each class and instance size. For the instances with an infeasible solution, all fixed variables in the branching are released and the current RMP is solved to integrality by the CPLEX. The total execution time, including CG time, branching, and solver time for each instance is less than 1000 seconds. The branching rules based on Ryan and Foster [98] are not only time-consuming for large instances, but also, the solutions are not superior to those generated by D_FH and CPLEX for any instance size. Therefore, we present only the results produced by D_FH branching and CPLEX.

We compare the CG technique with TS^{2}PACK developed by Crainic et al. [29] and the GRASP algorithm from Parreño et al. [31], as these two methods have given the best results for the 3DVLP in the literature. The computational results are summarized in Table 5-2. The class of an instance, vehicle size, and total number of items are presented in the first three columns respectively. The CG, TS^{2}PACK, and GRASP columns correspond to the mean number of vehicles over 10 instances generated using the CG technique, the TS^{2}PACK heuristic, and the

GRASP method. Note that TS^{2}PACK stops after 1,000 seconds for each instance. No TS^{2}PACK results for the classes 2 and 3 were reported in Crainic et al. [29], because the properties of such classes are the same as those of class 1. Moreover, the LB column shows the lower bound for the mean number of vehicles over 10 instances reported by Boschetti [17]. The last column indicates the average execution time of CG in seconds for each instance.

The best CG technique solutions, in comparison with both methods of GRASP and TS^{2}PACK, are highlighted in bold in Table 5-2. This comparison shows that the CG technique gives equal or better results compared to GRASP for all instances except for one sub-instance in class 6. Total gap between CG and TS^{2}PACK is less than 1 %.

Table 5-2: CG technique, TS2PACK and GRASP results for the 3DVLP

Class	Vehicle size	Number of items	CG	TS^{2}Pack	GRASP	LB	CPU time
1	100×100	50	**13.4**	13.4	13.4	12.9	15
		100	**26.6**	26.7	26.6	25.6	100
		150	**36.3**	37.0	36.4	35.8	500
		200	**50.7**	51.1	50.9	49.7	700
2	100×100	50	**13.8**	-	13.8	13	10
		100	**25.5**	-	25.7	25.1	70
		150	**36.4**	-	36.9	35.7	500
		200	**49.3**	-	49.4	48.1	800
3	100×100	50	**13.3**	-	13.3	12.7	20
		100	**25.9**	-	26.0	24.7	60
		150	**37.4**	-	37.6	36.4	500
		200	**49.7**	-	50.0	48.6	800
4	100×100	50	**29.4**	29.4	29.4	29.0	4
		100	59.0	58.9	59.0	58.5	11
		150	**86.8**	86.8	86.8	86.4	30
		200	**118.8**	118.8	118.8	118.3	60
5	100×100	50	**8.3**	8.3	8.3	7.6	11
		100	**15.0**	15.2	15.0	14.0	50
		150	**20.0**	20.1	20.1	18.8	500
		200	**27.1**	27.4	27.1	26.0	500
6	10×10	50	9.9	9.8	9.8	9.4	10
		100	**19.0**	19.1	19.0	18.4	150
		150	**29.2**	29.2	29.2	28.5	300
		200	**37.4**	37.7	37.4	36.7	1000
7	40×40	50	**7.4**	7.4	7.4	6.8	10
		100	12.4	12.3	12.5	11.5	100
		150	**15.5**	15.8	16.0	14.4	1000
		200	**23.5**	23.5	23.5	22.7	1000
8	100×100	50	**9.2**	9.2	9.2	8.7	12
		100	18.9	18.8	18.9	18.4	120
		150	**23.5**	23.9	24.1	22.5	1000
		200	**29.4**	30.0	29.8	28.2	1000
Total Vehicles		Classes1,4-8	**726.78**	729.8	728.6	708.8	
		All Classes	**978.08**		981.3		

5.2 3DVLP-AC

5.2.1 Problem Tests

There are no specified problem instances in the literature for 3DVLP-AC. In fact, this problem has not been considered on its own in previous research. We use data and instance generators introduced by Gendreau et al. [51], creating a set of 200 problem instances. For all instances, the length L, width W, and height H of the vehicle space are set to 60, 25, and 30. Considering a given vehicle capacity and a number of customers with specified capacities, we generate 10 different problem instances based on different items assigned to the customers. For each customer k, the number of requested items is stochastically generated within the specified interval [1,3]. The dimensions l_i , w_i, and h_i for each item i are integer values from intervals [0.2L,0.6L], [0.2W,0.6W], and [0.2H,0.6H], respectively.

5.2.2 TS and CPLEX Results

The TS heuristic algorithm is tested on a set of 200 instances. The results are summarized in Table 5-3. The weight capacity of the vehicle, number of customers and number of items are presented in the first three columns. The n column gives the mean number of vehicles over 10 instances, resulting from the TS method. The average computation times in seconds are presented in the *Sec* column for TS.

Unfortunately, there are no lower bounds (LB) for the 3DVLP-AC reported in the literature with which to evaluate the TS results. We compare such results with the solutions obtained from CPLEX for the small-sized instances. The ratio $\frac{(UB\_LB)}{LB} \times 100$ over instances is given in Gap column for only those instances that can be solved by CPLEX. Upper bound (UB) represents the average solution obtained by the TS method. LB is the average optimal solution given by CPLEX. It should be noted that optimal solutions resulted from CPLEX are exactly same as TS solutions for the small-sized instances.

The *N of optimal* column gives the number of instances out of 10 that are solved to optimality by CPLEX. The “-” symbol in the table indicates that CPLEX can give no optimal solutions. In general, CPLEX can give an optimal solution, a feasible solution, or no solutions, which occurred when it stops because the time limit is exceeded or an out of memory error occurs. Moreover, it can usually solve instances with up to 40 customers in a reasonable amount of time as can be seen from Table 5-3. The average computation times in seconds are presented in *Sec* column for the CPLEX.

Table 5-3: TS and CPLEX results for 3DVLP-AC

	Capacity	Customers	Items	TS		*Gap %*	CPLEX	
				n	*Sec*		*N of Optimal*	*Sec*
1	90	15	29	3.3	0.89	0.00	10	170.00
2	55	15	30	5.0	0.05	0.00	10	3.38
3	85	20	40	4.4	0.64	0.00	7	285.44
4	58	20	40	6.0	0.06	0.00	10	20.32
5	4000	21	42	6.0	0.13	0.00	10	31.64
6	4500	22	44	4.7	1.34	-	-	
7	48	25	50	8.0	0.42	0.00	10	58.15
8	4500	29	60	6.8	53.43	-	-	-
9	68	30	60	9.0	0.60	0.00	8	394.00
10	67	35	61	11.0	0.35	0.00	9	500.00
11	60	40	69	14.0	0.70	0.00	9	1000.0
12	2010	44	89	8.2	185.00	-	-	-
13	160	50	100	9.0	777.00	-	-	-
14	30000	71	144	12.5	500.00	-	-	-
15	180	75	150	13.0	1000.00	-	-	-
16	140	75	150	13.0	1000.00	-	-	
17	100	75	150	14.3	504.00	-	-	-
18	200	100	200	16.0	1000.00	-	-	
19	200	100	200	19.0	1000.00	-	-	-
20	112	100	200	17.0	1000.00	-	-	-

5.3 3L-CVRP

5.3.1 Problem Tests

The CG algorithm for 3L-CVRP is tested on a set of 27 instances introduced by Gendreau et al. [51]. The graph, the weight capacity demanded by the customers, and the vehicle weight capacity were derived from 27 Euclidean CVRP instances (see Toth and Vigo [54]). The length L, width W, and height H of the vehicles (loading spaces) are set to 60, 25, and 30, respectively. For each customer i, the number of requested items (m_i) is stochastically generated within the interval [1,3]. For each item I_{ik},($k = 1, \dots, m_i$), dimensions l_{ik} , w_{ik}, and h_{ik} are integer values within intervals [$0.2L$, $0.6L$], [$0.2W$, $0.6W$], and [$0.2H$, $0.6H$], respectively.

5.3.2 CG Results

The pricing problem in the CG technique is solved using two approaches, as follows: 1) the heuristic pricing method (HP), and 2) by solving an integrated problem of ESPPRC and the loading problem denoted as ESPPRC-L. The results of the CG technique using HP and ESPPRC-L are summarized in Table 5-4. The instance name, total number of customers, total number of items, and number of vehicles are presented in the first four columns. For each pricing method, the total routing cost (z) resulted from CG and the execution time (sec) in seconds, are reported. The results show that the performance of the CG technique with HP is better than with ESPPRC-L in terms of solution quality and execution time for all instances except E016-05m, E023-03g, and E026-08m.

It should be noted that ESPPRC stops when 200 labels with a negative reduced cost are extended to the depot node at each iteration of the CG technique. If no further columns with negative reduced costs can be found by the HP or there are no improvements in the RMP solution, then the CG stops.

Since the solutions to the RMP for all instances are non-integers, we applied branching rule D_FH to obtain an integer solution. D_FH with the limit value 0.9 leads to satisfactory integer solutions for most instances, but infeasible solutions for some others. For the instances with the infeasible solutions, all fixed variables in the branching are released and the RMP is solved to integrality by means of CPLEX.

Table 5-4: Results for the CG technique with HP and ESPPRC-L

Instances				HP		ESPPRC-L	
	n	m	v	z	sec_z	z	sec_z
E016-03m	15	32	5	315.16	19.73	315.16	415.85
E016-05m	15	26	5	345.28	5.69	341.93	10.63
E021-06m	20	37	5	391.55	36.48	393.47	672.38
E021-06m	20	36	6	430.78	12.44	445.11	201.83
E022-04g	21	45	7	447.56	38.17	447.56	856.40
E022-06m	21	40	6	498.97	18.65	500.93	181.19
E023-03g	22	46	6	793.40	47.50	789.8	585.09
E023-05s	22	43	8	807.01	42.79	848.52	605.51
E026-08m	25	50	8	653.73	35.24	647.33	525.89
E030-03g	29	62	10	883.57	128.18	883.57	930.95
E030-04s	29	58	9	820.19	156.18	823.92	545.57
E031-09h	30	63	9	614.42	70.07	614.42	819.63
E033-03n	32	61	9	2735.18	185.79	2789.19	936.86
E033-04g	32	72	11	1504.25	295.60	1504.25	752.75
E033-05s	32	68	10	1412.89	330.06	1412.89	749.33
E036-11h	35	63	11	698.42	98.12	698.42	527.54

E041-14h	40	79	14	866.21	84.05	866.21	331.21
E045-04f	44	94	14	1275.85	606.64	1344.08	1980.04
E051-05e	50	99	13	799.37	1280.08	846.96	1956.77
E072-04f	71	147	20	632.43	1464.27	642.21	1723.30
E076-07s	75	155	18	1145.11	1700.46	1295.93	3244.25
E076-08s	75	146	19	1237.53	1180.41	1299.86	2758.94
E076-10e	75	150	18	1246.86	1353.24	1246.86	1101.85
E076-14s	75	143	18	1244.84	1089.99	1264.66	3814.46
E101-08e	100	193	24	1456.95	2435.53	1585.58	2299.33
E101-10c	100	199	28	1711.10	2815.44	1870.89	2076.56
E101-14s	100	198	25	1642.56	3553.39	1700.62	3318.47
Average				985.598	706.825	1015.56	1256.39

The performance of the CG technique using HP is compared with the TS algorithm from Gendreau et al. [51] and the guided tabu search (GTS) from Tarantilis et al. [74] in Table 5-5. The differences between total routing costs of CG and TS as $100(z_{CG}-z_{TS})/z_{TS}$ and for CG and GTS as $100(z_{CG}-z_{GTS})/z_{GTS}$ are represented in the *Gap* column. The overall results indicate that the CG technique outperforms TS and GTS in terms of quality and execution time. CG improves the average solution from TS by 5.04% and GTS by 1.5%. The quality of the solutions obtained using the CG technique is better than that of TS for all instances. The best CG technique solutions, in comparison with both the TS and GTS methods, are highlighted in bold in Table 5-5.

Table 5-5: Results for the CG, TS and GTS

Instances	*CG*		*TS*		*GTS*		*Gap %*	
	z_{CG}	Sec_z	z_{TS}	Sec_z	z_{GTS}	Sec_z	*CG-TS*	*CG-GTS*
E016-03m	**315.16**	19.73	316.32	129.5	321.47	7.8	-0.37	-1.96
E016-05m	345.28	5.69	350.58	5.3	334.96	7.2	-1.51	3.08
E021-06m	**391.55**	36.48	447.73	461.1	430.95	352.6	-12.55	-9.14
E021-06m	**430.78**	12.44	448.48	181.1	458.04	204.0	-3.95	-5.95
E022-04g	**447.56**	38.17	464.24	75.8	465.04	61.3	-3.59	-3.76
E022-06m	**498.97**	18.65	504.46	1167.9	507.96	768.8	-1.09	-1.77
E023-03g	**793.40**	47.50	831.66	181.1	796.61	241.5	-4.60	-0.40
E023-05s	**807.01**	42.79	871.77	156.1	880.93	140.0	-7.43	-8.39
E026-08m	653.73	35.24	666.10	1468.5	642.22	604.7	-1.86	1.79
E030-03g	**883.57**	128.18	911.16	714.0	884.74	803.1	-3.03	-0.13
E030-04s	820.19	156.18	819.36	396.4	873.43	308.5	0.10	-6.10
E031-09h	**614.42**	70.07	651.58	268.1	624.24	180.8	-5.70	-1.57
E033-03n	**2735.18**	185.79	2928.34	1639.1	2799.74	1309.5	-6.60	-2.31
E033-04g	**1504.25**	295.60	1559.64	3451.6	1504.44	2678.1	-3.55	-0.01
E033-05s	**1412.89**	330.06	1452.34	2327.4	1415.42	1466.3	-2.72	-0.18
E036-11h	**698.42**	98.12	707.85	2550.3	698.61	2803.2	-1.33	-0.03
E041-14h	**866.21**	84.05	920.87	2142.5	872.79	1208.6	-5.94	-0.75
E045-04f	**1275.85**	606.64	1400.52	1452.9	1296.59	1300.9	-8.90	-1.60
E051-05e	**799.37**	1280.08	871.29	1822.3	818.68	1438.4	-8.25	-2.36
E072-04f	**632.43**	1464.27	732.12	790.0	641.57	1284.8	-13.62	-1.42
E076-07s	**1145.11**	1700.46	1275.20	2370.3	1159.72	1704.8	-10.20	-1.26
E076-08s	**1237.53**	1180.41	1277.94	1611.3	1245.35	1663.5	-3.16	-0.63
E076-10e	1246.86	1353.24	1258.16	6725.6	1231.92	3048.2	-0.90	1.21
E076-14s	1244.84	1089.99	1307.09	6619.3	1201.96	2876.8	-4.76	3.57
E101-08e	1456.95	2435.53	1570.72	5630.9	1457.46	3432.0	-7.24	-0.03
E101-10c	**1711.10**	2815.44	1847.95	4123.7	1711.93	3974.8	-7.41	-0.05
E101-14s	**1642.56**	3553.39	1747.52	7127.2	1646.44	5864.2	-6.01	-0.24
Average	985.598	706.82	1042.26	2058.9	997.156	1471.0	-5.04	-1.50

5.3.3 Sensitivity to the Loading Constraints in 3L-CVRP

To analyse the effect of different loading constraints on the results, we conduct five sets of experiments. In the first set, all the loading constraints are considered and the model is executed. In set 2, fragility constraints are eliminated. Note that the fragility constraints demand that non-fragile items not be stacked on the surfaces of fragile items. Each item can be stacked on non-fragile items. In the third set, we eliminate the LIFO constraints. Under LIFO, all items should be directly unloaded through a sequence of straight movements parallel to the length of vehicle without repositioning the other items. Thus, no item to be delivered later may be placed over the first item for the delivery or between this item and the rear of the vehicle. In the fourth set, the stability constraints are eliminated. According to these constraints, each item not packed on the vehicle floor needs to have enough supporting surface. The supporting surface of an item should be at least 75 percent of base area of the item. In the fifth set, only the 3D loading is considered without any of the aforementioned constraints.

Table 5-6 gives the total routing costs resulted from CG with HP for different experiments. The first column gives the total routing cost solution (z) for all constraints. The solutions with no fragility constraints, no LIFO constraints and no stability constraints ($z1$ to $z3$) are reported in columns 1 to 3, respectively. The total routing cost solution ($z4$) reported in the last column is obtained without considering any constraints. In the last row (*Gap*), the differences between average solutions for all constraints and each of the loading constraints as $100(z-zi)/z$ are reported. It can be seen from Table 5-6 that, the average solution values are reduced by 0.9%, 2.38% and 2.77% when the fragility constraints, the LIFO constraint, and the stability constraint, respectively, are removed. In removing all three constraints, the average solution value is reduced by 9.49% which is greater than the reduction that resulted by separately removing each of three constraints.

Table 5-6: CG solutions for various loading constraints configuration

Instances	All Constraints	No Fragility	No LIFO	No Stability	Just 3D Loading
	z	z_1	z_2	z_3	z_4
E016-03m	315.16	305.71	305.71	301.95	290.92
E016-05m	345.28	334.89	334.89	334.89	334.89
E021-06m	391.55	378.32	364.19	378.32	362.18
E021-06m	430.78	430.78	430.78	430.78	430.78
E022-04g	447.56	445.4	440.7	438.22	396.08
E022-06m	498.97	498.05	498.21	495.71	495.71
E023-03g	793.40	789.65	742.14	752.65	739.83
E023-05s	807.01	807.01	801.36	802.77	735.03
E026-08m	653.73	632.57	629.99	629.99	628.08
E030-03g	883.57	858.53	863.9	838.48	756.02
E030-04s	820.19	816.26	809.31	783.69	760.07
E031-09h	614.42	614.42	610.06	610.06	610.06
E033-03n	2735.18	2735.09	2720.56	2720.11	2378.53
E033-04g	1504.25	1498.61	1471.73	1457.32	1318.86
E033-05s	1412.89	1397.5	1390.13	1389.64	1361.15
E036-11h	698.42	698.42	698.42	698.42	698.42
E041-14h	866.21	866.21	866.21	863.06	861.57
E045-04f	1275.85	1256.79	1261.14	1245.85	1149.08
E051-05e	799.37	784.67	773.65	757.09	694.38
E072-04f	632.43	626.43	607.61	609.27	544.31
E076-07s	1145.11	1130.91	1125.8	1124.16	1033.31
E076-08s	1237.53	1223.12	1187.47	1208.84	1108.33
E076-10e	1246.86	1225.84	1212.53	1175.8	1034.4
E076-14s	1244.84	1197.52	1151.41	1184.82	1109.59
E101-08e	1456.95	1453.35	1432.09	1441.23	1337.55
E101-10c	1711.10	1705.06	1690.7	1642.17	1496.75

E101-14s	1642.56	1638.04	1558.05	1558.67	1418.53
Average	985.59	975.89	962.17	958.29	892.01
Gap %		-0.9	-2.38	-2.77	-9.49

Chapter 6:
Conclusions and Future Works

This thesis addresses the three-dimensional vehicle routing problem with loading constraints (3L-CVRP). The 3L-CVRP consists of finding feasible routes with the minimum total travel cost while satisfying customers' demands, expressed in terms of cuboid and weighted items. We address the problem in two stages. In the first stage, we address the general three-dimensional vehicle loading problem (3DVLP) and in the second stage, the 3L-CVRP.

The 3DVLP deals with the way of loading cuboid items (e.g., boxes) in one or more vehicles such that empty space of vehicles is minimized for the purpose of minimizing vehicle movements. Preliminary work has been done on modeling 3DVLP. In fact the mathematical models are very useful for clarifying problems and evaluating the quality of solutions that have resulted from heuristic algorithms. The mathematical model of the 3DVLP problem can be solved by CPLEX in a reasonable period of time for small instances of up to 30 items. To deal with larger instances, we employ a set-partitioning formulation of the problem based on the well-known Dantzig-Wolfe decomposition. This formulation is solved by the CG-based heuristic method. Our extensive computational results indicate that the CG technique outperforms other techniques proposed in the literature.

The 3DVLP with allocation and capacity constraints, called 3DVLP-AC, is also considered. For the 3DVLP-AC, CPLEX could handle moderate-sized instances with up to 40 customers. To deal with large-sized instances, a TS heuristic algorithm is developed. Unfortunately, there are no solution methods or lower bounds (LBs) for the 3DVLP-AC existent in the literature by which to evaluate the TS results. Therefore, we evaluate our TS with the CPLEX results for small instances.

For the 3L-CVRP, we propose an LP model based on the classical two-index flow model. The 3L-CVRP is represented as a set-partitioning formulation based on the well-known Dantzig-Wolfe decomposition. This set-partitioning formulation is solved by the CG technique embedded in the branch-and-bound (B&P) method. To generate new columns, an integrated approach using the shortest path problem and the 3D loading problem is applied. To speed up the CG technique, fast CG is also carried out by applying a heuristic pricing method. The CG technique with the heuristic pricing outperforms the efficient tabu search technique proposed in the literature in terms of solution quality and execution time.

Future Works

The heuristic methods for 3DVLP typically consider the following two phases simultaneously: 1) assigning items to the vehicles, and 2)-optimizing the approach to positioning items within the vehicles. Since this second phase has a considerable effect on the performance of a solution method, future research should concentrate on the following:

- Improving methods of positioning items within a single vehicle; and

- Considering the effect of each of the aforementioned practical constraints on how items are packed.

There are no benchmark instances and LBs for the 3DVLP with allocation and capacity constraints in the literature that can be used to evaluate and test a solution to this problem. Efficient LBs should be developed in future studies to improve an approach aimed to solving this problem. Another interested topic for future works would be to assess the effects of the following constraints on the routing aspect of the 3L-CVRP:

- The use of different types of vehicles;
- Investigation of the time windows associated with customers; and
- Consideration of the dynamic elements of travel time and customer demands.

The performance of the CG technique for the 3L-CVRP strongly depends on three elements: 1) the pricing problem, 2) the heuristic method of checking feasibility in terms of loading constraints, and 3) branching rules. It is suggested that future work should include the modification and development of efficient methods to deal with each of these elements.

Bibliography

[1] A. Benjelloun, T. G. Crainic, and Y. Bigras, “Towards a taxonomy of City Logistics projects,” *Procedia-Social and Behavioral Sciences*, vol. 2, no. 3, pp. 6217–6228, 2010.

[2] E. Taniguchi and R. G. Thompson, “Modeling city logistics,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1790, no. 1, pp. 45–51, 2002.

[3] T. G. Crainic, N. Ricciardi, and G. Storchi, “Models for evaluating and planning city logistics systems,” *Transportation science*, vol. 43, no. 4, pp. 432–454, 2009.

[4] S. Anderson, J. Allen, and M. Browne, “Urban logistics—-how can it meet policy makers’ sustainability objectives?,” *Journal of Transport Geography*, vol. 13, no. 1, pp. 71–81, 2005.

[5] A. Bortfeldt and G. Wäscher, *Container loading problems: A state-of-the-art review*. Univ., Faculty of Economics and Management, 2012.

[6] H. Dyckhoff, “A typology of cutting and packing problems,” *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.

[7] G. Wäscher, H. Haußner, and H. Schumann, “An improved typology of cutting and packing problems,” *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, 2007.

[8] D. Pisinger, “Heuristics for the container loading problem,” *European Journal of Operational Research*, vol. 141, no. 2, pp. 382–392, 2002.

[9] A. Scholl, R. Klein, and C. Jürgens, “BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem,” *Computers & Operations Research*, vol. 24, no. 7, pp. 627–645, 1997.

[10] P. Schwerin and G. Wäscher, “The Bin-Packing Problem: A Problem Generator and Some Numerical Experiments with FFD Packing and MTP,” *International Transactions in Operational Research*, vol. 4, no. 5–6, pp. 377–389, 1997.

[11] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley &amp; Sons, Inc., 1990.

[12] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

[13] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem-Part II," *Operations research*, vol. 11, no. 6, pp. 863–888, 1963.

[14] S. P. Fekete and J. Schepers, "On more-dimensional packing I: Modeling," Mathematisches Institut, Universit€at zu Koln, Technical paper ZPR97-288, 1997.

[15] D. Pisinger and M. Sigurd, "Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem," *INFORMS Journal on Computing*, vol. 19, no. 1, pp. 36–51, 2007.

[16] S. Martello and D. Vigo, "Exact solution of the two-dimensional finite bin packing problem," *Management science*, vol. 44, no. 3, pp. 388–399, 1998.

[17] M. A. Boschetti, "New lower bounds for the three-dimensional finite bin packing problem," *Discrete Applied Mathematics*, vol. 140, no. 1–3, pp. 241–258, 2004.

[18] A. Caprara and M. Monaci, "Bidimensional packing by bilinear programming," *Math. Program.*, vol. 118, no. 1, pp. 75–108, 2009.

[19] J. Coffman, M. R. Garey, D. S. Johnson, and R. E. Tarjan, "Performance bounds for level-oriented two-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 808–826, 1980.

[20] A. Lodi, S. Martello, and D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems," *INFORMS Journal on Computing*, vol. 11, no. 4, pp. 345–357, 1999.

[21] J. Puchinger and G. R. Raidl, "Models and algorithms for three-stage two-dimensional bin packing," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1304–1327, 2007.

[22] C. S. Chen, S. M. Lee, and Q. S. Shen, "An analytical model for the container loading problem," *European Journal of Operational Research*, vol. 80, no. 1, pp. 68–76, 1995.

[23] S. Martello, D. Pisinger, and D. Vigo, "The three-dimensional bin packing problem," *Operations Research*, vol. 48, no. 2, pp. 256–267, 2000.

[24] E. denBoef, J. Korst, S. Martello, D. Pisinger, and D. Vigo, “Erratum to ‘The Three-Dimensional Bin Packing Problem’: Robot-Packable and Orthogonal Variants of Packing Problems,” *Oper. Res.*, vol. 53, no. 4, pp. 735–736, 2005.

[25] S. Martello, D. Pisinger, D. Vigo, E. D. Boef, and J. Korst, “Algorithm 864: General and Robot-packable Variants of the Three-dimensional Bin Packing Problem,” *ACM Trans. Math. Softw.*, vol. 33, no. 1, 2007.

[26] O. Faroe, D. Pisinger, and M. Zachariasen, “Guided local search for the three-dimensional bin-packing problem,” *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 267–283, 2003.

[27] A. Lodi, S. Martello, and D. Vigo, “Heuristic algorithms for the three-dimensional bin packing problem,” *European Journal of Operational Research*, vol. 141, no. 2, pp. 410–420, 2002.

[28] T. G. Crainic, G. Perboli, and R. Tadei, “Extreme point-based heuristics for three-dimensional bin packing,” *Informs Journal on computing*, vol. 20, no. 3, pp. 368–384, 2008.

[29] T. G. Crainic, G. Perboli, and R. Tadei, “TS2PACK: A two-level tabu search for the three-dimensional bin packing problem,” *European Journal of Operational Research*, vol. 195, no. 3, pp. 744–760, 2009.

[30] S. P. Fekete and J. Schepers, “A new exact algorithm for general orthogonal d-dimensional knapsack problems,” in *Algorithms—ESA’97*, 1997, pp. 144–156.

[31] F. Parreño, R. Alvarez-Valdés, J. F. Oliveira, and J. M. Tamarit, “A hybrid GRASP/VND algorithm for two-and three-dimensional bin packing,” *Annals of Operations Research*, vol. 179, no. 1, pp. 203–220, 2010.

[32] B. Mahvash, A. Awasthi, and S. Chauhan, “Column Generation Based Heuristic for the Three Dimensional Vehicle Loading Problem,” in *Computational Logistics*, D. Pacino, S. Voß, and R. M. Jensen, Eds. Springer Berlin Heidelberg, 2013, pp. 257–268.

[33] S. Martello, M. Monaci, and D. Vigo, “An exact approach to the strip-packing problem,” *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 310–319, 2003.

[34] B. S. Baker, J. Coffman, and R. L. Rivest, “Orthogonal packings in two dimensions,” *SIAM Journal on Computing*, vol. 9, no. 4, pp. 846–855, 1980.

[35] E. K. Burke, G. Kendall, and G. Whitwell, “A new placement heuristic for the orthogonal stock-cutting problem,” *Operations Research*, vol. 52, no. 4, pp. 655–671, 2004.

[36] K. A. Dowsland, “Some experiments with simulated annealing techniques for packing problems,” *European Journal of Operational Research*, vol. 68, no. 3, pp. 389–399, 1993.

[37] S. Jakobs, “On genetic algorithms for the packing of polygons,” *European Journal of Operational Research*, vol. 88, no. 1, pp. 165–181, 1996.

[38] M. C. Riff, X. Bonnaire, and B. Neveu, “A revision of recent approaches for two-dimensional strip-packing problems,” *Engineering Applications of Artificial Intelligence*, vol. 22, no. 4, pp. 823–827, 2009.

[39] A. Bortfeldt and D. Mack, “A heuristic for the three-dimensional strip packing problem,” *European Journal of Operational Research*, vol. 183, no. 3, pp. 1267–1279, 2007.

[40] Y. Wu, W. Li, M. Goh, and R. de Souza, “Three-dimensional bin packing problem with variable bin height,” *European Journal of Operational Research*, vol. 202, no. 2, pp. 347–355, 2010.

[41] S. D. Allen, E. K. Burke, and G. Kendall, “A hybrid placement strategy for the three-dimensional strip packing problem,” *European Journal of Operational Research*, vol. 209, no. 3, pp. 219–227, 2011.

[42] J. Egeblad and D. Pisinger, “Heuristic approaches for the two-and three-dimensional knapsack packing problem,” *Computers & Operations Research*, vol. 36, no. 4, pp. 1026–1049, 2009.

[43] H. Gehring and A. Bortfeldt, “A Genetic Algorithm for Solving the Container Loading Problem,” *International Transactions in Operational Research*, vol. 4, no. 5–6, pp. 401–418, 1997.

[44] A. Moura and J. F. Oliveira, “A GRASP approach to the container-loading problem,” *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 50–57, 2005.

[45] F. Parreño, R. Alvarez-Valdes, J. F. Oliveira, and J. M. Tamarit, “Neighborhood structures for the container loading problem: a VNS implementation,” *J Heuristics*, vol. 16, no. 1, pp. 1–22, 2010.

[46] T. Fanslau and A. Bortfeldt, “A Tree Search Algorithm for Solving the Container Loading Problem,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 222–235, 2009.

[47] J. F. Gonçalves and M. G. Resende, “A parallel multi-population biased random-key genetic algorithm for a container loading problem,” *Computers & Operations Research*, vol. 39, no. 2, pp. 179–190, 2012.

[48] K. Mathur, “An integer-programming-based heuristic for the balanced loading problem,” *Operations Research Letters*, vol. 22, no. 1, pp. 19–25, 1998.

[49] A. Lodi, S. Martello, and D. Vigo, “Approximation algorithms for the oriented two-dimensional bin packing problem,” *European Journal of Operational Research*, vol. 112, no. 1, pp. 158–166, 1999.

[50] M. Eley, “A bottleneck assignment approach to the multiple container loading problem,” *OR Spectrum*, vol. 25, no. 1, pp. 45–60, 2003.

[51] M. Gendreau, M. Iori, G. Laporte, and S. Martello, “A Tabu Search Algorithm for a Routing and Container Loading Problem,” *Transportation Science*, vol. 40, no. 3, pp. 342–350, 2006.

[52] L. Junqueira, R. Morabito, and D. Sato Yamashita, “Three-dimensional container loading models with cargo stability and load bearing constraints,” *Computers & Operations Research*, vol. 39, no. 1, pp. 74–85, 2012.

[53] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[54] P. Toth and D. Vigo, *The vehicle routing problem*. Philadelphia: Society for Industrial and Applied Mathematics, 2002.

[55] S. N. Parragh, K. F. Doerner, and R. F. Hartl, “A survey on pickup and delivery problems,” *Journal für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, 2008.

[56] J. F. Bard, G. Kontoravdis, and G. Yu, “A branch-and-cut procedure for the vehicle routing problem with time windows,” *Transportation Science*, vol. 36, no. 2, pp. 250–269, 2002.

[57] G. Gutiérrez-Jarpa, G. Desaulniers, G. Laporte, and V. Marianov, “A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows,” *European Journal of Operational Research*, vol. 206, no. 2, pp. 341–349, 2010.

[58] S. Ropke, J. F. Cordeau, M. Iori, and D. Vigo, “Branch-and-cut-and-price for the capacitated vehicle routing problem with two-dimensional loading constraints,” *Proceedings of ROUTE*, 2007.

[59] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, “Classical and modern heuristics for the vehicle routing problem,” *International transactions in operational research*, vol. 7, no. 4–5, pp. 285–300, 2000.

[60] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet, “A guide to vehicle routing heuristics,” *Journal of the Operational Research society*, vol. 53, no. 5, pp. 512–522, 2002.

[61] G. J. Füllerer, “Vehicle routing with multi-dimensional loading constraints,” Uniwien, 2008.

[62] R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck, “Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem,” *Math. Program.*, vol. 106, no. 3, pp. 491–511, 2006.

[63] R. Baldacci and A. Mingozzi, “A unified exact method for solving different classes of vehicle routing problems,” *Mathematical Programming*, vol. 120, no. 2, pp. 347–380, 2009.

[64] R. Baldacci, N. Christofides, and A. Mingozzi, “An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts,” *Math. Program.*, vol. 115, no. 2, pp. 351–385, 2008.

[65] B. L. Golden, S. Raghavan, and E. A. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, vol. 43. Springer, 2008.

[66] G. Laporte, “Fifty years of vehicle routing,” *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

[67] R. Baldacci, P. Toth, and D. Vigo, “Recent advances in vehicle routing exact algorithms,” *4OR*, vol. 5, no. 4, pp. 269–298, 2007.

[68] M. Iori, J.-J. Salazar-González, and D. Vigo, “An exact approach for the vehicle routing problem with two-dimensional loading constraints,” *Transportation Science*, vol. 41, no. 2, pp. 253–264, 2007.

[69] M. Gendreau, A. Hertz, and G. Laporte, “New insertion and postoptimization procedures for the traveling salesman problem,” *Operations Research*, vol. 40, no. 6, pp. 1086–1094, 1992.

[70] G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori, “Ant colony optimization for the two-dimensional loading vehicle routing problem,” *Computers & Operations Research*, vol. 36, no. 3, pp. 655–673, 2009.

[71] E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis, “A guided tabu search for the vehicle routing problem with two-dimensional loading constraints,” *European Journal of Operational Research*, vol. 195, no. 3, pp. 729–743, 2009.

[72] C. Duhamel, P. Lacomme, A. Quilliot, and H. Toussaint, “A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem,” *Computers & Operations Research*, vol. 38, no. 3, pp. 617–640, 2011.

[73] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, “Heuristics for multi-attribute vehicle routing problems: a survey and synthesis,” *European Journal of Operational Research*, vol. 231, no. 1, pp. 1–21, 2013.

[74] C. D. Tarantilis, E. E. Zachariadis, and C. T. Kiranoudis, “A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 255–271, 2009.

[75] W. Zhu, H. Qin, A. Lim, and L. Wang, “A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP,” *Computers & Operations Research*, vol. 39, no. 9, pp. 2178–2195, 2012.

[76] M. A. Wisniewski, M. Ritt, and L. S. Buriol, “A Tabu Search Algorithm for the Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints,” *Anais do XLI Simposio Beasilero de*.

[77] A. Bortfeldt, “A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints,” *Computers & Operations Research*, vol. 39, no. 9, pp. 2248–2257, 2012.

[78] Y. Tao and F. Wang, “A new packing heuristic based algorithm for Vehicle Routing Problem with Three-dimensional Loading constraints,” in *2010 IEEE Conference on Automation Science and Engineering (CASE)*, 2010, pp. 972–977.

[79] L. Wang, S. Guo, S. Chen, W. Zhu, and A. Lim, “Two natural heuristics for 3D packing with practical loading constraints,” in *PRICAI 2010: Trends in Artificial Intelligence*, Springer, 2010, pp. 256–267.

[80] G. Fuellerer, K. F. Doerner, R. F. Hartl, and M. Iori, “Metaheuristics for vehicle routing problems with three-dimensional loading constraints,” *European Journal of Operational Research*, vol. 201, no. 3, pp. 751–759, 2010.

[81] M. Reimann, K. Doerner, and R. F. Hartl, “D-ants: Savings based ants divide and conquer the vehicle routing problem,” *Computers & Operations Research*, vol. 31, no. 4, pp. 563–591, 2004.

[82] Q. Ruan, Z. Zhang, L. Miao, and H. Shen, “A hybrid approach for the vehicle routing problem with three-dimensional loading constraints,” *Computers & Operations Research*, vol. 40, no. 6, pp. 1579–1589, 2013.

[83] Y. Marinakis, M. Marinaki, and G. Dounias, "Honey bees mating optimization algorithm for the vehicle routing problem," in *Nature inspired cooperative strategies for optimization (NICSO 2007)*, Springer, 2008, pp. 139–148.

[84] P. Lacomme, H. Toussaint, and C. Duhamel, "A GRASP×ELS for the vehicle routing problem with basic three-dimensional loading constraints," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1795–1810, 2013.

[85] C. Prins, "A GRASP × Evolutionary Local Search Hybrid for the Vehicle Routing Problem," in *Bio-inspired Algorithms for the Vehicle Routing Problem*, F. B. Pereira and J. Tavares, Eds. Springer Berlin Heidelberg, 2009, pp. 35–53.

[86] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[87] F. Wang, Y. Tao, and N. Shi, "A survey on vehicle routing problem with loading constraints," in *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, 2009, vol. 2, pp. 602–606.

[88] M. Iori and S. Martello, "Routing problems with loading constraints," *TOP*, vol. 18, no. 1, pp. 4–27, 2010.

[89] L. R. Ford Jr and D. R. Fulkerson, "A suggested computation for maximal multi-commodity network flows," *Management Science*, vol. 5, no. 1, pp. 97–101, 1958.

[90] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations research*, vol. 8, no. 1, pp. 101–111, 1960.

[91] F. Glover, "Tabu Search—Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[92] F. Glover, "Tabu Search—Part II," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.

[93] M. Gendreau, *An introduction to tabu search*. Springer, 2003.

[94] A. Lodi, S. Martello, and D. Vigo, "TSpack: a unified tabu search code for multi-dimensional bin packing problems," *Annals of Operations Research*, vol. 131, no. 1–4, pp. 203–213, 2004.

[95] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks*, vol. 44, no. 3, pp. 216–229, 2004.

[96] M. Desrochers, “An algorithm for the shortest path problem with resource constraints,” GERAD, Technical Report G-88-27, 1988.

[97] R. Bellman, “Dynamic programming and Lagrange multipliers,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 42, no. 10, p. 767, 1956.

[98] D. M. Ryan and B. A. Foster, “An integer programming approach to scheduling,” *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pp. 269–280, 1981.

[99] J. O. Berkey and P. Y. Wang, “Two-dimensional finite bin-packing algorithms,” *Journal of the operational research society*, vol. 38, no. 5, pp. 423–429, 1987.

Appendix A

Given a sequence of customers, a three-dimensional single vehicle loading problem with LIFO constraints consists of packing customers' items into a single vehicle as per sequence of their delivery to customers. Let S be a sequence of customers listed according to the order of their visit. For each k and $k' \in S, k < k'$, customer k should be visited before customer k'. In LIFO policy, no items demanded by customer k' may be placed over the items of customer k or between them and the rear of the vehicle. On this basis, it is impossible to pack the first item for delivery at the bottom of the vehicle. The objective is finding the minimum length of a vehicle needed to pack all items. All the parameters and variables are same as those presented in the Chapter 3.

The problem is formulated as following linear mixed integer programming model:

$$Min\ L_v \tag{1}$$

s.t:

$$x_i + (l_i - w_i)o_i + w_i \leq L_v \qquad \forall\, i \in I \tag{2}$$

$$y_i + l_i + (w_i - l_i)o_i \leq W \qquad \forall\, i \in I \tag{3}$$

$$z_i + h_i \leq H \qquad \forall\, i \in I \tag{4}$$

$$x_i + (l_i - w_i)o_i + w_i \leq x_{i'} + \left(1 - f_{ii'}^1\right)M \qquad \forall\, i, i' \in I_k\ , i < i'\ , k \in S \tag{5}$$

$x_{i'} + (l_{i'} - w_{i'})o_{i'} + w_{i'} \leq x_i + (1 - f_{ii'}^2)M \qquad \forall\, i, i' \in I_k\,, i < i'\,, k \in S \qquad (6)$

$y_i + l_i + (w_i - l_i)o_i \leq y_{i'} + \left(1 - f_{ii'}^3\right)M \qquad \forall\, i, i' \in I_k\,, i < i'\,, k \in S \qquad (7)$

$y_{i'} + l_{i'} + (w_{i'} - l_{i'})o_{i'} \leq y_i + (1 - f_{ii'}^4)M \qquad \forall\, i, i' \in I_k\,, i < i'\,, k \in S \qquad (8)$

$z_i + h_i \leq z_{i'} + \left(1 - f_{ii'}^5\right)M \qquad \forall\, i, i' \in I_k\,, i < i'\,, k \in S \qquad (9)$

$z_{i'} + h_{i'} \leq z_i + (1 - f_{ii'}^6)M \qquad \forall\, i, i' \in I_k\,, i < i'\,, k \in S \qquad (10)$

$f_{ii'}^1 + f_{ii'}^2 + f_{ii'}^3 + f_{ii'}^4 + f_{ii'}^5 + f_{ii'}^6 \geq 1 \qquad \forall\, i, i' \in I_k\,, i < i', k \in S \qquad (11)$

$x_{i'} + (l_{i'} - w_{i'})o_{i'} + w_{i'} \leq x_i + (1 - f_{ii'}^2)M \qquad \forall\, i \in I_k, \forall i' \in I_{k'}\,, k \,\&\, k' \in S\,, k < k' \qquad (12)$

$y_i + l_i + (w_i - l_i)o_i \leq y_{i'} + \left(1 - f_{ii'}^3\right)M \qquad \forall\, i \in I_k, \forall i' \in I_{k'}\,, k \,\&\, k' \in S\,, k < k' \qquad (13)$

$y_{i'} + l_{i'} + (w_{i'} - l_{i'})o_{i'} \leq y_i + (1 - f_{ii'}^4)M \qquad \forall\, i \in I_k, \forall i' \in I_{k'}\,, k \,\&\, k' \in S\,, k < k' \qquad (14)$

$z_{i'} + h_{i'} \leq z_i + (1 - f_{ii'}^6)M \qquad \forall\, i \in I_k, \forall i' \in I_{k'}\,, k \,\&\, k' \in S\,, k < k' \qquad (15)$

$f_{ii'}^2 + f_{ii'}^3 + f_{ii'}^4 + f_{ii'}^6 \geq 1 \qquad \forall\, i \in I_k, \forall i' \in I_{k'}\,, k \,\&\, k' \in S\,, k < k' \qquad (16)$

$f_{ii'}^1, f_{ii'}^2, f_{ii'}^3, f_{ii'}^4, f_{ii'}^5, f_{ii'}^6, o_i \in \{0,1\} \qquad i\,, i' \in I \qquad (17)$

$x_i, y_i, z_i \in \mathbb{Z}^+ \qquad \forall\, i \in I \qquad (18)$

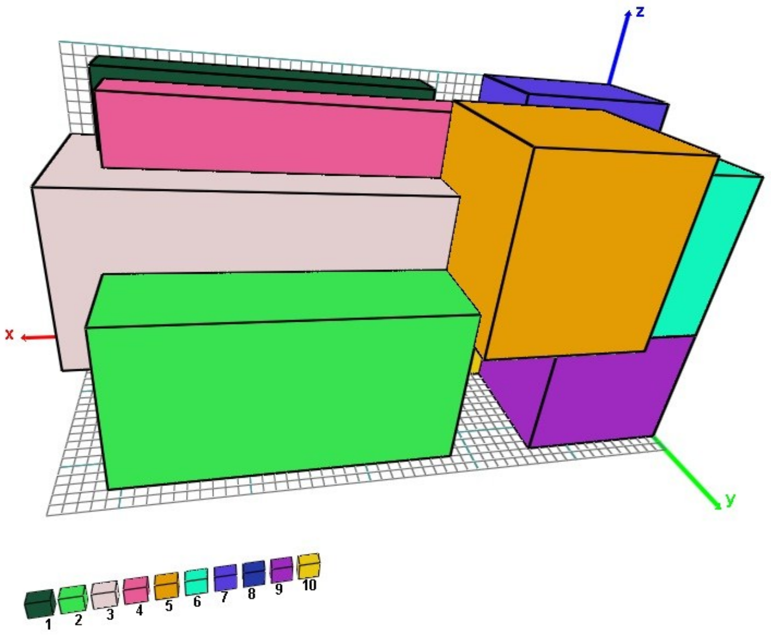
The minimum length of vehicle required to pack all items is selected in objective function (1). Constraints (2) to (4) ensure that each item is in vehicle geometric boundary. Constraints (5) to (11) ensure that items related to one customer do not overlap each other. Constraints (12) to (16) ensure that a pair of items delivered to different customers does not overlap each other while satisfying LIFO policy. Note that the objective function in the mathematical formulation can be extended to the problem case where the width or height of vehicle is unknown.

To validate the mathematical model, an example problem of 6 customers is solved by CPLEX. In this example S= {1, 2, 3, 4, 5, 6}, and the width and height of vehicle are set equal to 25 and 30 respectively. The total number of items is 10.The position and orientation of items resulted by CPLEX are summarized in four last columns in the following table:

Table A-1: Result for an example problem of 3DVLP-LIFO

Customer	Item number	Item dimension			Orientation	Item coordinate		
		l	w	h	o	x	y	z
1	1	30	5	7	1	22	0	22
2	2	29	8	15	1	22	15	0
3	3	33	15	16	1	22	0	6
	4	36	5	6	1	15	5	22
4	5	15	15	17	1	7	10	12
5	6	13	7	15	0	0	10	12
	7	15	10	8	1	0	0	22
6	8	20	9	16	1	0	0	6
	9	12	14	12	0	0	11	0
	10	27	11	6	1	0	0	0

First and second columns give customers and items related to each customer respectively. The minimum length of vehicle needed to pack items as per sequence of their delivery to customers is 55. We use the software available at http://www.isima.fr/~lacomme/3lcvrp/3lcvrp.html to give the graphical representation of the solution obtained by CPLEX. Parts a and b in following figures give two different view sides of graphical representation of loading solution due to LIFO policy. For clarity, items are represented with colour boxes. It is seen from the figure that items related to customer 1 can be uploaded without moving the other items.



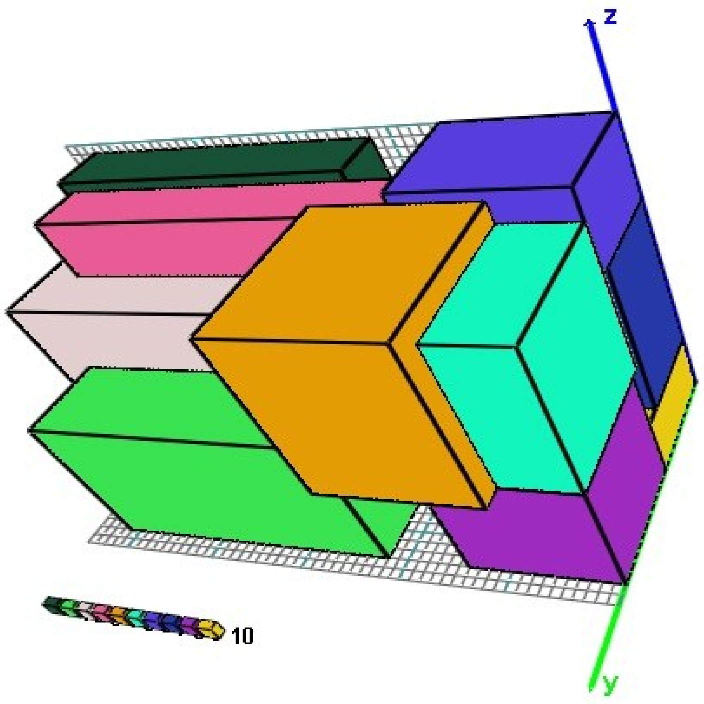


Figure A-1: Different view sides of loading pattern solution to the example problem