Studies on Optimization of Container Loading and Vehicle Routing for Green Logistics

2012

Jidong Ren

Graduate School of Engineering
Kyoto University
JAPAN

Acknowledgements

First of all, I would like to express my sincere appreciation to Professor Tetsuo Sawaragi for his guidance. He gave me much opportunity and continuous support.

I am deeply grateful to Dr. Yajie Tian. She gave me much guidance and encourage. Her advice provided insights into research activity for me.

I would like to thank Ph.D. Yukio Horiguchi, Ph. D. Hiroaki Nakanishi, Ms. Shinobu Minato, Ms. Aya Araki, and all the members in Sawaragi lab. They have been always willing to help me.

Finally, I would like to express my gratitude to my family for their heartfelt cooperation and encouragement.

Abstract

Traditional logistics has concentrated on minimizing costs subject to operational constraints. In recent years there has been increasing concern about the environmental effects on the planet of human activity and current logistics practices may not be sustainable in the long term. Green logistics includes logistics practices and strategies that reduce the environmental and energy footprint of freight distribution. It focuses on material handling, waste management, packaging and transport. The container loading problem (CLP) and vehicle routing problem (VRP) are of importance in both traditional logistics and green logistics. The CLP concerns to best possible capacity utilization of space, and the VRP is used for finding the minimum-cost to minimum-distance route in transportation. In recent years many transportation management systems (TMS) are designed to manage transportation operations. However, in most existing TMS the CLP and VRP were treated separately, and the green logistics issue was not taken into account. The author has several years of experience in developing practical logistics systems. For improving existing transportation management systems, this research aims at providing a link between the CLP and VRP. Moreover, fuels consumption and GHG emissions are also taken into account. Therefore this research has key roles to play in dealing with green logistics issue.

Chapter 1 introduces the background of this research, including loading and routing optimization in green logistics, NP-hard problems and algorithms, the CLP, VRP and their combination – the three-dimensional loading capacitated vehicle problem (3L-CVRP).

In Chapter 2, an exact algorithm is proposed for the single CLP. In the algorithm an effective method is used for generating possible positions for packing items, which greatly reduce the search scope. For the reduction of the computing time, a heuristics method is incorporated in the exact algorithm. In the heuristics, blocks made up of identical items with the same orientation are selected so that they can be packed into a container. Five evaluation functions are proposed for block selection, and the different blocks selected by each evaluation function constitute the branches of the search tree. The methods of space splitting and merging are also embedded in the algorithm to facilitate efficient use of the container space. In addition, the proposed algorithm covers an important constraint called shipment priority. This constraint of shipment priority is important, both for the multiple CLP and the 3L-CVRP, as mentioned later.

In Chapter 3, the multiple CLP is addressed, in which the objective is to minimize the number of containers or to maximize the average utilization of multiple containers. An exact algorithm is proposed, in which the distribution of items into containers is treated as partition of multiset. In consequence repetitions caused by same size items are avoided and the computing time is reduced. Heuristic algorithms for the multiple CLP are also proposed. The items with large volume are usually difficult to make efficient use of the container space. These items are set higher priority over other items, and preferentially assigned into the containers. Within the proposed algorithms a CLP algorithm is used for solving the single container loading under the constraint of shipment priority. The proposed algorithms achieve excellent results with reasonable computing time. For rail transport, ship transport and airline transport, the model is usually "one start point, one destination point", and the number of containers is directly relevant to the cost, energy consumption and CO2 emissions. However, in road transport, items should be delivered from one or more depot to multiple customers, therefore the combination of routing and loading must be considered, as mentioned in the next section.

Chapter 4 addresses the 3L-CVRP. Both the single CLP algorithms and the multiple CLP algorithms are used. In addition, algorithms for travelling salesman problem (TSP) are also used. A branch and bound algorithm based on set partition model is proposed. It is very time consuming to check whether the items can be feasibly loaded into a vehicle. Therefore the loading constraint is replaced with the constraint of volume ratio. The validity of the proposed algorithms for the 3L-CVRP is examined by using the test data that come from the literature. Considering the issue of green logistics, we take the fuel consumption and CO2 emissions into account. As mentioned in the report of European Committee for Standardization, for a certain vehicle travelling with a constant speed, the fuel consumption and CO2 emissions are approximately proportional to the travelling distance and linear correlate to the weight of loaded items. The traditional models and algorithms cannot be applied directly. An algorithm is proposed for solving the TSP with fuel consumption, which can be incorporated in the 3L-CVRP algorithm. Because the CO2 emission has similar function to the fuel consumption, a similar algorithm is proposed for minimizing the CO2 emissions. Computational experiments show that the fuel consumption and CO2 emissions can be reduced with a small increase in travelling distance. Furthermore, in the real-world instance, for improving the utilization of vehicles, some constraints for traditional VRP may be violated. The items demanded by one customer may exceed the capacity of one vehicle. Therefore one customer may be served by multiple vehicles. In another real-world instance, the items are divided into groups according to their due dates of delivery, and only the high-priority items should be delivered completely. Algorithms for the generalized 3L-CVRPs have also been proposed. The computational results for real-world instance show that proposed algorithms are fast and high-quality for solving practical problems.

Chapter 5 summarizes this research. This research links the CLP and the VRP, and the green logistics issue is considered. Some well defined data structures are used, such as the staircase packing and the multiset partition. The integration of the exact and heuristic algorithms obtained excellent results, both for test data that come from the literature and for real-world instance.

Moreover, we can reduce fuel consumption and CO2 emissions in transportation, with little increase in travelling distance.

Only rectangular items have been considered in this research. Further research is required to deal with other shapes of items, such as circular column, sphere or irregular items. Other variants of 3L-CVRP in real-world applications should also be considered, such as 3L-CVRP with time windows, 3L-CVRP with multiple depots and 3L-CVRP with multiple vehicle types. Algorithms for routing and loading optimization are, in the real world, just part of the story. The algorithms have to be embedded in a system that enables the decision-maker to actually use it. The system has to be integrated into the information system of the enterprise. Database and user interface are also important part of the system. The system usually has to interact with a number of different systems in an organization. It may receive information from a higher level system and provide information to a lower system. Robustness and reactive decision making is also an important issue for real-world transportation management system. In practice, it often happens that soon after a vehicle schedule has been generated, an unexpected event happens that forces the decision-maker to make changes. It is necessary for the original vehicle schedule to be robust so that the changes after a disruption are minimal.

Contents

Acknowledgments

Abstract

1 Introduction 1
- 1.1 Loading and Routing Optimization for Green Logistics . . 1
- 1.2 NP-Hard Problems and Algorithms 3
- 1.3 Container Loading Problem 6
- 1.4 Vehicle Routing Problem 9
- 1.5 The Combination of Loading and Routing Problems . . . 9

2 Single Container Loading Problem 12
- 2.1 Problem Formulation of SCLP 12
- 2.2 Exact algorithm for SCLP 14
- 2.3 Heuristic Algorithm for SCLP 18
  - 2.3.1 Block Building 20
  - 2.3.2 Block Evaluation 22
  - 2.3.3 Procedure of the Greedy Heuristic 24
  - 2.3.4 Space Splitting and Merging 26
  - 2.3.5 Tree Search 25
- 2.4 Shipment Priority 28
- 2.5 Computational Results 32
  - 2.5.1 Computational Results for LN Data 35
    - 2.5.1.1 Results for SCLP 35
    - 2.5.1.2 Results for SCLPSP 36
  - 2.5.2 Computational Results for BR Data 37
    - 2.5.2.1 Results for SCLP 37
    - 2.5.2.2 Results for SCLPSP 39
- 2.6 Conclusions 39

3 Multiple Container Loading Problem 41
- 3.1 Problem Formulation of MCLP 41
- 3.2 Set Partition and Multiset Partition 42
- 3.3 Exact Algorithm For MCLP 43
- 3.4 The Priority-Considering Approach for MCLP 45
  - 3.4.1 The Approach for 3DBPP 45

3.4.2 The Approach for 3DKP 49
3.5 Computational Results 50
3.6 Conclusions 55

4 Three-Dimensional Loading Capacitated Vehicle Routing Problem 57

4.1 Problem Description of 3L-CVRP 57
4.2 NP-Hard problem with NP-Hard Constraint 58
4.3 Practical Loading Constraints in Vehicle Routing Context . 59
4.4 Exact Algorithm For 3L-CVRP 60
4.5 Relaxation Method For 3L-CVRP 63
4.6 Fuel Consumption and GHG Emissions 64
4.7 The Traveling Salesman Problem with Fuel Consumption . 65
4.8 The 3L-CVRP without Item Partition and Shipment Priority 68
4.9 Computational Results 69
4.9.1 Computational Results for Distance 70
4.9.2 Computational Results for Fuel Consumption and CO2 Emissions 73
4.9.3 Computational Result for an Real-world Instance . 73
4.10 Conclusions 76

5 Conclusions 77

References 79

Published Papers 83

List of Figures

1 The CLP 7
2 The 3L-CVRP 10
3 Container in Three-Dimensional Coordinate System 13
4 Item Orientations 14
5 Staircase Packing and Pseudo-Staircase Packing 16
6 Block in Space 20
7 Possible Blocks 22
8 Evaluations of Block 23
9 Algorithm HEUR_SCLP 25
10 Split Space 26
11 Merge spaces 27
12 Tree Search for SCLP 27
13 Background of SCLPSP 30
14 Three Loading Models 31
15 Tree Search for SCLPSP 31
16 Volume Utilization in Different *breath* Values 34
17 Running Time in Different *breath* Values 34
18 Illustrative Result of LN02 35
19 Illustrative Result of LN01&02 37
20 Supporting Area 59
21 The TSP Considering Distance or Fuel Consumption 66

List of Tables

1 Algorithm EXAC_SCLP 19
2 Algorithm TRS_SCLP 29
3 Delete Infeasible Partial Solutions 32
4 Algorithm TRS_SCLPSP 33
5 Results for LN Data 36
6 Results for LN Data Considering Priority 37
7 Results for BR Data 38
8 Frequencies of the Five Evaluations 38
9 Results from BR Data Considering Priority 39
10 Algorithm EXAC_MCLP . . . .
45
11 Algorithm PRI_BBP 46
12 Procedure LMC_BBP 46
13 Algorithm PRI_KP 48
14 Procedure LMC_KP 49
15 Comparative Results for IV1 data 51
16 Results for IV1 data with Different Values of d 52
17 Comparative Results for IV2 data 53
18 Comparative Results for MO data 54
19 Running Time (Sec.) for One Test Case 55
20 Algorithm EXAC_3LCVRP 61
21 Comparative Results for GE Data 71
22 Average Running Time 71
23 Results for Different Loading Constraints 72
24 Results for Fuel Consumption and CO2 Emissions 74
25 Real-World Instance 75
26 Comparative Results for Real-World Instance 75

List of Symbols

A,A_1: set of point 17
$\bar{A}$: supporting area 60
a_{ij}: distance of two vertices 57
$b_{i\omega}$: binary coefficient 60
bl: length of block 20
bw: width of block 20
bh: height of block 20
$breadth$: in tree search only the best $breadth$ nodes are kept 28
c_i: a customer in permutation 65
D: weight of all the items 65
d: division parameter 47
E, E': set of edges 57
e: base of natural logarithm 42
F_k: set of possible points 15
f: number of customers 57
g_i: total weight of items demanded by customer i 62
H: height of container (vehicle) 13
h, h_i: height of item i 13
I: Set of items 13
i: general index 13
il: length of item 20
iw: width of item 20
ih: height of item 20
j: general index 13
K: loaded weight of a vehicle 64
k: general index 13
L: length of container (vehicle) 13
l, l_i: length of item 13
M, M_0: CO2 emissions 64
M_1: in Step 1 of RELAX_3LCVRP, only M_1 best solutions are kept 64
M_2: in Step 2 of RELAX_3LCVRP, only M_2 feasible routes are selected 64
m: number of item types 13
$\boldsymbol{N}$: number of items 43
n_i: number of items of type i 13
P, P_k: set of packed items 13
p_i: packed item 13
Q, Q_0, Q_1, Q_2: fuel consumption 65
q: demanded weight of items 63

R: travel distance of a vehicle 64
r: general index 17
s_{ij}: number of items of type i packed in the j-th packed container . . 41
sl: length of space 20
sw: width of space 20
sh: height of space 20
T: weight capacity of vehicle 57
t: number of packed containers (vehicles) 41
u_k: number of packed containers of type k 41
$V, V\prime$: set of vertices (depot and customers) 57
v_i: volume of an item of type i 41
W: width of container (vehicle) 13
w, w_i: width of item 13
x_i: x coordinate 13
xN: number of items along the x-axis in a block 20
y_i: y coordinate 13
yN: number of items along the y-axis in a block 20
z_i: z coordinate 13
zN: number of items along the z-axis in a block 20
α: minimum fraction of supported area 60
β: unit fuel consumption (g/tone-kilometre) 65
Γ: available number of vehicles 57
γ: unit CO2 emissions (ml/tone-kilometre) 64
δ: number of customers (in TSP) 65
ζ_ω: binary variable 60
η_i: total volume of items demanded by customer i 62
Θ: parameter for modifying volume ratio 63
θ: number of container types 41
Λ: the set of all possible loaded weight of a single route 63
μ_i: value per unit volume of an item of type i 41
ρ: a given ratio of the vehicle space 63
σ_ω: optimal distance of route ω 60
φ_k: volume of a container of type k 41
$\psi(i)$: the value of a least distance single route passing through i . . . 41
ω: a single route 60

Chapter 1

Introduction

1.1 Loading and Routing Optimization for Green Logistics

Logistics is the management of the flow of goods between the point of origin and the point of use in order to meet the requirements of customers or corporations. Logistics involves the integration of information, transportation, inventory, warehousing, material handling, and packaging. Logistics is a channel of the supply chain which adds the value of time and place utility (Wallenburg, 2011). In the last decades, logistics has become one of the key factors in economy, and received considerable attention from both governments and enterprises. In recent yeas the proportions of logistics cost to GDP are about 10% in US, Europe and Japan; in China the proportion is above 16% (Lin, 2011).

Traditional logistics for production and distribution has concentrated on minimizing costs subject to operational constraints. In recent years there has been increasing concern about the environmental effects on the planet of human activity and current logistics practices may not be sustainable in the long term. There is therefore increasing interest in green logistics from companies and governments. Green logistics is concerned with producing and distributing goods in a sustainable way, taking account of environmental and social factors (Sbihi and Eglese, 2010). Green logistics focus on material handling, waste management, packaging, warehousing and transportation. Green logistics activities include measuring the environmental impact of different distribution strategies, reducing the energy usage in logistics activities, reducing waste and managing its treatment.

Cutting and packing problem (C&P; Dyckhoff 1990) is a large class of problems which represent problems of the optimal use of resources. Cutting problem focuses on minimizing the waste of material, which involves, for example, the cutting of paper rolls into narrower rolls in the paper industry, the cutting of large wooden boards into smaller rectangular panels in the furniture industry. Packing problem focuses on minimizing the waste of space, including packing items into warehouse, containers, trucks or pallets in logistics applications. Both cutting problem and packing problem have essentially the same logical structure; they require that 'large objects' are to be divided into 'small items' in such a way that waste is minimized. The development of even more effective cutting and packing methods is an important task in green logistics because, in view of the size of today's productions and distribution processes, even relatively minor growth in the utilization of material and space capacities can result in considerable material and energy savings and reductions in carbon emissions. For example, better packaging of goods reduces materials consumption and waste, higher utilization of warehouse space involves reduction of energy consumption.

One issue of this paper is the container loading problem (CLP), which calls for packing a set of three-dimensional rectangular items in one or more three-dimensional rectangular containers, and the objective is the maximum utilization of container space or minimum number of containers. Here 'container' is a general word, which represents a lot of real-world objects in which small items can be packed, such as box, pallet, vehicle and warehouse space. The CLP is the most widely used problem among the cutting and packing problems. In real-world applications, packing of different shapes of items is also involved, such as packing of circular column, sphere or irregular items. However, in most cases the packages of items are three-dimensional rectangular. The CLP is closely related to other cutting and packing problems. For example, the CLP contains the two dimension rectangular packing problem, and can be easily adapted for rectangular cutting problem because they have the same logical structure (Fanslau and Bortfeldt, 2010).

More important, this paper focuses on the combination of the loading optimization and routing optimization, due to the requirement of transportation management (TM). TM is the main part of the logistics processes, which includes scheduling, lead time, routing and loading of vehicles and other transport medium. Considering that, on average, 3.5% of manufacturers sales costs and 40-60% of total logistics costs are devoted to the movement of products, TM is a crucial issue in todays business environment (Aprile et al., 2007). Transportation can be divided into two parts: line-haul transportation and branch transportation. Line-haul transportation is the movement of items between two major cities or ports by rails, ships, airlines or main high-ways. The items are usually delivered through constant route, and the number of containers is directly relevant to the cost, energy consumption and CO2 emissions. Another part is the branch transportation, in which items are delivered through a much more complex network of high-ways and urban roads. Usually items are delivered from one or more depots to a large number of customers. In branch transportation, both the loading optimization and routing optimization should be taken into account. In the literature, many authors addressed the routing optimization which is known as the vehicle routing problem (VRP). It calls for the determination of the optimal set of routes to be performed by a fleet of vehicles to serve a given set of customers. However, very few papers have dealt with integrated approaches for the CLP and the VRP. Only in recent years algorithms combining these two problems have been proposed in the literature. Combining two difficult problems leads to a considerable increase of difficulty, but on the other hand it allows a better solution of the corresponding logistics targets.

In recent years, many transportation management systems (TMS) are designed to manage transportation operations. These systems reduced transportation costs and raised service levels. However, as mentioned before, in most systems the CLP and VRP were treated separately. Therefore there may be great discrepancy between the computer simulated result and the practical transportation plan. Furthermore, in most systems the only some traditional goals were taken into account, such as costs and service levels, and few existing systems considered to reduce the energy (fossil fuels) usage and the greenhouse gas (GHG) emissions, which are required by green logistics.

For improving the existing transportation management systems, this paper addresses a model which links the CLP and VRP, and moreover, fuels usage and GHG emissions are also taken into account. Therefore the model has key roles to play in dealing with green logistics issue.

The contributions of this paper are the following:

- Effective algorithms are proposed for the CLP and the combination of CLP and VRP.
- Green logistics issues are taken into account, such as energy consumption and CO2 emissions.
- Issues for real-world application are considered, which generalize the traditional models of loading and routing optimization.

The rest of this chapter is organized as follows. Sections 1.2 introduces the NP-hard problems and algorithms, because both the CLP and VRP are belong to the class of NP-hard problem, and this paper focus on proposing algorithms for solving the problems. Sections 1.3 - 1.5 are literature overviews of the CLP, VRP and the combination of the two problems, respectively.

1.2 NP-hard Problems and Algorithms

Many operational research models in logistics management systems are combinatorial optimization problems. An optimization problem is the problem of finding the best solution from all feasible solutions. Usually the 'optimal' value of an objective function is defined by the minimum or maximum. Especially, for an optimization problem, if the set of feasible solutions is discrete or can be reduced to discrete, the problem is a combinatorial optimization problem. There are a variety of combinatorial optimization problems that appear in many application fields, including the traveling salesman problem (TSP), the minimum spanning tree problem (MSTP), the bin packing problem (BPP), and the CLP and VRP mentioned before.

An algorithm is a specific, finite set of instructions for carrying out a procedure or solving a problem. In the late 1960s, the fundamental nature of algorithms was discussed; Edmonds (1962) called an algorithm which runs in polynomial time of the input size a 'good' algorithm. In computational complexity theory, an algorithm for a problem is regarded as efficient if the time complexity of an algorithm is bounded above by a polynomial of the instance size of every problem instance. Some combinatorial optimization problems are polynomial solvable problems, for instance, the shortest path problem can be can be solved by the well-known Dijkstras algorithm and the minimum spanning tree problem can be solved by Prims algorithm, both in polynomial time. However, there are many combinatorial optimization problems to which no efficient algorithms are known. These problems were known to be difficult to obtain an exact optimal solution and the difficulties were proved in the sense of NP-hardness, which was the notion proposed around 1970.

For each optimization problem, there is a corresponding decision problem that asks whether there exist a feasible solution such that the value of objective function is better (larger or smaller) than a particular value. A class NP is the set of decision problems such that any yes instance has a certificate that can be verified in polynomial time. A class NP-complete is a subclass of NP, which has a property that any problem in NP can be reduced to problems in NP-complete in polynomial time. This means that if some problem in NP-complete can be solved in polynomial time, then all problems in NP can be solved in polynomial time as well. A class NP-hard is s set of optimization problems that is at least as hard as NP-complete. It is strongly believe that NP-hard problems admit no polynomial time algorithm. In other words, solving these problems exactly may necessitate enumerating an essential portion of all the solution candidates in a given instance, whose number increase exponentially as the problem size grows.

Cook (1971) first proved that SAT is an NP-complete problem, and in the subsequent years, the foundations for the theory of NP-completeness were established. A class NP-hard is a set of optimization problems that is at least as hard as NP-complete. Nowadays many combinatorial optimization problems are proved to be NP-hard, such as the TSP, BBP, VRP and CLP. It is strongly believed that an NP-hard problem cannot be solved in polynomial time of the input size. In other words, solving an NP-hard problem exactly may necessitate enumerating an essential portion of the set of all solutions, whose number increases exponentially as problem size grows.

Because of the difficulty and enormous practical importance of combinatorial optimization problems, a large number of solution techniques have been proposed. The available algorithms can be classified into two main classes: exact and approximate algorithms. Exact algorithms are guaranteed to find the optimal solution and to prove its optimality for every finite size instance of a combinatorial optimization problem within an instance-dependent, finite run-time. If optimal solutions cannot be computed efficiently in practice, the only possibility is to trade optimality for efficiency. In other words, the guarantee of finding optimal solutions can be sacrificed for the sake of getting very good solutions in polynomial time. The approximate algorithms include two class: heuristic methods and metaheuristics.

Some well known exact algorithms are linear and integer programming, branch-and-bound, lagrangian relaxation and dynamic programming. In recent years remarkable improvements have been reported for exact algorithms when applied to some NP-hard problems. The exact algorithms are usually time consuming and difficult to extend if some details of the problem formulation change. However the following advantages of exact algorithms attract researchers' interesting:

- Proven optimal solutions can be obtained if the algorithm succeeds.
- Valuable information on solution structure are obtained from a well-designed exact algorithm.
- Upper/lower bounds to the optimal solution can be obtained even if the algorithm is stopped before completion.

Most heuristic algorithms are experiment-based and problem-specified. Examples includes the nearest neighbor method for the TSP, the saving method for the VRP, the block building method for the CLP.

A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms (Blum and Roli,2003). Most metaheuristics are inspired by biological or physical phenomena. Examples include simulated annealing (SA), tabu search (TS), genetic algorithm (GA), ant colony optimization algorithm (ACO) and particle swarm optimization (PSO. Most of them are belong to the class of local search, which iteratively applying small modifications (local moves) to a solution in the hope of finding a better one. In the last decades, metaheuristics have been shown to be the most successful class of approximate algorithms. Many metaheuristics implement some form of stochastic optimization. Because metaheuristic frameworks are defined in general terms, metaheuristic algorithms can be adapted to fit the needs of most optimization problems. Especially for complicated real-world problems or large-scale instances, metaheuristics often offer a better trade-off between solution quality and computing time.

In recent years, algorithmic developments in exact method, heuristics and metaheuristics have recently drawn the three fields closely together, and combinations of the three method are now common. The resulting methods often integrate existing exact procedures to solve subproblems generated by a decomposition strategy, a restriction strategy or a relaxation strategy. The results of solving these subproblems are used to guide a higher-level heuristic (Raidl and Puchinger, 2008).

A famous formula in computer science is 'Algorithms + Data Structures = Programs' (Wirth, 1976). Data structure is the way to store and organize the data. For example, for TSP, a data structure of solution is permutation. Many combinatorial optimization problems can be classified into some basic classes according to their data structure: the class of subset problems, the class of permutation problems, and the class of partition problems (Woeginger, 2001). In a subset problem, every feasible solution can be specified as a subset of an underlying ground set. For instance, fixing a truth-assignment in the satisfiability problem corresponds to selecting a subset of TRUE variables. In the independent set problem, every subset of the vertex set is a solution candidate. In a permutation problem, every feasible solution can be specified as a total ordering of an underlying ground set. For instance, in the TSP every route corresponds to a permutation of the cities. In single machine scheduling problems, feasible schedules are often specified as permutations of the jobs. In a partition problem, every feasible solution can be specified as a partition of an underlying ground set. For instance, in parallel machine scheduling problems, feasible schedules are often specified by partitioning the job set and assigning every part to another machine.

This paper uses some problem-specified exact algorithms, incorporating heuristics for reducing computing time. Some well-designed data structures are also used. The reasons for not using metaheuristics are the folloing:

- Metaheuristics do not suit for all kinds of data structures.
- Metaheuristics are much more time consuming than heuristic algorithms.
- For small-scale instances, metaheuristics are usually inferior to well-designed exact algorithms.
- Metaheuristics are problem-independent frameworks, which makes it difficult for them to make good use of problem-specified properties.

1.3 Container Loading Problem

The container loading problem (CLP), in its basic model, can be described as follows: a set of three-dimensional, rectangular items are to be packed in one or more containers in a manner that uses the container space as efficient as possible (Figure 1). The problem has several applications in manufacturing, logistics and so on. Here 'container' is a general word, which can represent a lot of in real-world objects, including box, pallet, vehicle, ship and warehouse space.

Dyckhoff (1990) classified the CLP into two categories: single container loading problem (SCLP) and multiple containers loading problem (MCLP). In the SCLP, only one container is considered and the objective is to maximize the utilization of the container space. In the MCLP, more than one container is concerned. MCLP can also be classified into two types of problems. One is the three-dimensional bin packing problem (3DBPP), for which the available container space is sufficient to pack all the items and the objective is to minimize the number of containers used, or to minimize the total cost of the containers used (if more than one container type is available, i.e., the containers are of different sizes). Usually the cost of a container has been assumed to be proportional to the volume of the container, when the cost data on real-life problems is absent. The other is the three-dimensional knapsack problem (3DKP), for which the available container space is not enough to pack all items and the objective is to maximize the volume or the value of the packed items.

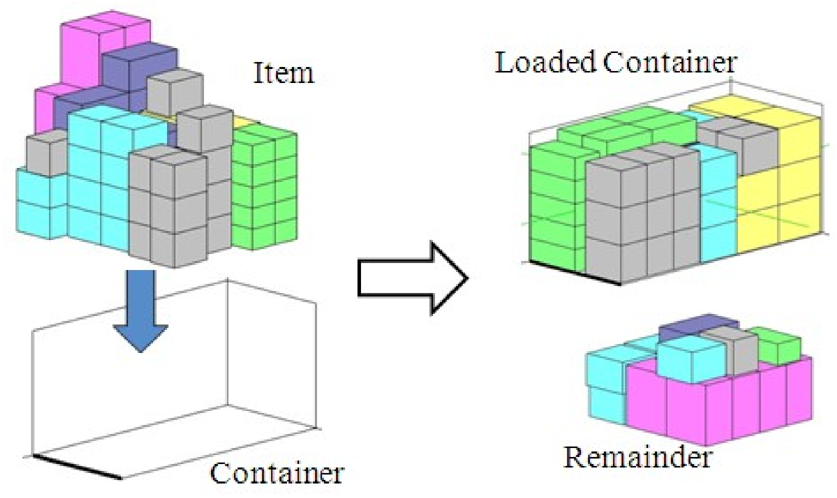


Figure 1: The CLP.

The items to be packed are categorized into types. Two items are the same type if they have the same dimensions. If there is only one item type in a item set, it is described as homogeneous. If there are only a few item types with a relatively large number of specimens per type, this is a weakly heterogeneous item set; on the other hand, if there are many item types with only a few exemplars per type, the item set is strongly heterogeneous.

Wächer et al. (2007) proposed a more elaborate typology for cutting and packing problems, under the typology the SCLP can be classified into the Single Large Object Placement Problem (SLOPP, weakly heterogeneous) and the Single Knapsack Problem (SKP, strongly heterogeneous).

And the MCLP can be classified into seven types of problems as follows:

- Single Bin Size Bin Packing Problem (SBSBPP).
- Multiple Bin Size Bin Packing Problem (MBSBPP).
- Residual Bin Packing Problem (RBPP).
- Multiple Identical Large Object Placement Problem (MILOPP).
- Multiple Heterogeneous Large Object Placement Problem (MHLOPP).
- Multiple Identical Knapsack Problem (MIKP).
- Multiple Heterogeneous Knapsack Problem (MHKP).

The typology is according to the assortment of containers and items. For example, in MBSBPP the containers are weakly heterogeneous, and in MIKP the items are strongly heterogeneous and the containers are identical. The first three types of problems belong to the 3DBPP and the last four types belong to the 3DKP.

In the last years, several types of algorithms have been proposed to solve the SCLP. Most algorithms are heuristics (George and Robinson, 1980; Bischoff and Ratcliff, 1995; Moura and Oliveira, 2005), or metaheuristics such as genetic algorithms (Bortfeldt and Gehring, 2001, Gehring and Bortfeldt, 2002, Techanitisawad and Tangwiwatwong, 2004), tabu search (Bortfeldt and Gehring, 1998), and simulated annealing (Jin et al., 2004). Tree search methods (Pisinger, 1998; Eley, 2002; Wang et al., 2008; Fanslau and Bortfeldt, 2010) have also been proposed for solving the CLP.

Many existing SCLP algorithms are based on different heuristic packing approaches such as wall building approach, stack building approach, guillotine cutting approach, and block building approach (or cuboid arrangement approach). Pisinger (2002) gave an excellent overview of these approaches. For instance, the wall building approach fills the container with vertical layers ('walls') that follow along the longest side of the container (George and Robinson, 1980; Pisinger, 1998; Pisinger, 2002; Moura and Oliveira, 2005). The block building approach fills the container with cuboid blocks that mostly contain only identical items with the same spatial orientation (Bortfeldt and Gehring, 1998; Eley, 2002; Mack et al., 2004; Parreao et al., 2008; Fanslau and Bortfeldt, 2010; Ren et al., 2011).

Unlike the SCLP, not much research has been devoted to the MCLP. Usually the sequential strategy or the simultaneous strategy is used for adapting the SCLP algorithm to the MCLP algorithm.

- Sequential strategy: the containers are filled one by one by using a SCLP algorithm.
- Simultaneous strategy: the items are simultaneously stowed into multiple containers.

Ivancic et al. (1989) addressed the 3DBPP and Mohanty et al. (1994) addressed the 3DKP. Both of them used sequential strategies and considered different container types. Bischoff and Ratcliff (1995) and Eley (2002) presented approaches for the 3DBPP with only one container type. They used both a sequential and a simultaneous loading strategy and compared the results of both strategies. Bortfeldt (2000) examined both problem types with different container types and extended the sequential strategy by diversifying the search. Other approaches were also presented. For example, Terno (2000) used a pre-assignment strategy to solve the multiple pallet loading problem with only one pallet type. In his strategy large and small items are distributed uniformly over all containers, and then a single container algorithm is applied for each pre-assigned container. Eley (2003) presented a bottleneck approach based on integer programming for solving both the 3DBPP and the 3DKP with different container types. Within his approach a single container algorithm is used to produce alternative loading patterns and a solution can be considered as a linear combination of these loading patterns.

Few papers proposed exact algorithms for CLP. Martello et al. (2000) proposed an exact branch-and-bound algorithm for the 3DBPP with one container type, which also incorporated approximation algorithms. In his paper an exact algorithm has also been proposed for SCLP, which is based on the data structure of staircase packing.

1.4 Vehicle Routing Problem

The vehicle routing problem (VRP) can be described as follows: a set of routes for a fleet of vehicles based at one or several depots must be determined for a number of geographically dispersed cities or customers, and the objective is to deliver a set of customers with known demands of items on minimum-cost vehicle routes originating and terminating at a depot.

In practice several variants of the problem exist because of the diversity of operating rules and constraints encountered in real-life applications. Thus the VRP should perhaps be viewed as a class of problems, such as the Distance-Constrained VRP, the VRP with time windows, the VRP with bachhauls, and the VRP with pickup and delivery. For example, in the VRP with time windows, each customer specifies a time interval, called time window. The travel time between two customers and the service time for customer and the time when a vehicle leaves a depot are known in advance. The service for each customer must start within the given time window, moreover when vehicles arrive before the beginning of a time window they have to wait until the beginning of the time window to start their service.

In 1959, Dantzig and Ramser first introduced the VRP. The described a real-world application concerning the delivery of gasoline to service stations and proposed a mathematical formulation and an algorithm. A few years later in 1964, Clarke and Wright proposed an effective greeedy heuristic that improved on the Dantzi-Ramser algorithm. In the last decades, hundreds of models and algorithms were present for the exact and approximate solution of the different variants of the VRP. Recent overview on heuristic and meta-heuristic approaches for the VRP can be seen in Toth and Vigo (2005).

The VRP generalizes the well-known traveling salesman problem (TSP) but is much more difficult to solve in practice. Whereas there exist exact algorithms capable of routinely solving TSPs containing hundreds or thousands of vertices (Applegate et al., 2007), this is not the case of the VRP for which the best exact algorithms can only solve instances involving approximately 100 vertices (Baldacci et al., 2008). Because real instances of the VRP often exceed this size and solutions must often be determined quickly, most algorithms used in practice are heuristics. In recent years, several powerful metaheuristics have been developed.

1.5 The Combination of Container Loading and Vehicle Routing Problems

As mentioned by Davies and Bischoff (1999), much of the literature on CLP has considered the container purely as a storage device rather than a transport medium. In other words, only the maximum volume utilization or minimum number of containers has been considered and the transportation of the loaded containers has been ignored. On the other hand, most literature on the VRP

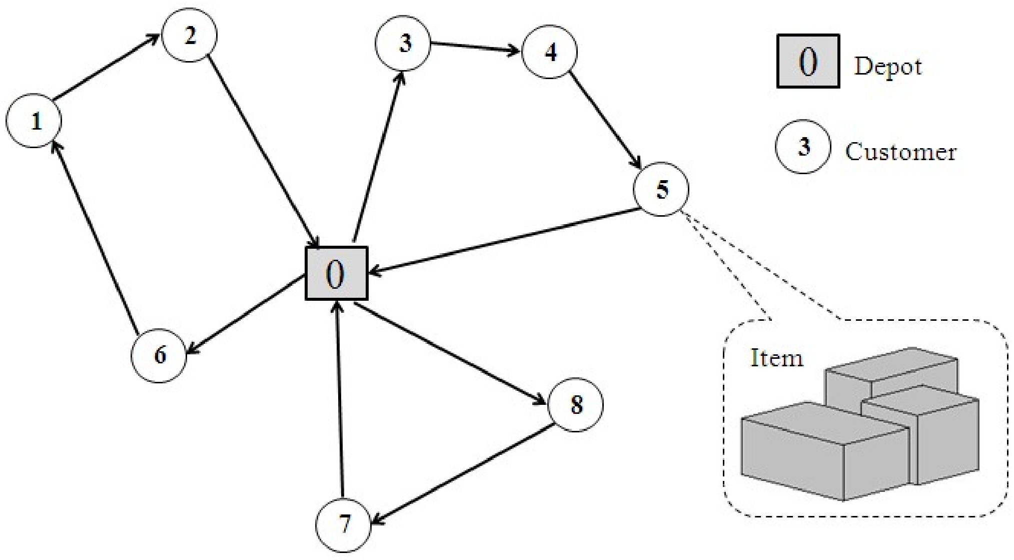


Figure 2: The 3L-CVRP.

problem has not explicitly taken the three-dimensional loading into account, except checking that, for each vehicle, the total weight of the loaded items does not exceed the given vehicle weight capacity.

The three-dimensional loading capacitated vehicle routing problem (3L-CVRP) is a highly complex problem combining the VRP and the CLP. The problem calls for the determination of the routes travelled by a vehicle fleet for delivering items to customers. Items consist of rectangular boxes of given size and weight, and must be feasibly loaded within the vehicles before they are shipped (Figure 2).

The 3L-CVRP is of both practical interest and theoretical interest. From the viewpoint of practical application, the 3L-CVRP is especially relevant for the cases that suppliers have to deal with large items and the loading aspect is not trivial, i.e., when one is distributing kitchen components, auto parts, mechanical components , and household appliances, the loading problem must be taken into account. Concerning theoretical importance, the 3L-CVRP is a very challenging problem for it generalizes two of the most well known NP-hard problems: the VRP and the CLP, which have been studied widely but independently.

The 3L-CVRP is first introduced by Gendreau M. et al. (2006). In their paper, a tabu search algorithm has been proposed that iteratively invokes an inner tabu search procedure for the solution of the loading subproblem. Other literatures about the 3L-CVRP includes: Aprile et al. (2007) proposed an simulate annealing algorithml; an integrated approach has been proposed by Moura (2009); and an ant colony algorithm has been proposed by Fuellerer (2010). However, no exact algorithm has been proposed for the 3L-CVRP.

In this paper, for solving the 3L-CVRP, both the single CLP algorithms and the MCLP algorithms are used. In addition, algorithms for VRP and TSP are also used. It is very time consuming to check whether the items can be feasibly loaded into a vehicle. Therefore the loading constraint is replaced with the constraint of volume ratio. Considering the issue of green logistics, we take the fuel consumption and CO2 emissions into account. As mentioned in the report of European Committee for Standardization, for a certain vehicle which travelling with a constant speed, the fuel consumption is approximately proportional to the travelling distance and linear correlate to the weight of loaded items. Furthermore, in the real-world instance, for improving the utilization of vehicles, some constraints for traditional VRP may be violated. Algorithm for the generalized 3L-CVRPs have also been proposed. The computational result for real-world instance shows that proposed algorithms are fast and high-quality for solving practical problems.

The rest of the paper is organized as follows. Chapter 2, 3, and 4 give formulation, algorithms and computational results for the SCLP, MCLP and 3L-CVRP, respectively. And Chapter 5 summarizes the paper.

Chapter 2

The Single Container Loading Problem

2.1 Problem Formulation of the SCLP

As shown in Figure 3, a three-dimensional coordinate system and the directions 'front', 'back', 'left', 'right', 'up' and 'down' are illustrated. The container is placed in the system with its back-left-down corner in the origin, and its length, width and height are parallel to the x-, y- and z-axes, respectively. Similar to many existing CLP algorithms (e.g., Bortfeldt and Gehring, 2001; Eley, 2002; Moura and Oliveira, 2005), we assume the following:

- Each item is placed completely within the container.
- Each item does not overlap with another item.
- Only orthogonal packing is considered, i.e., each items is placed parallel to the edges of the container.
- Item can be rotated. Consequently, up to six different orientations are allowed.

The six possible orientations of item are shown in Figure 4. An item with length l, width w and height h has six possible orientations whose dimensions on the x-, y-, and z-axes are $(l, w, h), (w, l, h), (l, h, w), (h, l, w), (h, w, l)$, and (w, h, l), respectively.

A solution of the SCLP consists of a set of packed items, i.e., items packed into the container. The following notations are used in the formulation.

Notation:

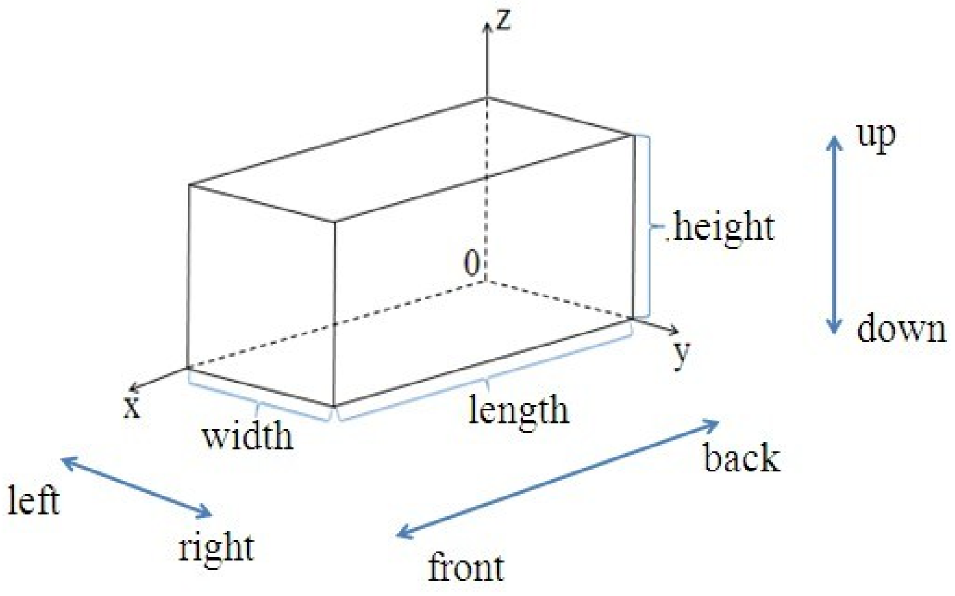


Figure 3: Container in Three-Dimensional Coordinate System.

(L, W, H)	length,width and height of the container
$I = \{1 \cdot n_1, \ldots, m \cdot n_m\}$	the set of items
$P = \{p_1, \ldots, p_n\} \subseteq I$	the set of packed items
(l_i, w_i, h_i)	length, width and height of p_i $(i = 1, \ldots, n)$
(x_i, y_i, z_i)	x, y and z coordinate of p_i $(i = 1, \ldots, n)$

To avoid repetition, the items of the same size and permitted orientations are categorized into a same type, and n_j is the available number of items of type j $(j = 1, \ldots, m)$. Therefore the set of items are denoted by a multiset I. The notion of multiset is a generalization of the notion of set in which elements are allowed to appear more than once. The operations and relations of multiset are similar to that of set. The interested reader is referred to Wayne (1989) and Knuth (2005). The set of packed items P is also a multiset, because different packed items may be of the same type. The coordinates (x_i, y_i, z_i) of p_i means that the back-left-down corner of p_i is positioned at the point (x_i, y_i, z_i).

The SCLP is formulated as follows:

[SCLP]

$$\begin{aligned}
\text{Maximize} \quad & \sum_{i=1}^{n} l_i w_i h_i, && (2.1)\\
\text{subject to} \quad & x_i + l_i \leq L, \forall i = 1, 2, \ldots, n, && (2.2)\\
& y_i + w_i \leq W, \forall i = 1, 2, \ldots, n, && (2.3)\\
& z_i + h_i \leq H, \forall i = 1, 2, \ldots, n, && (2.4)\\
& x_i, y_i, z_i \geq 0, \forall i = 1, 2, \ldots, n, && (2.5)\\
& x_i + l_i \leq x_j \text{ or } x_j + l_j \leq x_i \text{ or } y_i + w_i \leq y_j \text{ or} \\
& y_j + w_j \leq y_i \text{ or } z_i + h_i \leq z_j \text{ or } z_j + h_j \leq z_i,
\end{aligned}$$

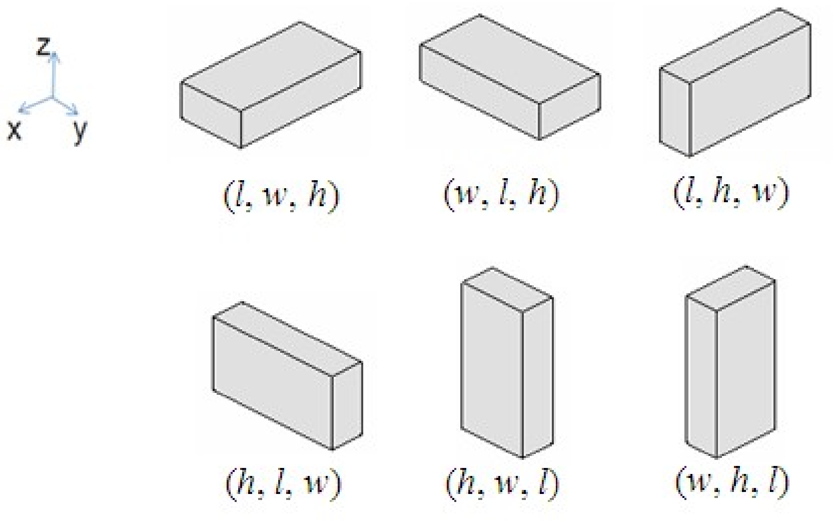


Figure 4: Item Orientations.

$$\forall i, j = 1, 2, \ldots, n, i \neq j, \tag{2.6}$$

The constraints (2.2)-(2.5) require that each packed item lies completely in the container. The constraint (2.6) prevents packed items from overlapping.

Definition 1 (*feasible packing*). A set of packed items $P = \{p_1, \ldots, p_n\}$ is called a feasible packing if it satisfies the constraints (2.2) - (2.6).

The rest of this chapter is organized as follows. Section 2.2 and 2.3 provide exact and heuristic algorithms for SCLP, respectively. Section 2.4 addresses a practical constraint – shipment priority. Section 2.5 is the computational experiments and Section 2.6 summarizes this chapter.

2.2 Exact algorithm for SCLP

In this section we propose a branch-and-bound algorithm (algorithm EXAC_SCLP, described later in Table 1) to find the optimal solution of the CLP, i.e., to find a feasible packing that achieves maximum utilization of the container. The operation of generating feasible packing may be drastically simplified through the following definitions and lemmas.

Definition 2 (*adjoin*). For $\forall p_i, p_j \in P$, we say that p_j x-adjoin p_i, if

$$x_i = x_j + l_j, \tag{2.7}$$
$$y_i < y_j + w_j, \tag{2.8}$$
$$z_i < z_j + h_j, \tag{2.9}$$
$$y_j < y_i + w_i, \tag{2.10}$$
$$z_j < z_i + h_i. \tag{2.11}$$

Analogously, we can define y- and z-adjoin.

Definition 3 (*staircase packing*). A feasible packing P is called a staircase packing, if

$$\forall p_i \in P \text{ and } x_i > 0, \exists p_j \in P, p_j \text{ x-adjoin } p_i, \tag{2.12}$$
$$\forall p_i \in P \text{ and } y_i > 0, \exists p_j \in P, p_j \text{ y-adjoin } p_i, \tag{2.13}$$
$$\forall p_i \in P \text{ and } z_i > 0, \exists p_j \in P, p_j \text{ z-adjoin } p_i, \tag{2.14}$$
$$\forall p_i, p_j \in P \text{ and } i > j, x_i \geq x_j + l_j \text{ or } y_i \geq y_j + w_j \text{ or } z_i \geq z_j + h_j. \tag{2.15}$$

Intuitively, the shape of a staircase packing looks like a staircase from a side view (Figure 5 (a)), and no packed-items can be moved leftward, downward, or backward. The idea of using staircase placements was first proposed in Martello et al. (2000) and applied for the two-dimensional strip packing problem by Kenmochi et al. (2009).

Lemma 1. Any feasible packing can be replaced by an equivalent staircase packing.

Lemma 1 has been proved by Martello et al. (2000) by (i) moving the packed items leftward, downward, or backward (to satisfy constraints (2.12) - (2.14)) and (ii) reordering the packed items (to satisfy constraint (2.15)).

However it is still difficult to directly generate staircase packing. Thus we propose the notions of *pseudo-adjoin* and *pseudo-staircase packing* as follows.

Definition 4 (*pseudo-adjoin*). For $\forall p_i, p_j \in P$, we say that p_j x-pseudo-adjoin p_i, if the constraints (7) - (9) are satisfied. Analogously, we can define y- and z-pseudo-adjoin.

Definition 5 (*pseudo-staircase packing*). A feasible packing P is called a pseudo-staircase packing, if

$$\forall p_i \in P \text{ and } x_i > 0, \exists p_j \in P, p_j \text{ x-pseudo-adjoin } p_i, \tag{2.16}$$
$$\forall p_i \in P \text{ and } y_i > 0, \exists p_j \in P, p_j \text{ y-pseudo-adjoin } p_i, \tag{2.17}$$
$$\forall p_i \in P \text{ and } z_i > 0, \exists p_j \in P, p_j \text{ z-pseudo-adjoin } p_i, \tag{2.18}$$

and (2.15).

As shown in Figure 5, if p_1, p_2 and p_3 have the same y-coordinate, then both (a) and (b) are pseudo-staircase packing. However, (b) is not a staircase packing, because p_2 z-pseudo-adjoin p_3 but not z-adjoin p_3.

Definition 6 (*possible points*). For a pseudo-staircase packing $P_k = \{p_1, \ldots, p_k\}$, a possible point is a position (x, y, z) such that there exists a residual item p_{k+1} (i.e., $p_{k+1} \in I \setminus P_k$), and if p_{k+1} is packed at (x, y, z), $P_{k+1} = \{p_1, \ldots, p_{k+1}\}$ is still a pseudo-staircase packing. The set of all of the possible points of P_k is denoted by F_k. Specifically for $P_0 = \emptyset$, let $F_0 = \{(0, 0, 0)\}$.

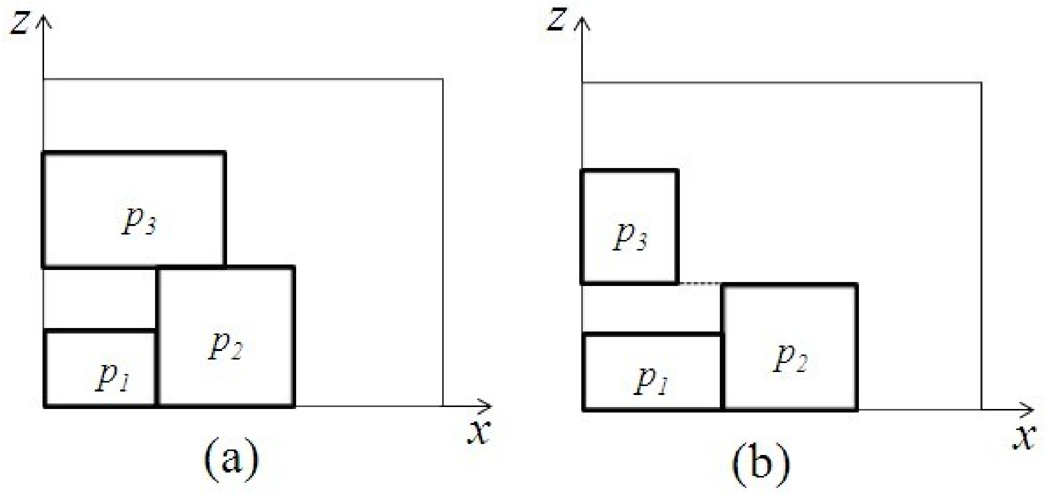


Figure 5: Staircase Packing and Pseudo-Staircase Packing (perspective: side view).

Obviously, any staircase packing is also a pseudo-staircase packing. And if $\{p_1, \ldots, p_n\}$ is a pseudo-staircase packing, its subset $P_k = \{p_1, \ldots, p_k\}$ is also a pseudo-staircase packing ($k = 1, \ldots, n$). In algorithm EXAC_SCLP, different kinds of pseudo-staircase packing are generated in a branch-decision tree. At each node of the tree, a current partial solution, which packs the items of a certain subset of I, is increased by selecting in turn each residual item and generating child nodes by placing the item into all the possible points.

First at all, we use a recursion method (called algorithm POSSIPOIONTS, described later) to generate the possible points, i.e., given a set of existing possible points F_k and an additional item p_{k+1}, to generate the set of new possible points F_{k+1}. The following lemmas and corollary are proposed for this purpose.

Lemma 2. For a pseudo-staircase packing $P_{k+2} = \{p_1, \ldots, p_k, p_{k+1}, p_{k+2}\}$, we have the following:

(i) If p_{k+1} doesn't pseudo-adjoin p_{k+2}, then $(x_{k+2}, y_{k+2}, z_{k+2}) \in F_k$;

(ii) If p_{k+1} x-pseudo-adjoin p_{k+2}, then $\exists (x, y, z) \in F_k, y = y_{k+2}, z = z_{k+2}$;

(iii) If p_{k+1} y-pseudo-adjoin p_{k+2}, then $\exists (x, y, z) \in F_k, x = x_{k+2}, z = z_{k+2}$;

(iv) If p_{k+1} z-pseudo-adjoin p_{k+2}, then $\exists (x, y, z) \in F_k, x = x_{k+2}, y = y_{k+2}$.

Proof.

(i) If p_{k+1} doesn't pseudo-adjoin p_{k+2}, then delete p_{k+1} from P_{k+2}. By Definition 5, the set $\{p_1, \ldots, p_k, p_{k+2}\}$ is also a pseudo-staircase packing. Then by Definition 6, $(x_{k+2}, y_{k+2}, z_{k+2}) \in F_k$.

(ii) If p_{k+1} x-pseudo-adjoin p_{k+2}, then delete p_{k+1} from P_{k+2}, and move p_{k+2} backwards, which makes $\{p_1, \ldots, p_k, p_{k+2}\}$ become a pseudo-staircase packing.

In detail, let $P^* = \{p_i \in P_k : y_i + w_i > y_{k+2}, z_i + h_i > z_{k+2}\}$,

$$x^* = \begin{cases} \max_{p_i \in P^*}(x_i + l_i), & P^* \neq \emptyset, \\ 0, & P^* = \emptyset, \end{cases}$$

and move p_{k+2} to the position (x^*, y_{k+2}, z_{k+2}). By Definition 5, the set $\{p_1, \ldots, p_k, p_{k+2}\}$ is also a pseudo-staircase packing. Then by Definition 6, $(x^*, y_{k+2}, z_{k+2}) \in F_k$.

(iii) and (iv) are similar to (ii). ∎

By Lemma 2, we can easily induce the following corollary.

Corollary 1. Each point in F_{k+1} belongs to F_k, or belongs to the (orthographic) projection of F_k on plane $x = x_{k+1} + l_{k+1}$, $y = y_{k+1} + w_{k+1}$ or $z = z_{k+1} + h_{k+1}$.

Definition 7 (*contain*). For any point (x, y, z) in the container, and item $r \in I$, if there exists a permitted orientation of r whose dimensions on the x-, y- and z-axis are (l', w', h'), and $l' \leq L - x, w' \leq W - y, h' \leq H - z$, we say that (x, y, z) contains item r.

Lemma 3. Let $A = \{(x, y, z) \in F_k : x \leq x_{k+1} + l_{k+1}, y \leq y_{k+1} + w_{k+1}, z \leq z_{k+1} + h_{k+1}\}, A_1 = \{(x, y, z) \in A :$ for $\forall (x', y', z') \in A$ and $(x, y, z) \neq (x', y', z')$, $y' > y$ or $z' > z\}$, we have the following:

(i) For $\forall (x, y, z) \in F_k \backslash A$, if $\exists r \in I \backslash P_{k+1}, (x, y, z)$ contains r, then $(x, y, z) \in F_{k+1}$;

(ii) For $\forall (x, y, z) \in A_1$, if $\exists r \in I \backslash P_{k+1}, (x, y, z)$ contains r, then $(x_{k+1} + l_{k+1}, y, z) \in F_{k+1}$;

Proof.

(i) is easy to prove by Definition 5.

(ii) Proof by contradiction. Assume that $(x_{k+1} + l_{k+1}, y, z) \notin F_{k+1}$. Pack r on $(x_{k+1} + l_{k+1}, y, z)$. Similar to the proof of Lemma 2 (ii), move r leftward and downward to make $\{p_1, \ldots, p_k, p_{k+1}, r\}$ become a pseudo-staircase packing. Then delete p_{k+1} from P_{k+1} and move r backward to make $\{p_1, \ldots, p_k, r\}$ become a pseudo-staircase packing. Assume the new coordinate of r is (x', y', z'). Obviously $(x', y', z') \in A, (x, y, z) \neq (x', y', z'), y' \leq y, z' \leq z$, which conflicts with that $(x, y, z) \in A_1$. ∎

As shown in Corollary 1, all of the points of F_{k+1} belong to F_k or the projection of F_k. By Lemma 3 (i), we can search for the points of F_{k+1} in F_k, and by Lemma 3 (ii), we can search for the points of F_{k+1} in the projection of F_k on the plane $x = x_{k+1} + l_{k+1}$. Analogously, we can search for the points of F_{k+1} in the projection of F_k on the plane $y = y_{k+1} + w_{k+1}$ or $z = z_{k+1} + h_{k+1}$. It is not difficult to prove that no other points belong to F_{k+1}. Therefore all the points of F_{k+1} can be generated. Algorithm POSSIPOIONTS is described as follows:

[Algorithm POSSIPOIONTS]

Input: F_k, p_{k+1}.

Output: F_{k+1}.

- **Step 1**: Search for possible points in $F_k \backslash A$ (Lemma 3 (i)).
- **Step 2**: Search for possible points in the projection of A_1 on plane $x = x_{k+1} + l_{k+1}$ (Lemma 3 (ii)).
- **Step 3**: Similar to Step 2. Select possible points in the projection on plane $y = y_{k+1} + w_{k+1}$ and on plane $z = z_{k+1} + h_{k+1}$.

In Step 1, the the computational complexity of generating $F_k \backslash A$ is $O(|F_k|)$. In Step 2, the computational complexity of generating A_1 is $O(|F_k|)$, plus $O(|F_k| \log |F_k|)$ for the initial sorting of possible points. Therefore if we don't consider whether a point can contain an item (in fact it can be tested in the following algorithm EXAC_SCLP), the computational complexity of algorithm POSSIPOIONTS is $O(|F_k|) + 3(O(|F_k|) + O(|F_k| \log |F_k|)) = O(|F_k| \log |F_k|$. In Martello et al. (2000), the computational complexity of the same-function algorithm is $O(m^2)$ (m is the number of items). In computational experiments, algorithm POSSIPOIONTS is much faster than Martello's algorithm.

In algorithm EXAC_SCLP, each node denote a partial solution, which includes information on possible points and residual items. Especially, the root node denotes an empty container, which has only one possible point (0,0,0) and the residual items is I. Child nodes are generated by packing each residual item with each of its permitted orientation at each possible point. Similar to Martello et al. (2000), we can calculate the upper bound of each node. The lower bound of each node can be calculated by using the heuristic algorithm (mentioned in the next section). The maximum value of the lower bounds serves as the current solution. If the upper bound of a node is less than the current solution, the node is killed. Algorithm EXAC_SCLP is described by the pseudo-code in Table 1.

2.3 Heuristic Algorithm for SCLP

This heuristic algorithm is a tree search heuristics, which hierarchically consists of a subordinated and a superior module. The subordinated module is a greedy heuristic, which serves the complete loading of the container. The superior module is a tree search which improves the solution generated by the greedy heuristic.

The proposed greedy heuristic algorithm (algorithm HEURI_SCLP) is similar to other existing block building approaches (Eley, 2002; Parreto et al., 2008). The container is filled with homogeneous blocks. Each block is composed of identical items that have the same orientation. As is usual in block-building approaches, each feasible placement position where a block may be packed is called a(n) (empty) space. Beginning with the entire empty container, which is initialized as the first space, new spaces are generated when a block is packed in a space. The main difference between the proposed heuris-

Table 1: Algorithm EXAC_SCLP

```
**Input**: items, container.
**Output**: $MaxU$ (maximum utilization of container).
initialize $MaxU = 0$;
sort the item types in noincreasing order of item volume;
initialize $Sol := \{(F_0 = \{(0,0,0)\}, I)\}$;// the set of all partial solutions
while $Sol \neq \emptyset$
    $tempSol := \emptyset$;//temporary set of partial solutions
    for all solutions in $Sol$ do
        for all possible points in current solution do
            for all residual items and their permitted orientations do
                generated possible points by using algorithm POSSIPOIONTS;
                if the current possible point can contain the current item then
                    generate a new solution by packing current item in current
                    possible point;
                endif
                calculate the upper bound and lower bound of the new solution;
                if the upper bound is less than $MaxU$ then
                    continue;
                endif
                add the new solution to $tempSol$;
                if the lower bound $> MaxU$ then
                    $MaxU$ = container utilization of the new solution;
                      endif
             if the current possible point cannot contain any item do
                 delete it from the set of possible points;
             endif
            endfor
        endfor
    endfor
    $Sol := tempSol$;
endwhile
```

tic and other existing approaches lies in the evaluation functions for selecting blocks. Five evaluations are defined for block selection. Before we present a description of the entire procedure of the heuristic, the following highlights the strategies of block building and block evaluation.

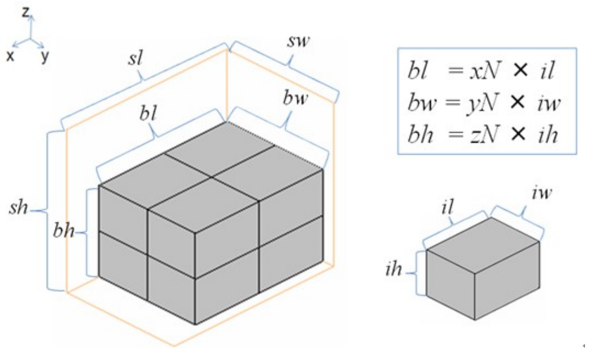


Figure 6: Block in Space.

2.3.1 Block Building

As shown in Figure 6, let sl, sw, and sh be the length, width, and height, respectively, of a space S. For a block that is composed of items of a specific type and orientation status, let il, iw, and ih be the dimensions of the items on the x-, y-, and z-axes, respectively; N be the residual number of the items of the given type; xN, yN, and zN be the number of items along the x-, y-, and z-axes in the block, respectively; and bl, bw, and bh be the length, width, and height of the block, respectively.

A block can be packed into S if the following constraints are satisfied:

$$xN \times yN \times zN \leq N \tag{2.19}$$

$$bl = xN \times il \leq sl \tag{2.20}$$

$$bw = yN \times iw \leq sw \tag{2.21}$$

$$bh = zN \times ih \leq sh \tag{2.22}$$

Example 2.1 (*possible blocks*): Assume that the three dimensions of the space S are $sl \times sw \times sh = 370 \times 250 \times 220$ along the x-, y-, and z-axes, respectively. There are two types of items: I_1 and I_2. Both item types can only be horizontally rotated, i.e., only the orientations "(l, w, h)" and "(w, l, h)" are permitted.

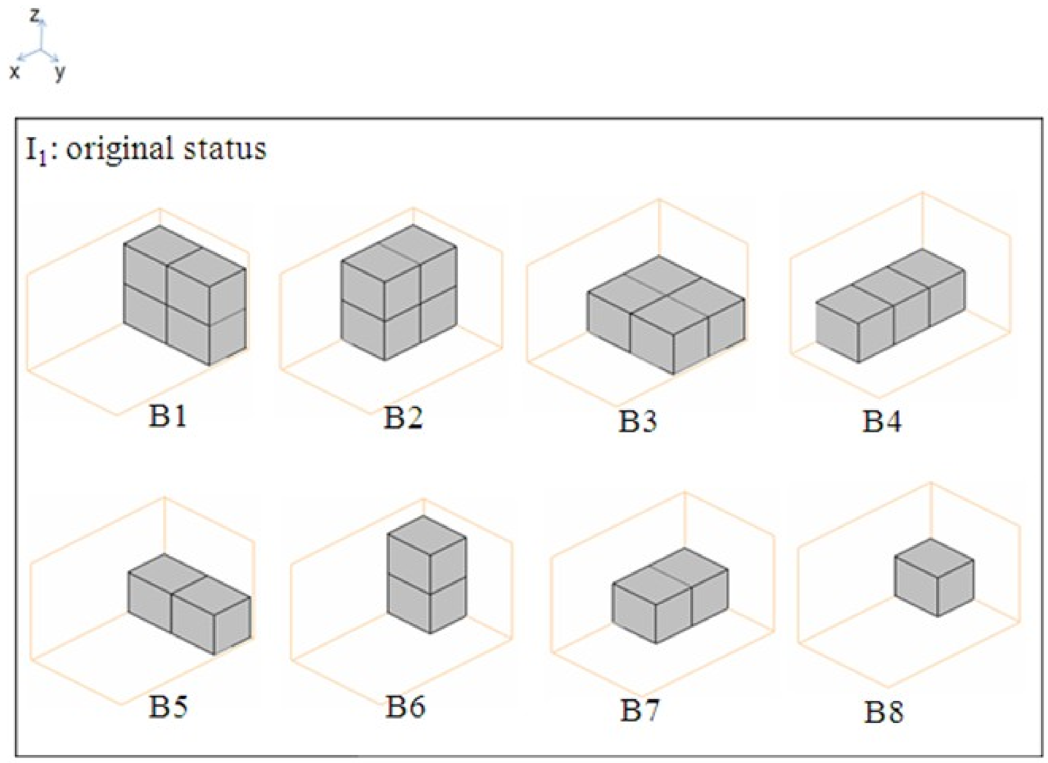
The length, width and height of I_1 are 100, 120, and 90, respectively.

The length, width and height of I_2 are 240, 45, and 200, respectively.

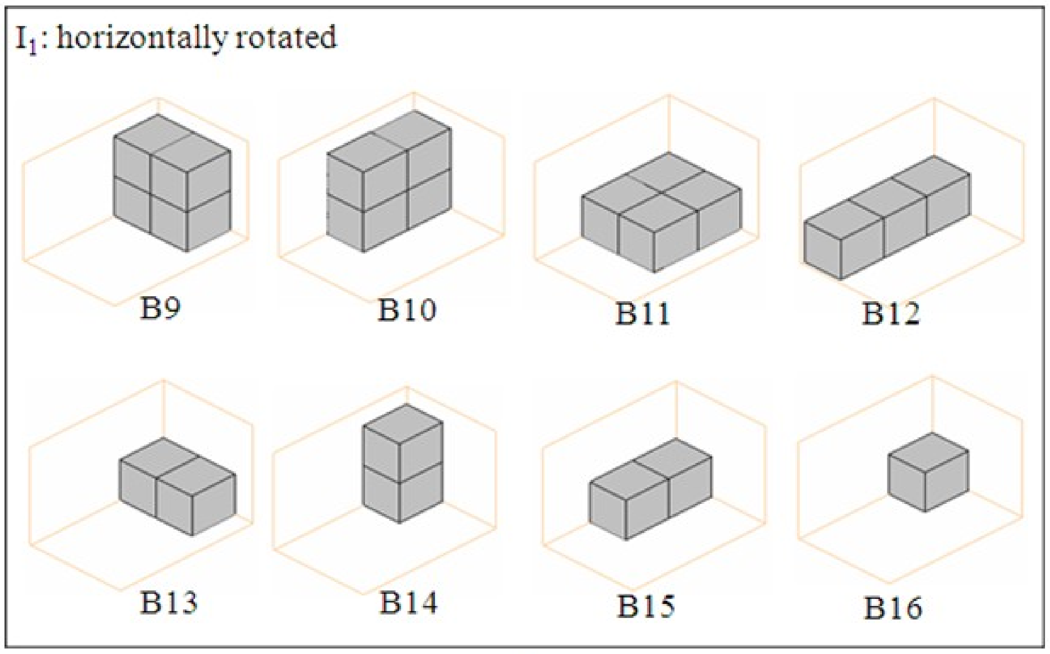
The residual numbers of I_1 and I_2 are 4 and 2, respectively.

Figure 7 shows all the possible blocks (B1 - B20) that satisfy constraints (2.19) - (2.22). Note that the list of blocks is sorted according to the following criteria:

- Main criterion: The order of item types (I_1, I_2).
- Tie-breaker 1: The order of orientations (from (l, w, h) to (w, h, l)).
- Tie-breaker 2: Decreasing order of bw.
- Tie-breaker 3: Decreasing order of bh.


z
x
y
I1: original status
B1
B2
B3
B4
B5
B6
B7
B8

(a)


I1: horizontally rotated
B9
B10
B11
B12
B13
B14
B15
B16

(b)

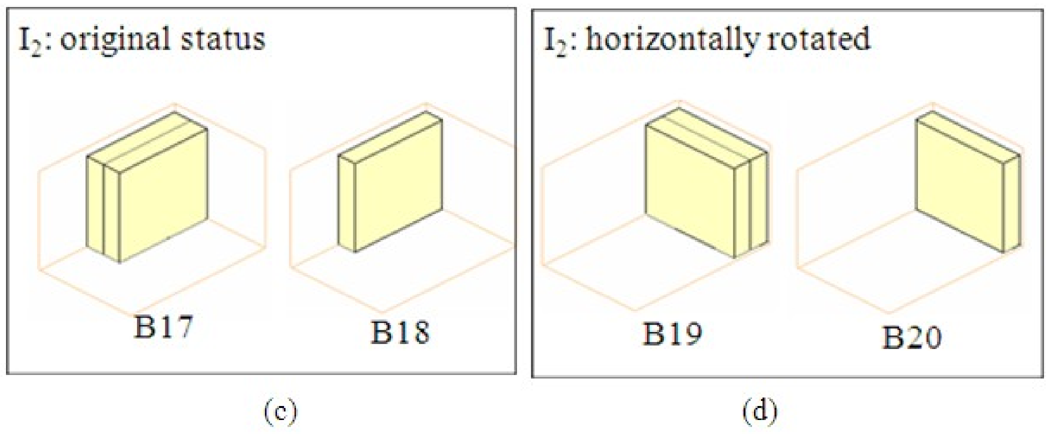


Figure 7: Possible Blocks.

- Tie-breaker 4: Decreasing order of *bl*.

Blocks B1 - B8 are made up of I_1, which is in its original status, i.e., $il = 100, iw = 120, ih = 90$ (Figure 7 (a)).

Blocks B9 - B16 are made up of I_1, which is horizontally rotated, i.e., $il = 120, iw = 100, ih = 90$ (Figure 7 (b)).

Blocks B17 and B18 are made up of I_2, which is in its original status, i.e., $il = 240, iw = 45, ih = 200$ (Figure 7 (c)).

Blocks B19 and B20 are made up of I_2, which is horizontally rotated, i.e., $il = 45, iw = 240, ih = 200$ (Figure 7 (d)).

Therefore, for items of the same type with identical orientations, multiple types of blocks can be built by changing the item numbers along the x-, y-, and z-axes. Considering all of the item types and permissible orientations, a large number of blocks can be built. It is crucial to evaluate all of the possible blocks and to select a block for packing into the space.

2.3.2 Block Evaluation

Five evaluation functions for block selection are defined as follows:

- (E1) $-\min((sl - bl), (sw - bw), (sh - bh))$,
- (E2) $bw \times bh$,
- (E3) $bl \times bh$,
- (E4) $bl \times bw$,
- (E5) $bl \times bw \times bh$,

All of the above evaluations are greedy criteria and should be maximized. E1 evaluates the utilization of one dimension (i.e., the length, width or height of the space), E2 - E4 evaluate the utilization of two dimensions (i.e., the YZ-Area, XZ-Area or XY-Area of the space), and E5 evaluates the utilization of

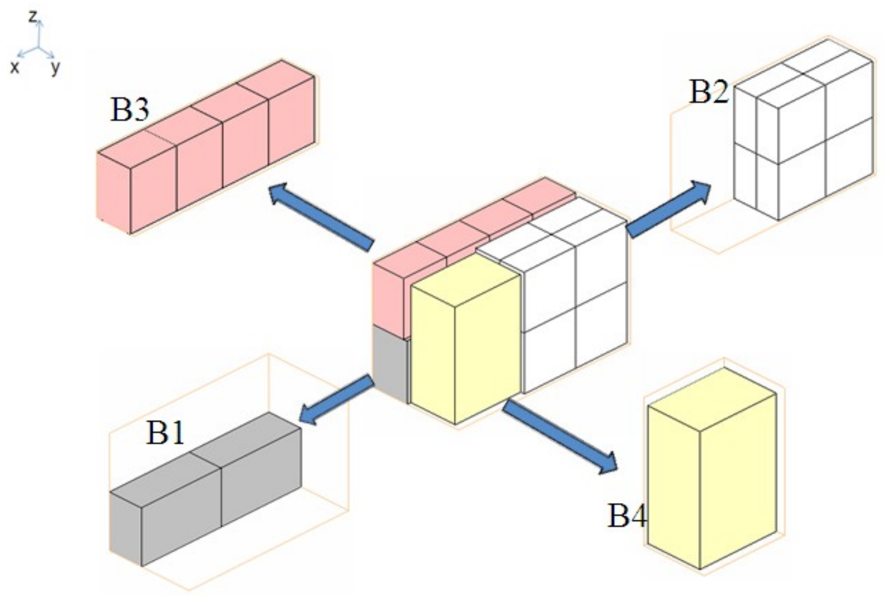


Figure 8: Evaluations of Block.

three dimensions (i.e., the volume of the space). A number of experiments have shown that no single evaluation is efficient for all situations, and the five evaluation alternative is much more efficient. The frequencies of the five evaluations in the numerical experiments are shown in Section 2.5.2.2.

Example 2.2 (*Evaluations of Block*): As shown in Figure 8, a container is filled with a sequence of blocks (B1, B2, B3 and B4). First, to pack block B1, it is arranged along the x-axis to make the best use of the length; then, to pack block B2, it is arranged along the y- and z-axes to make the best use of the YZ-area; next, to pack block B3, it is arranged along x- and z-axes to make the best use of the XZ-area; finally, block B4 is packed so as to make the best use of the three-dimensional space. Coincidences still exist; e.g., B4 also makes good use of one dimension or two dimensions.

The five evaluations can be applied in either a simultaneous manner or a sequential manner, as described below.

Simultaneous manner: A block with the highest value for one of the five evaluations is called a feasible block, and it may be selected for packing into the space.

Sequential manner: A block is selected according to the following criteria:

- Main criterion: Largest value of E1.
- Tie-breaker 1: Largest value of E2.
- Tie-breaker 2: Largest value of E3.
- Tie-breaker 3: Largest value of E4.
- Tie-breaker 4: The top block satisfying Tie-breaker 3 according to the order criteria of blocks.

If multiple blocks satisfy the above criteria, only the first one (according to position in the list of possible blocks) among them is selected. Evaluation E5 is not mentioned here because if two blocks achieve identical values for E2, E3 and E4, respectively, they must also achieve an identical value for E5.

Example 2.3 (*simultaneous manner*): Consider the data shown in Example 2.1. As shown in Figure 7, for example, for block B1, evaluation E1$= -\min((sl-bl), (sw-bw), (sh-bh)) = -\min((sl-xN\times il), (sw-yN\times iw), (sh-zN\times ih)) = -\min((370-1\times 100), (250-2\times 120), (220-2\times 90)) = -10$. In fact, no other block achieves a larger value for E1. Therefore, -10 is the largest value for E1.

B1, B3, B5, B12, B19 and B20 all achieve the largest value for E1.

B19 and B20 achieve the largest value for E2.

B17 and B18 achieve the largest value for E3.

B3 and B11 achieve the largest value for E4.

B1, B2, B3, B9, B10, B11, B17 and B19 achieve the largest value for E5.

Delete the identical ones. Therefore B1, B2, B3, B5, B9, B10, B11, B12, B17, B18, B19 and B20 are all selected as feasible blocks.

Example 2.4 (*sequential manner*): Consider the data shown in Example 2.1. Using the main criterion, blocks B1, B3, B5, B12, B19 and B20 are selected because all of these blocks achieve the largest value for E1. Then, using E2 as the first tie-breaker, the list can be narrowed down to blocks B19 and B20, all of which achieve the same largest value for E2. Using E3 as the second tie-breaker then narrows the list to only block B19 because it achieves the largest value for E3.

In a simultaneous manner, usually multiple blocks can be selected as feasible blocks. However, in the sequential manner, only one block is selected for packing into the space. The simultaneous manner is used in the tree search, which will be described in Section 2.3.5, and the sequential manner is used in the greedy heuristic shown in Section 2.3.3.

2.3.3 Procedure of the Greedy Heuristic

The whole procedure of the heuristic algorithm for the SCLP (HEURI_SCLP) is described as follows (Figure 9):

[Algorithm HEUR_SCLP]

Input: items, container.

Output: packed container.

- **Step 0** (*Initialize*): The current space $S :=$ whole empty container;.
- **Step 1** (*Select block*): If there is no residual item, stop. Otherwise, build all of the possible blocks (i.e., blocks that satisfy constraints (2.19)

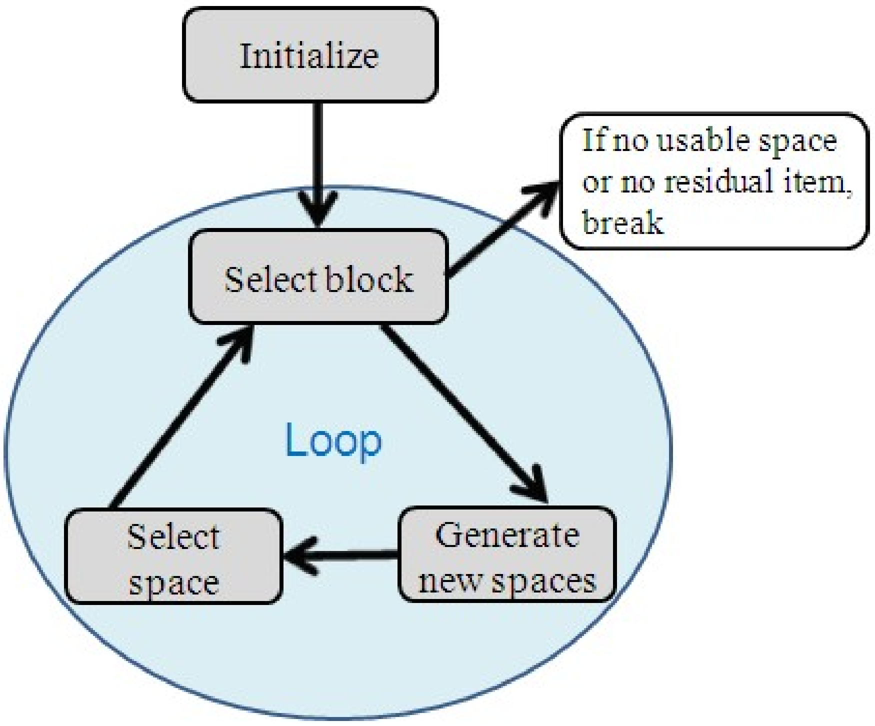


Figure 9: Algorithm HEUR_SCLP.

- (2.22)) in the current space S. Then, from the list of possible blocks, select a feasible block by using the five evaluations E1 - E5 (in the sequential manner described before) and pack it into S. A detailed explanation of this step has been provided in Sections 2.3.4 and 2.3.2.

- **Step 2** (*Generate new spaces*): Split the residual space of S into new spaces, and add them into $sList$. Delete S from $sList$. Merge any two contiguous spaces in $sList$ if they satisfy certain conditions. Mark any space in $sList$ as "unusable" if no residual item can be packed into it. This procedure will be described in more detail in Section 2.3.4, which discusses space splitting and merging.

- **Step 3** (*Select space*): If all of the spaces in $sList$ are marked as 'unusable', stop. Otherwise select the usable space (i.e., the space not marked as 'unusable') that has the smallest Euclidean distance between its back-left-down corner and the origin. Assign the selected space to S and go to Step 1.

The entire process from Step 1 to Step 3 is called an iteration of the heuristic. In each iteration, a block is selected and packed into the container. The iteration is repeated until either all items have been packed or no more items can be packed into the container.

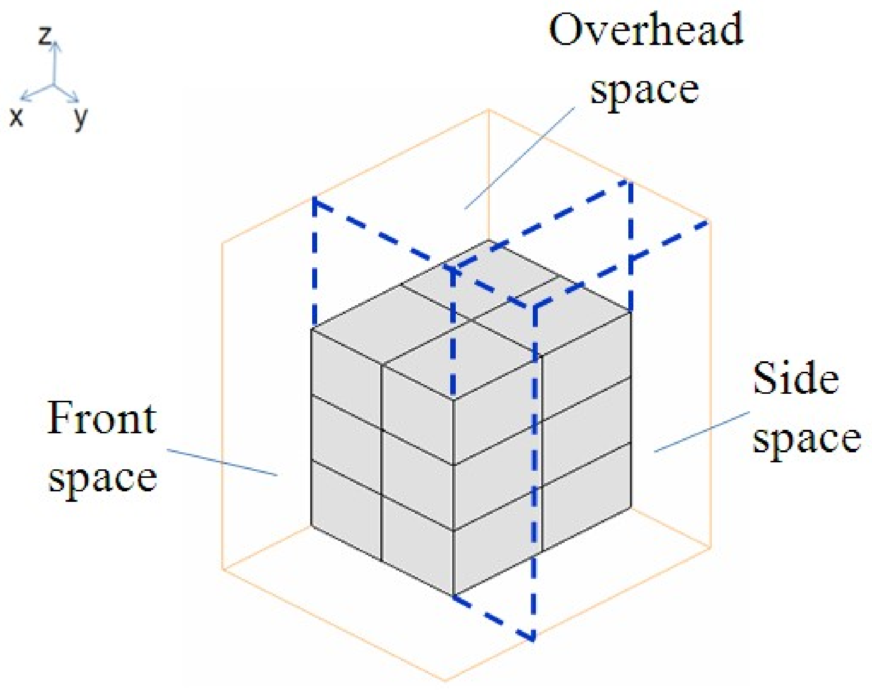


Figure 10: Split Space.

2.3.4 Space Splitting and Merging

The residual space of S can be divided into three sub-spaces (i.e., the front space, the side space, and the overhead space shown in Figure 10). Check each of the three newly generated spaces, and if there is no any of the residual items that can be packed in the space, mark it as ”unusable”. Then add the three newly generated spaces into $sList$ and delete S from $sList$.

Any two adjoining spaces in the space list that have the same z-coordinate (for their back-left-down corner) can be merged if one of the following conditions is satisfied:

(i) The two spaces have a common edge along the x-axis or y-axis as shown in Figure 11 (a).

(ii) The two spaces do not have a common edge along the x-axis or y-axis, but both are marked as ”unusable” as shown in Figure 11 (b).

In both of these conditions, the merged space replaces the two original spaces in the space list. If the merged space is too small to contain any of the residual items, it is marked as ‘unusable’. Furthermore, in condition (ii), the contiguous edges of two spaces do not completely coincide with each other. Therefore, other new spaces are generated. They are marked as ’unusable’ and added into the space list. The advantage of condition (ii) is making original spaces, which cannot be filled before be reused by merging them. Other new generated spaces are also marked as ‘unusable’. The method of merging spaces has also been proposed by other authors (Bortfeldt and Gehring, 2001; Eley, 2002). However, Eley only considered the merging in condition (i), and Bortfeldt and Gehring only considered the merging of the front space and side

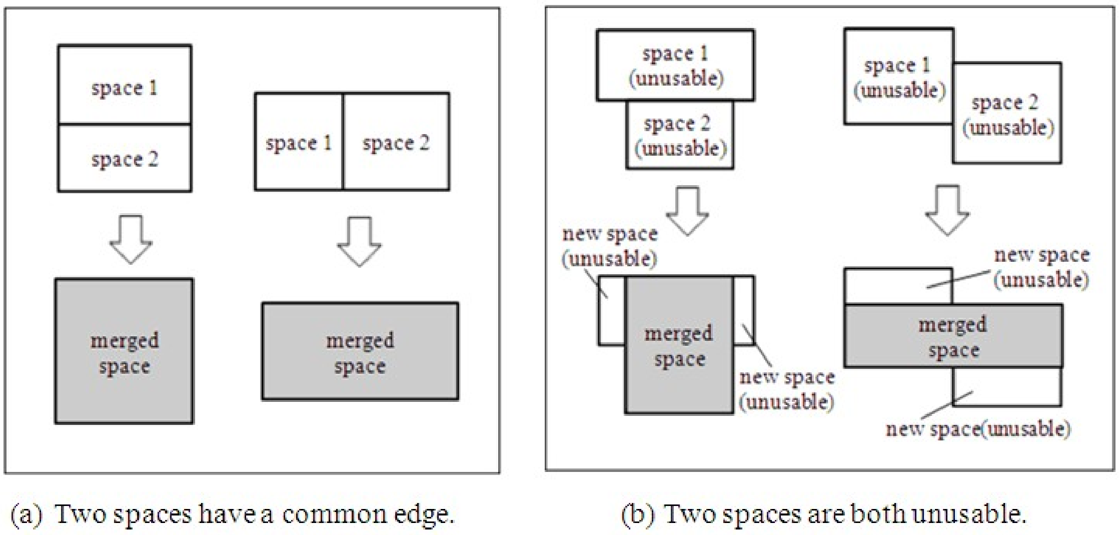


Figure 11: Merge spaces (perspective: top view).

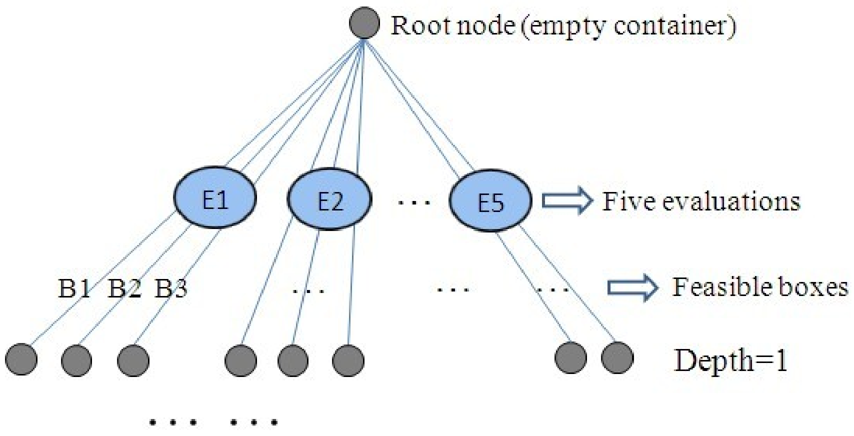


Figure 12: Tree search for SCLP.

space, which are sub-spaces that are generated from the same space.

2.3.5 Tree Search

To improve the solution generated by algorithm HEURI_SCLP, a tree search method called TRS_SCLP is implemented, which allows the selection of different blocks for each space.

As mentioned in Section 2.3.2, when using evaluations E1 - E5 in a sequential manner, only one block is selected in each iteration of algorithm HEURI_SCLP. However, in the tree search, all five evaluations are applied in a simultaneous manner. Usually multiple feasible blocks are selected, and these blocks constitute the branches of the search tree.

The root node represents an empty container, and each node of the search tree represents a partially filled container (i.e., a partial solution). Each node is branched into some sub-nodes according to all of the feasible blocks that achieve the highest value for one of the five evaluations. For example, as shown in Figure 12, the first three sub-nodes are generated according to the blocks B1, B2 and B3 that are selected using evaluation E1, and other sub-nodes are generated according to other blocks that are selected using E2, E3, E4, or E5. The set of all partial solutions at the same depth in the search tree is called a partial solution list.

Considering the exponentially increasing number of nodes, only a fixed number (parameter *breadth*) of nodes are retained from each depth. Similar to Eley (2002), a best search strategy is applied, which selects the *breadth* of nodes that obtained the highest ranking from an evaluation function. This function should not only consider the volume utilization obtained thus far, but it should also evaluate the potential for filling the residual spaces with the residual items. The function is a lower bound (for the volume utilization of the container) derived by filling the residual spaces of the corresponding partial solution by applying the greedy heuristic (HEURI_SCLP). Furthermore, to avoid one good solution replacing all other good solutions, we use the following criterion: if multiple nodes at the same depth in the tree achieve the same evaluation function value, only the nodes among them with the largest volume utilization obtained thus far (not including the items packed using the greedy heuristic) are retained, and the remaining nodes are deleted.

The tree search is described by the pseudo-code shown in Table 2.

2.4 Shipment Priority

Additional constraints can be added to the general SCLP (i.e., SCLP in basic form) to take into account some aspects of real-life problems. These constraints include load bearing strength, multi-drop, load stability, shipment priority, weight distribution and so on (Bischoff and Ratcliff, 1995). In this paper the single container loading problem with the constraint of shipment priority (SCLPSP) is addressed. The constraint of shipment priority can be described as follows: some items have higher priority over others; an item with low priority should not be packed into the container if it leads to high-priority items being left behind. In some practical situations, the shipment of some items may be more important than that of others, e.g., because of the delivery deadlines or the shelf life of the product concerned. In such situations, the prior loading of the important items must be taken into account.

In recent literature, increased attention has been focused on the SCLP with additional constraints. For instance, Davies and Bischoff (1999) and Eley (2002) took into account the weight distribution within a container. Bischoff (2006) considered the load bearing strength. Christensen and Rousøe (2009) addressed the container loading problem with multi-drop constraints.

However, no single study has addressed the SCLPSP. Bischoff and Ratcliff (1995) suggested that the priorities for shipment can be viewed in a knapsack model simply as objective function coefficients which define or adjust the value

Table 2: Algorithm TRS_SCLP

```
Input: items, container.
Output: $MaxU$ (maximum utilization of container).
initialize $MaxU = 0$;
initialize $Sol$ :={empty container};// the set of all partial solutions
while $Sol \neq \emptyset$
    for all solutions in $Sol$ do
        for all feasible blocks that achieve the largest value for one of the
        five evaluations (E1 - E5) do
            add current feasible block to current partial solution to generate
            a new partial solution;
            add the new partial solution to $Sol$;
        endfor
        if no item could be packed into the container and the volume
        utilization of current partial solution is higher than $MaxU$ then
            $MaxU$ := volume utilization of the current solution;
        endif
        delete current partial solution from $Sol$;
    endfor
    delete the partial solutions that have identical evaluation function value;
    select breadth best partial solutions from $Sol$;
endwhile
```

ratings of the items. However, they did not propose any suitable algorithm based on this idea.

The authors have had several years of experience in the development of practical container loading systems. Shipment priority has been considered in most of these practical systems. A practical case is mentioned here to illustrate the background of the SCLPSP. As shown in Figure 14, the items are required to be packed in containers and then transported by vehicles from a depot to a customer. Assume the earliest arriving time of the vehicles is set to be $t1$ and the second earliest arriving time is set to be t_2 ($t_1 < t_2$). When we consider the loading of the earliest arriving vehicle, the items can be divided into two groups according to their deadlines (i.e., the latest allowed arrival time for the items). The items whose deadlines $< t_2$ are classified into the high-priority group (group A), and those whose deadlines $\geq t_2$ are classified into the low-priority group (group B). Assume that the high-priority items can be fully loaded in the container but that the capacity of the container is not sufficient to load all of the items. Along with the requirement of maximal loading, it is a reasonable requirement that none of the high-priority items be left behind.

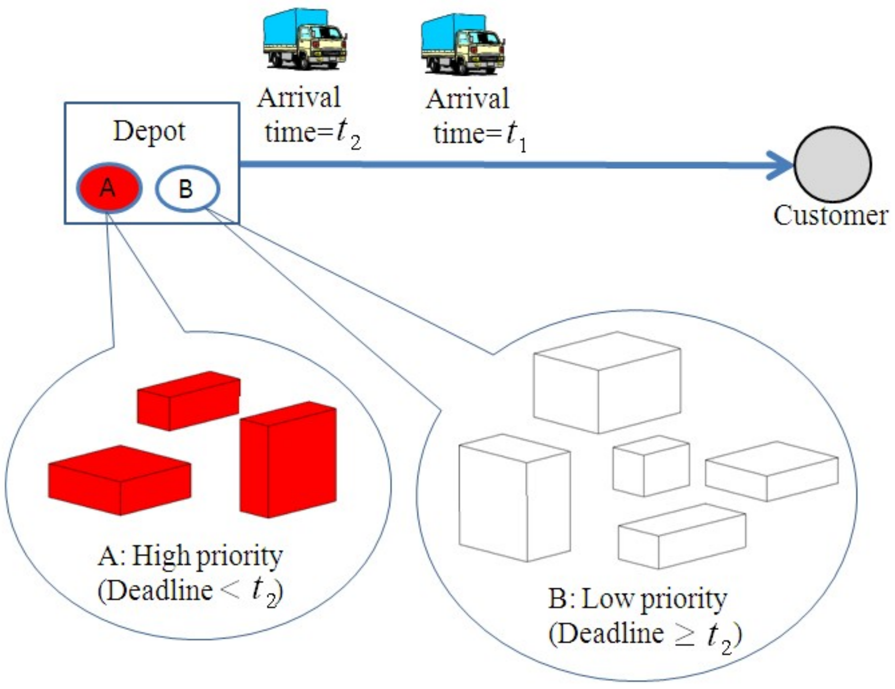

Arrival time=t_2
Arrival time=t_1
Depot
A
B
Customer
A: High priority (Deadline < t_2)
B: Low priority (Deadline ≥ t_2)

Figure 13: Background of SCLPSP.

There are also other applications of shipment priority, such as applications for MCLP and 3L-CVRP, which will be mentioned in the following chapters.

An algorithm for solving the general SCLP cannot be applied directly for the above case, though it may provide a high volume utilization of the container. However, such an algorithm cannot guarantee that all of the high-priority items will be loaded into the container (Figure 14(a)). Sequential loading (first loading the high-priority items, then loading the low-priority items) can require the high-priority items to be loaded into the container, but it is difficult to obtain both high volume utilization and packing stability because of the strict packing order (Figure 14(b)).

Note that the SCLPSP considers that all items in the same container will be unloaded at the same destination. It does not consider the positions of items in the same container, i.e., the high-priority items can be stowed anywhere inside the container (Figure 14(c)). Appropriate allocation of the items is determined by the volume utilization of the container and the priorities of the items.

The TRS_SCLP algorithm can be generalized and adapted to solve the SCLPSP when the constraint of shipment priority is considered. The generalized tree search algorithm is called TRS_SCLPSP.

Five alternative evaluations (E1 - E5) are defined in algorithm HEURI_SCLP for block selection. Similarly, the following two criteria are proposed for the proper allocation of the high- and low-priority items:

(P1): Select a block composed of high-priority items.

(P2): Select a block composed of low-priority items.

These two criteria can be combined with evaluations E1 - E5 to constitute

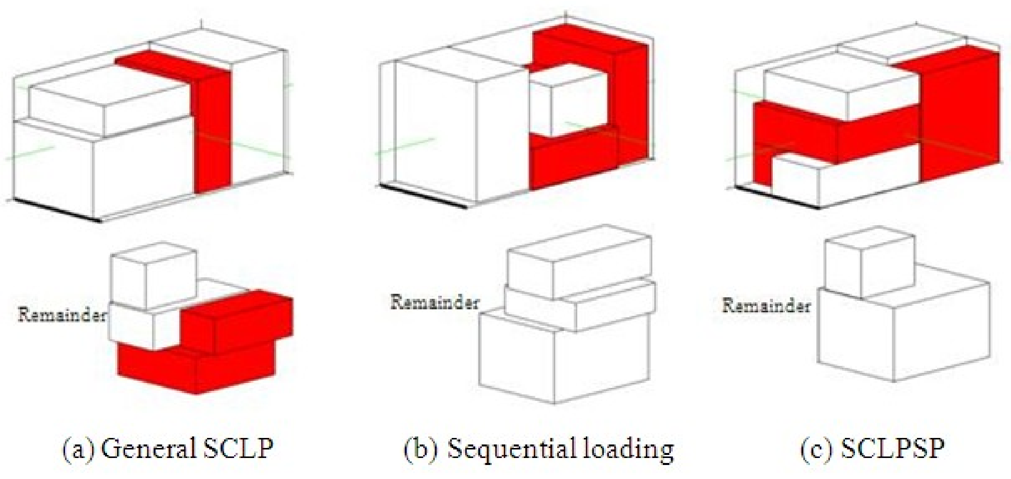


Figure 14: Three Loading Models.

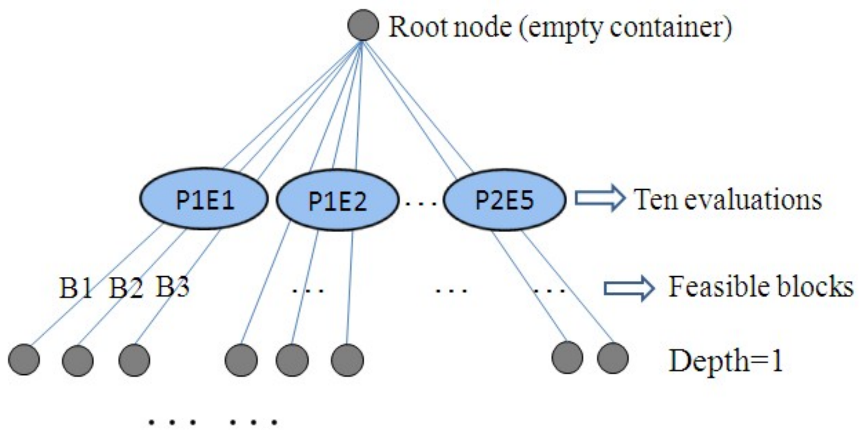


Figure 15: Tree Search for SCLPSP.

ten evaluations for block selection: P1E1, P1E2, P1E3, P1E4, P1E5, P2E1, P2E2, P2E3, P2E4, P2E5. For example,

$$\text{P1E2} = \begin{cases} bw \times bh & \text{, if the item is high priority,} \\ -\infty & \text{, if the item is low priority;} \end{cases} \tag{2.23}$$

$$\text{P2E2} = \begin{cases} -\infty & \text{, if the item is high priority,} \\ bw \times bh & \text{, if the item is low priority.} \end{cases} \tag{2.24}$$

Similar to the TRS_SCLP algorithm, the blocks that achieve the highest values for one of the ten evaluations are called feasible blocks, and these blocks constitute the branches of the search tree (Figure 15).

Considering the characteristics of the SCLPSP, any infeasible solution (or partial solution) that violates the priority constraint must be eliminated. In detail, the process can be described as follows:

(i) For each solution, if high-priority items are left behind, the solution is eliminated.

(ii) For each partial solution, the total volume of the residual high-priority items and the total volume of the usable spaces are calculated. If the former exceeds the latter, then the partial solution is eliminated.

The procedure of deleting the infeasible partial solutions, as mentioned in (ii), is explained in terms of pseudo-code (Table 3).

Table 3: Delete Infeasible Partial Solutions

```
Input: space list, Sol.
Output: Sol.
for all partial solution in partial_solution_list do
    high_priority_volume:= 0;
    usable_volume:= 0;
    for all item types do
        if current item type is high priority then
            high_priority_volume := high_priority_volume + residual volume
            of current item type;
        endif
    endfor
    for all space in space list do
    if current space is not marked as 'unusable' then
        usable_volume := usable_volume + volume of current space;
    endif
    endfor
    if high_priority_volume > usable_volume then
        delete current partial solution from partial_solution_list;
    endif
endfor
```

The pseudo-code of the TRS_SCLPSP algorithm is shown in Table 4.

Then the TRS_SCLPSP algorithm is incorporated in algorithm EXAC_SCLP to generate the exact algorithm for SCLPSP (EXAC_SCLPSP).

2.5 Computational Results

The proposed approach was implemented in Visual C++ 2003 under Windows XP. All tests were performed on an Intel Core 2 U9300 PC (1.2 GHz, 2 GB RAM).

Two well-known reference problem sets from literature are used for benchmarking purposes. These are the 15 test problems from Loh and Nee (1992)

Table 4: Algorithm TRS_SCLPSP

```
Input: items, container.
Output: $MaxU$ (maximum utilization of container).
initialize $MaxU = 0$;
initialize $Sol$ :={empty container};// the set of all partial solutions
while $Sol \neq \emptyset$
    for all partial solutions in $Sol$ do
        for all feasible blocks that achieve the largest value for one of the
        five evaluations (P1E1 - P2E5) do
            add current feasible block to current partial solution to generate
            a new partial solution;
            add the new partial solution to $Sol$;
        endfor
        if no item could be packed into the container and no high-priority
        items are left behind the volume and utilization of current partial
        solution is higher than $MaxU$ then
            $MaxU$ := volume utilization of the current partial solution;
        endif
        delete infeasible partial solutions;
        delete current partial solution from $Sol$;
    endfor
    delete the partial solutions that have identical evaluation function value;
    select $breadth$ best partial solutions from $Sol$;
endwhile
```

(LN data) and the 700 test problems from Bischoff and Ratcliff (1995) (BR data). For the LN data, the number of item types ranges from 6 to 10, and the number of available items ranges from 100 to 250. Therefore the problems are weakly heterogeneous. The BR data are divided into seven groups (BR1 - BR7) each containing 100 problems. With respect to the item types and numbers, the seven groups vary from weakly heterogeneous to strongly heterogeneous. Each group is distinguished by the number of different item types which increases from 3 types of BR1 to 20 types of BR7. The average number of items in each type is 50.2 for BR1, but decreases continuously and is only 6.5 for BR7. The container used in the BR data is the standard ISO 20FT container with three dimensions of 587cm $\times$ 233 cm $\times$ 220 cm.

Due to the lack of universally acknowledged test data for the SCLPSP, the test problems were generated from the above problems by adding shipment priorities to the different item types.

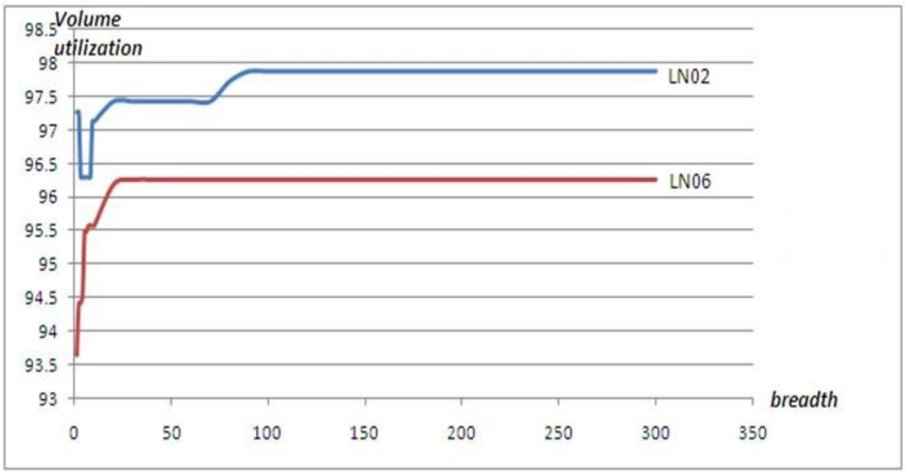


Figure 16: Volume Utilization in Different *breath* Values.

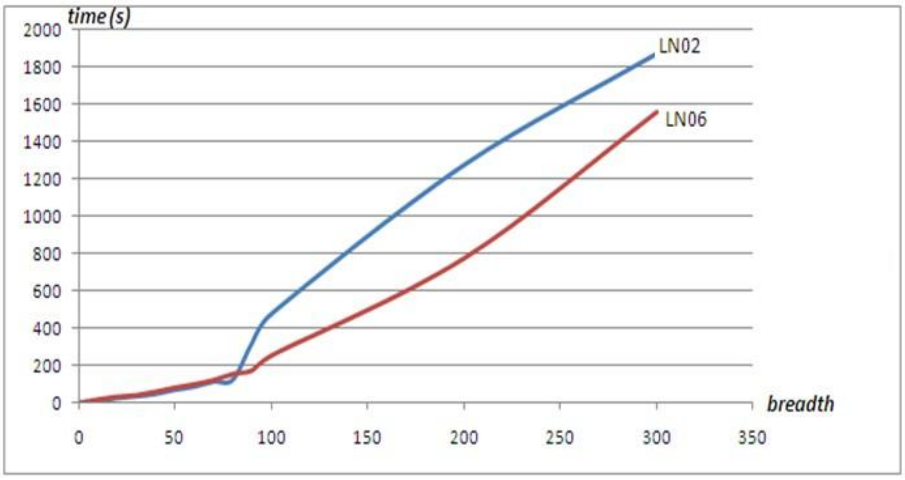


Figure 17: Running Time In Different *breath* Values.

A large number of experiments were carried out according to different values of the *breadth* parameter. Figure 16 and 17 show how the solution quality and the running times are affected by different values for *breadth* parameter. As shown in the two figures, for LN02 and LN06, $breadth \geq 100$ guarantees a stable, high volume utilization within a reasonable running time. Similar results were also obtained with other data. Therefore, the *breadth* parameter was set at 100.

In section 2.5.1.2 and 2.5.2.2, the shipment priority is considered and more branches are generated at each node of the search tree. Therefore, a larger value of *breadth* guarantees a stable, high volume utilization within a reasonable running time. A large number of experimental results showed that $breadth \geq 1000$ guarantees a stable, high volume utilization.

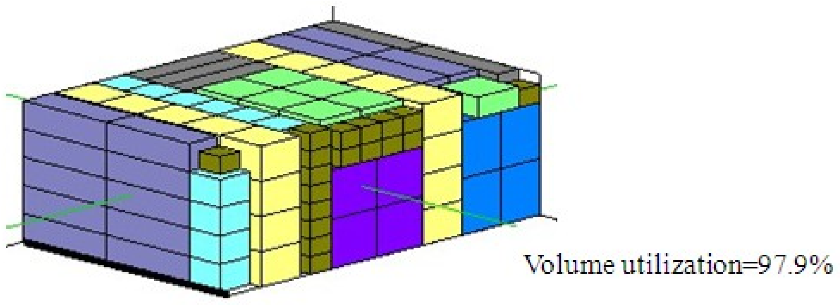


Figure 18: Illustrative Result of LN02.

2.5.1 Computational Results for the LN Data

2.5.1.1 Results for SCLP

The proposed EXAC_SCLP algorithm has been compared with the following seven approaches:

- BR_1995: the heuristic approach of Bischoff and Ratcliff (1995).
- BG_1998: the TS of Bortfeldt and Gehring (1998).
- EL_2002: the tree search approach of Eley (2002).
- TT_2004: the GA of Techanitisawad and Tangwiwatwong (2004).
- MO_2005: the GRASP approach of Moura and Oliveira (2005).
- LI_2007: the hybrid metaheuristic approach of Liang et al. (2007).
- WA_2008: the tree search approach of Wang et al. (2008).

Table 5 shows the volume utilization (%) obtained from the eight algorithms, including the proposed EXAC_SCLP algorithm. We found that in most problems other than LN02 and LN06, all items could be packed into one container by most algorithms. The best results for LN02 and LN06 were obtained with the proposed EXAC_SCLP algorithm. Therefore, the proposed EXAC_SCLP algorithm obtained the highest average volume utilization for all 15 problems. For highlighting, the average volume utilization for LN02 and LN06 are also listed. These results show that the proposed EXAC_SCLP algorithm has high validity for the general SCLP in comparison with other algorithms.

For the EXAC_SCLP algorithm, the time limit for each case is set to 300 seconds. The average running time is 72 seconds.

An illustrative result of LN02 is shown in Figure 18.

Table 5: Results for LN Data.

	*BR_*	*BG_*	*EL_*	*TT_*	*MO_*	*LI_*	*WA_*	*EXAC_*
Case	*1995*	*1998*	*2002*	*2004*	*2005*	*2007*	*2008*	*SCLP*
LN01	62.5	62.5	62.5	62.5	62.5	62.5	62.5	62.5
LN02	90.0(35)	96.7(28)	90.8(53)	91.3(31)	92.6(19)	89.7(*)	90.7(35)	97.9(25)
LN03	53.4	53.4	53.4	53.4	53.4	53.4	53.4	53.4
LN04	55	55	55	55	55	55	55	55
LN05	77.2	77.2	77.2	77.2	77.2	77.2	77.2	77.2
LN06	83.1(77)	96.2(32)	87.9(44)	90.5(52)	91.7(28)	91.4(*)	92.9(37)	96.3(21)
LN07	78.7(18)	84.7	84.7	84.7	84.7	84.6(*)	84.7	84.7
LN08	59.4	59.4	59.4	59.4	59.4	59.4	59.4	59.4
LN09	61.9	61.9	61.9	61.9	61.9	61.9	61.9	61.9
LN10	67.3	67.3	67.3	67.3	67.3	67.3	67.3	67.3
LN11	62.2	62.2	62.2	62.2	62.2	62.2	62.2	62.2
LN12	78.5	78.5	78.5	78.5	78.5	78.5	78.5	78.5
LN13	78.1(20)	85.6	85.6	85.6	85.6	85.6	85.6	85.6
LN14	62.8	62.8	62.8	62.8	62.8	62.8	62.8	62.8
LN15	59.5	59.5	59.5	59.5	59.5	59.5	59.5	59.5
Ave.	68.60	70.90	69.90	70.10	70.30	70.00	70.20	70.93
Ave1.	86.55	96.45	89.35	90.90	92.15	90.55	91.80	97.10

Note: the number of remaining items is given in parentheses;
(*) denotes the occurrence of remaining items, but the number of items is not provided.
Ave. denotes average value for all the cases, and *Ave1.* denotes average value for LN02 and LN06.

2.5.1.2 Results for SCLPSP

Algorithm EXAC_SCLPSP was used for the test of shipment priority. The test problems were generated from the LN problems by randomly setting the shipment priorities of the item types. As shown in Table 6, for example, the problem "LN01&02" was generated by combining the problems LN01 and LN02. The items in LN01 were all set to high priority, and the items in LN02 were all set to low priority. The problem "LN02_1" was generated from LN02 by setting the first four item types to high priority and the last four item types to low priority.

As shown in Table 6, even though some problems were generated from the same problem (LN02 or LN06), their volume utilizations differed depending on the assigned priorities. The results show that volume utilization depends not only on the item types, but also on the priorities of the items. The average of the volume utilizations for LN02_1 - LN02_5 is 97.00%, which is higher than the results of all the algorithms except for the proposed EXAC_SCLP algorithm in Table 5 for LN02. The average of the volume utilizations for LN06_1 - LN06_5 is 95.37%, which is higher than the results of all the algorithms except for the BG_1998 approach and the proposed EXAC_SCLP algorithm in Table 5 for LN06. The results show that, despite the addition of a new constraint (shipment priority), the proposed algorithm obtains better results than most algorithms that do not consider the shipment priority in Table 5.

The average running time for algorithm EXAC_SCLPSP is 1012 seconds, which is larger than that of EXAC_SCLP, because the parameter *breadth* is larger, and the shipment priority has been taken into account.

An illustrative result of LN01&02 is shown in Figure 19.

Table 6: Results for LN Data Considering Priority.

Case	*Original Case*	*High Priority*	*Low Priority*	*Volume Utilization (%)*
LN01&02	LN01, LN02	LN01	LN02	99.3
LN02_1	LN02	1,2,3,4	5,6,7,8	97.27
LN02_2	LN02	5,6,7,8	1,2,3,4	96.94
LN02_3	LN02	1,2,3,4,5	6,7,8	97.27
LN02_4	LN02	4,5	1,2,3,6,7,8	97.14
LN02_5	LN02	1,3,5,7	2,4,6,8	96.39
LN06_1	LN06	1,2,3,4	5,6,7,8	95.97
LN06_2	LN06	5,6,7,8	1,2,3,4	94.37
LN06_3	LN06	7,8	1,2,3,4,5,6	95.74
LN06_4	LN06	1,2,3,4,5	6,7,8	95.31
LN06_5	LN06	1,3,5,7	2,4,6,8	95.45
Ave.				96.47

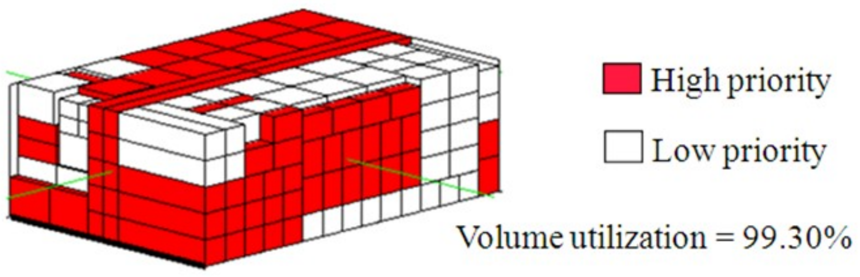


Figure 19: Illustrative Result of LN01&02.

2.5.2 Computational Results for BR Data

2.5.2.1 Results for SCLPS

The proposed EXAC_SCLP algorithm has been compared with the following seven approaches:

- BG_1998: the TS of Bortfeldt and Gehring (1998).
- GB_2002: the parallel GA of Gehring and Bortfeldt (2002).
- EL_2002: the tree search approach of Eley (2002).
- MA_2004: the parallel hybrid local search of Mack et al. (2004).
- BI_2006: the heuristic approach of Bischoff (2006).

- PA_2008: the GRASP approach of Parrenõ et al. (2008).
- FB_2010: the tree search approach of Fanslau and Bortfeldt (2010).

The average volume utilizations (%) of each problem are shown in Table 7. The proposed EXAC_SCLP algorithm competed well with the other approaches. Only the FB_2010 approach achieved higher average volume utilization for the 700 problems. The highest average volume utilization for BR5 was obtained with the proposed EXAC_SCLP algorithm.

The average running time of the EXAC_SCLP algorithm is 932 seconds.

Table 7: Results for BR Data.

	*BG_*	*GB_*	*EL_*	*MA_*	*BI_*	*PA_*	*FB_*	*EAXC_*
Case	*1998*	*2002*	*2002*	*2004*	*2006*	*2008*	*2010*	*SCLP*
BR1	92.63	88.1	88	93.7	89.39	93.85	94.51	93.9
BR2	92.7	89.56	88.55	94.3	90.26	94.22	94.73	94.54
BR3	92.31	90.77	89.5	94.54	91.08	94.25	94.74	94.35
BR4	91.62	91.03	89.3	94.27	90.9	94.09	94.41	94.08
BR5	90.86	91.23	89	93.83	91.05	93.87	94.13	94.17
BR6	90.04	91.28	89.2	93.34	90.7	93.52	93.85	93.52
BR7	88.63	91.04	88	92.5	90.44	92.94	93.2	92.9
Ave.	91.26	90.43	88.79	93.78	90.55	93.82	94.22	93.92

Table 8 shows the frequencies of the evaluations E1 - E5 in the experiment on the BR problems. For example, in BR1, 1088 blocks were packed into 100 containers, out of which 978 blocks achieved the highest value with E1 and 570 blocks with E2. The evaluation E1 is used more often than other evaluations (90.6%). It should be noted that in some cases, a block achieves the highest value with multiple evaluations; therefore, the sum of the frequencies of E1 - E5 is larger than the total number of the packed blocks.

Table 8: Frequencies of the Five Evaluations.

Case	*E1*	*E2*	*E3*	*E4*	*E5*	*Total*
BR1	978(89.9%)	570(52.4%)	536(49.3%)	563(51.7%)	529(48.6%)	1088
BR2	1397(89.6%)	856(54.9%)	776(49.8%)	798(51.2%)	833(53.4%)	1559
BR3	1823(87.9%)	1152(55.5%)	945(45.5%)	949(45.7%)	1096(52.8%)	2075
BR4	2132(90.3%)	1283(54.3%)	1056(44.7%)	1068(45.2%)	1235(52.3%)	2361
BR5	2538(92.7%)	1444(52.7%)	1208(44.1%)	1309(47.8%)	1389(50.7%)	2738
BR6	2751(91.1%)	1624(53.8%)	1352(44.8%)	1429(47.3%)	1619(53.6%)	3019
BR7	3264(91.0%)	1932(53.9%)	1518(42.3%)	1656(46.2%)	1888(52.6%)	3586
Total	14883(90.6%)	8861(53.9%)	7391(45.0%)	7772(47.3%)	8589(52.3%)	16426

2.5.2.2 Results for SCLPSP

As shown in Table 9, the test problems of SCLPSP were generated from problems BR1, BR4 and BR7 by adding different priorities in the following manner: BR1_1, BR4_1 and BR7_1 were generated by setting the first $[n/2]$ (n is the number of item types in each problem) item types to high priority and the remaining item types to low priority, and BR1_2, BR4_2 and BR7_2 were generated by setting the last $[n/2]$ item types to high priority and the remaining item types to low priority. Despite the addition of the shipment priority constraint, high volume utilizations were obtained.

The average running time of the EXCA_SCLPSP algorithm for each problem is 1825 seconds.

Table 9: Results from BR Data Considering Priority.

	High	*Low*	*Volume*
Case	*Priority*	*Priority*	*Utilization (%)*
BR1_1	1	2, 3	93.45
BR1_2	3	1, 2	93.48
BR4_1	1,2,. . . ,5	6,7,. . . ,10	92.99
BR4_2	6,7,. . . ,10	1,2,. . . ,5	93.14
BR7_1	1,2,. . . , 10	11,12,. . . ,20	91.75
BR7_2	11,12,. . . ,20	1,2,. . . ,10	91.74
Ave.			92.76

2.6 Conclusions

In the literature, mostly heuristic or metaheuristic algorithms have been proposed for the SCLP. In this paper an exact algorithm (EXAC_SCLP) has been proposed for solving the problem. Different from existing heuristic or metaheuristic algorithms, the proposed algorithm searches for optimal solution in the global solution space. Therefore, given enough computing time, stable and high-quality solution is guaranteed. An effective method to generate possible points for packing items has been proposed, which greatly reduces the search scope.

The SCLP is NP-hard. Thus the proposed exact algorithm is not a polynomial-time algorithm. For reducing the computing time, a heuristic method has been incorporated. Therefore the algorithm suits for both small and large instances because both solution quality and computing time are considered.

The heuristic method is a tree search method based on the greedy heuristic with five evaluations (TRS_SCLP). Five alternative evaluations for the utilization of one, two and three dimensions of the container space have been defined for block selection. The different blocks selected by each evaluation constitute the branches of the search tree. A method of space splitting and merging has also been embedded in the algorithm for making efficient use of the container space.

The 15 test problems generated by Loh and Nee (1992) and the 700 test problems generated by Bischoff and Ratcliff (1995) has been used to examine the validity of the EXAC_SCLP algorithm. The computational results show that the EXAC_SCLP algorithm obtains the highest average volume utilization for the LN problems and the second highest average volume utilization for the BR problems compared to the other algorithms.

A tree search considering the shipment priority (TRS_SCLPSP) generalized from the TRS_SCLP has been proposed to solve the SCLPSP. The TRS_SCLPSP has been incorporated in the exact algorithm for solving SCLPSP. The computational results show that the volume utilization depends not only on the item type but also on the priorities of the items. High volume utilization has been obtained despite the addition of the shipment priority constraint. Thus, the proposed algorithm for shipment priority is considered to be useful for solving practical problems with shipment priority. This constraint of shipment priority is important, for both the multiple CLP and the 3L-CVRP, as mentioned in the following chapters.

Chapter 3

Multiple Container Loading Problem

3.1 Problem Formulation of MCLP

The following notations are used in the formulations of multiple container loading problems.

θ	number of the container types.
t	number of the packed containers.
v_i	volume of an item of type i $(i = 1, \ldots, m)$.
μ_i	value per unit volume of an item of type i (thus, the value of an the item of this type is $= v_i\mu_i, i = 1, \ldots, m$).
e_k	available number of containers of type k $(k = 1, \ldots, K)$.
φ_k	volume of a container of type k $(k = 1, \ldots, K)$.
u_k	number of packed containers of type k $(k = 1, \ldots, K)$.
s_{ij}	number of items of type i packed in the j-th packed container $(i = 1, \ldots, m; j = 1, \ldots, t)$.

Other notations, such as the set of items I, have already been defined in Section 2.1, Chapter 1.

The three-dimensional bin packing problem (3DBPP) can then be formulated as follows.

[3DBPP]

$$\text{Minimize} \qquad \sum_{k=1}^{\theta} u_k\varphi_k, \tag{3.1}$$

$$\text{subject to} \qquad \sum_{j=1}^{t} s_{ij} = n_i, \forall i = 1, 2, \ldots, m. \tag{3.2}$$

The corresponding three-dimensional knapsack problem (3DKP) is stated as follows.

[3DKP]

$$\text{Maximize.} \qquad \sum_{i=1}^{m} \sum_{j=1}^{t} a_{ij} v_i \mu_i, \tag{3.3}$$

$$\text{subject to} \qquad \sum_{j=1}^{t} s_{ij} \leq n_i, \forall i = 1, 2, \ldots, m. \tag{3.4}$$

The rest of this chapter is organized as follows. Section 2.2 presents multiset partition model for MCLP. Section 3.3 and 3.4 provide exact and heuristic algorithms for MCLP, respectively. Section 3.5 presents the computational experiments and Section 3.6 summarizes this chapter.

3.2 Set Partition and Multiset Partition

The 3DBBP is belong to the class of partition problems, i.e., the set of items should be partitioned into some containers. A partition of a set is a set of nonempty subsets such that every element of the set is in exactly one of these subsets.

Example 3.1 (*set partition*): The set { 1, 2, 3 } has these five partitions:

{ {1}, {2}, {3} }.

{ {1, 2}, {3} }.

{ {1, 3}, {2} }.

{ {1}, {2, 3} }.

{ {1, 2, 3} }.

The total number of partitions of an n-element set is the Bell number as follows (Rota, 1964):

$$\frac{1}{e} \sum_{i=1}^{\infty} \frac{i^n}{i!}$$

Here e is the base of natural logarithm. Therefore it is very time consuming for generating all the partitions of a set. Particularly, in the 3DBBP for each partition of the items, a SCLP algorithm should be used to check whether the items in each subset can be packed into a container.

In real world instance, especially in mass production, usually some items have the same size, permitted orientations and other characteristics. As mentioned before, the items is denoted by a mutiset $\{1 \cdot n_1, \ldots, m \cdot n_m\}$, here n_i

is the number of items of type i ($i = 1, 2 \ldots, m$). Therefore the partition of items is a partition of multiset. Consideration of multiset partition can greatly reduce the repetition of set partition.

Example 3.2 (*repetitions in set partition*): For a set 1, 2, 3, 4, 5, if the item 1 and 2 are the same, and 3, 4, 5 are the same. The following partitions are the same partition:

{ {1, 3}, {2, 4, 5} },
{ {1, 4}, {2, 3, 5} },
{ {1, 5}, {2, 3, 4} },
{ {2, 3}, {1, 4, 5} },
{ {2, 4}, {1, 3, 5} },
...

As mentioned by Yorgey and Parker (2007), a partition of multiset can be transferred to a partition of vector. Knuth (2005) proposed an algorithm to generate all the partitions of multiset. However, in the MCLP, considering the capacity of container, not all the partitions are feasible. In the next section, we propose a exact algorithm for the MCLP, which is based on the partition of multiset.

3.3 Exact Algorithm For MCLP

For simplicity, we only consider the 3DBPP for single bin size bin packing problem. Hence the problem can be be formulated as follows.

$$\text{Minimize} \quad t, \tag{3.5}$$

$$\text{subject to} \quad \sum_{j=1}^{t} s_{ij} = n_i, \forall i = 1, 2, \ldots, m. \tag{3.6}$$

Other types of MCLPs can be solved analogously.

A solution of the 3DBPP is substantially a partition of the multiset I (i.e., m types of items are assigned into t packed containers). The whole number of items of each type is denoted by vector $\boldsymbol{N} = (n_1, \ldots, n_m)$, and the number of items of each type assigned into the j-th packed container is denoted by vector $\boldsymbol{s_j} = (s_{1j}, \ldots, s_{mj})$ ($j = 1, \ldots, t$). If constraint (3.6) holds, we say that $\boldsymbol{S} = \{\boldsymbol{s_1}, \ldots, \boldsymbol{s_t}\}$ is a partition of $\boldsymbol{N}$. In MCLP, considering the capacity of container, not all the partitions are feasible.

Without loss of generality, we assume that the packed containers are sorted in nonincreasing order of volume utilization. Obviously the following constraints hold, which greatly reduce the search scope:

$$\sum_{i=1}^{m} s_{i1} v_i \le LWH, \tag{3.7}$$

Table 10: Algorithm EXAC_MCLP

```
Input: items, container.
Output: $MinN$ (minimum number of packed containers).
initialize $MinN$ by using the priority-considering approach;
set $t := MinN$;
sort the item types in decreasing order of item volume;
while ($t > 0$)
    if $t =$ the continuous lower bound then
        return;
    endif
    $t = t - 1$;
        partial_solution_list:= $\{(0, \dots, 0)\}$;
        for $k = 0, \dots, t - 1$ do
            temp_partial_solution_list:= $\emptyset$;
            for all partial solution in partial_solution_list do
            for all possible values of $\boldsymbol{s_{k+1}}$ which satisfy (3.7) - (3.10) do
                generate a new partial solution;
                if the sum volume of items represented by $\boldsymbol{s_{k+1}}$ is larger than
                the volume of container then
                    continue;
                endif
                if the items represented by $s_{k+1}$ cannot be packed into a
                container by using the algorithm EXAC_SCLP then
                    continue;
                endif
                calculate lower and upper bound of the new partial solution;
                if $MinN <$ the lower bound of the new partial solution then
                    continue;
                endif
                add the new partial solution to temp_partial_solution_list;
                if $MinN >$ the upper bound of the new partial solution then
                    $MinN :=$ the upper bound of the new partial solution;
                endif
            endfor
        endfor
        if temp_partial_solution_list= $\emptyset$ then
            return;
        endif
    partial_solution_list:=temp_partial_solution_list;
    endfor
endwhile
```

$$\sum_{i=1}^{m} s_{i(k+1)}v_i \leq \sum_{i=1}^{m} s_{ik}v_i, \forall k = 1, \ldots, t-2, \tag{3.8}$$

$$\sum_{i=1}^{m} s_{i(k+1)}v_i \geq (\sum_{i=1}^{m} n_i v_i - \sum_{j=1}^{k}\sum_{i=1}^{m} s_{ij}v_i)/(t-k),$$
$$\forall k = 0, \ldots, t-2, \tag{3.9}$$

$$\sum_{i=1}^{m} s_{it}v_i = \sum_{i=1}^{m} n_i v_i - \sum_{j=1}^{t-1}\sum_{i=1}^{m} s_{ij}v_i. \tag{3.10}$$

We propose a branch and bound algorithm (algorithm EXAC_MCLP) to solve the MCLP. In the proposed algorithm, first we initialize the number of packed containers t by using the heuristic algorithm (the priority-considering approach mentioned in Section 3.4). If t equals the continuous lower bound (the total volume of items divided by the volume of a container, Martello et al., 2000), then output t as the optimal solution. Otherwise, set $t := t-1$ and use a branching tree to assign items to t packed containers without specifying their actual positions. The root node is $(0, \ldots, 0)$, which means that no item has been assigned to the packed containers. Each node in depth k of the tree denotes a partial solution $\{\boldsymbol{s_1}, \ldots, \boldsymbol{s_k}\}$, and all of the possible values of $\boldsymbol{s_{k+1}}$ that satisfy (3.7) - (3.10) are generated in its child nodes. For each of the child nodes, the SCLP algorithm is used to test whether the items can be packed into a container. If the answer is negative, the node is killed.

For decreasing the computing time, we calculate the upper bound (by using the priority-considering approach mentioned in the next section) and the continuous lower bound of each node. The minimum value of the upper bounds serves as the current solution. If the lower bound of a node is more than the current solution, the node is killed.

Algorithm EXAC_MCLP is described by the pseudo-code in Table 10.

3.4 The Priority-Considering Approach for MCLP

3.4.1 The Approach for 3DBPP

The item types are sorted in decreasing order of item volume. From practical experiments, we know that large items are usually more difficult to pack efficiently into the container space than small items. The pre-assignment of the large items can prevent them being left to the end and thus improve the utilization of the containers. However, no simple and efficient criterion has been found so far to distinguish between a large and a small item. For this reason, different divisions attempted, i.e., the division parameter d can vary from 0 to $m-1$. In the case $d = 0$, all items are set to low priority, and in the cases $d > 0$, the first d item types are set to high priority and are pre-assigned.

Table 11: Algorithm PRI_BBP

```
**Input**: items, containers.
**Output**: $MinV$ (minimum of the total volume of the containers used).
initialize $MinV := \infty$;
initialize $PList := \Phi$;//the packed-container list
sort the item types in decreasing order of item volume;
sort the container types in decreasing order of container volume;
for $d = 1, 2, \ldots, I - 1$ do
    use LMC_BBP to get $PList$;//LMC_BBP: load multiple containers
    if $MinV$ is larger than the container volume of $PList$ then
        set $MinV :=$ the container volume of $PList$;
    endif
endfor
```

Table 12: Procedure LMC_BBP

```
Input: items, containers, $d$.
Output: $PList$.
set $PList := \Phi$
initialize $PCont :=$ empty container;// packed-container
initialize $TempPCont :=$ empty container;// temporary packed-container
set the residual number of items of each type $(r_1, r_2, \ldots, r_m) := (n_1, n_2, \ldots, n_m)$;
set the first $d$ item types to high priority and the remaining types to low priority;
while there exist residual items do
    for container types $k = 1, 2, \ldots, K$ do
        use LOCP to get $TempPCont$;//LOCP: load one container considering
        // priority
        if the volume utilization of $TempPCont$ is larger than that of $PCont$
        then
            $PCont := TempPCont$;
        endif;
    endfor;
    add $PCont$ to $PList$;
    URNI;// update the residual number of items
endwhile
```

The best result from the I attempts is selected as the final result. The computational results with different values of the division parameter d are shown in Table 16, Section 3.5.

Note that when there are different container types (MBSBPP or RBPP), a greedy criterion is used for choosing the container type, i.e., the container type that achieves the highest volume utilization ratio is chosen. If all of the containers are identical (SBSBPP), then the choosing of the container type can be ignored.

The priority-considering approach for the 3DBBP (algorithm PRI_BBP) is described by the pseudo-code in Table 11. Table 12 shows the procedure LMC_BBP (load multiple containers), which is used in the algorithm PRI_BBP.

The procedure LOCP (load one container considering priority), which is used in the procedure LMC_BBP, is described as follows.

[procedure LOCP]

Input: items, containers, k, d.

Output: $TempPCont$.

- **Step 1**(*Pre-Load by the SCLP Algorithm*): If $d = 0$, or if no residual high-priority items exist, pack the low-priority items into a container of type k by using the SCLP algorithm. Set $TempPCont :=$ the packed-container and then stop. Otherwise, pack the high-priority items in the container by using the SCLP algorithm. Denote the number of high-priority items packed in the container by $(a_1^0, a_2^0, \ldots, a_d^0)$. Empty the packed-container.

- **Step 2**(*Load by the SCLPSP Algorithm*): Pack all the items (high- and low-priority items) in the container by using the SCLPSP algorithm, in which the high-priority items are packed with the constrained numbers $(a_1^0, a_2^0, \ldots, a_d^0)$. Set $TempPCont :=$ the packed-container. If no residual low-priority items exist, or some low-priority items have been packed into the container, then stop. Otherwise if the volume utilization of $TempPCont$ is larger than a given value (parameter $LeastU$), stop.

- **Step 3**(*(Improve the Packed-Container*): Generate another integer array $(a_1^*, a_2^*, \ldots, a_d^*)$ to replace $(a_1^0, a_2^0, \ldots, a_d^0)$. Here $(a_1^*, a_2^*, \ldots, a_d^*)$ is the solution of the following one-dimensional knapsack problem:

$$\text{Maximize} \quad \sum_{i=1}^{d} a_i^* v_i, \tag{3.11}$$

$$\text{subject to} \quad \sum_{i=1}^{d} a_i^* v_i < \sum_{i=1}^{d} a_i^0 v_i, \tag{3.12}$$

$$a_i^* \leq n_i, \forall i = 1, 2, \ldots, d. \tag{3.13}$$

  The above problem can be easily solved, for example, by a dynamic programming. If there exists a solution of the problem, pack all the items in the container by using the SCLPSP algorithm, in which the high-priority items are packed with the constrained numbers $(a_1^*, a_2^*, \ldots, a_d^*)$.

If the volume utilization of the packed-container is higher than that of $TempPCont$, set $TempPCont$:= the packed-container.

Table 13: Algorithm PRI_KP

```
Input: items, containers.
Output: $MaxV$ (maximum of the sum value of packed items.
sort the item types in decreasing order of $\mu_i$ and increasing order of item volume;
sort the container types in decreasing order of container volume;
initialize $MaxV := 0$;
find the greatest number $m^*$ for which the sum of the volumes of the first $m^*$
items does not exceed the sum of the volumes of the containers;
initialize $FulPack$ :=FALSE;// $FulPack$: whether the first $m^*$ items have been
//fully packed into the containers
initialize $Low := 0$;// low value for the binary search
initialize $High := m^*$; // high value for the binary search
while $High >= Low$ do
    set $m^* := (High + Low)/2$;
    set $i^*$:= the least index for which $\sum_{i=1}^{i^*} n_i \geq m^*$; // i.e., the $m^*$-th item is of
    // the $i^*$-th item type
    set $n^* := m^* - \sum_{i=1}^{i^*-1} n_i$;
    for $d = 0, 1, \ldots, i^* - 1$ do
        use LMC_KP to get $MaxV$ and $FulPack$; // LMC_KP: load multiple
        // containers
    endfor
    if $FulPack$ = TRUE then
        $Low := m^* + 1$;
    else
        $High := m^* - 1$;
    endif
endwhile
```

Step 3 serves to prevent the container space from being occupied only by the high-priority items and the volume utilization from being too low (less than the parameter $LeastU$). The volume of the high-priority items in the container is reduced slightly to allow the low-priority items to be packed in.

The procedures URNI (update the residual number of items), which is used in the procedure LMC_BBP, is described as follows: For a packed-container in which the numbers of packed items of each type are $(a_1, a_2, \ldots, a_m)$, update the residual number of items $(r_1, r_2, \ldots, r_m) := (r_1 - a_1, r_2 - a_2, \ldots, r_m - a_m)$.

Table 14: Procedure LMC_KP

```
Input: items, containers, $d, m^*, n^*, i^*$.
Output: $MaxV, FulPack$.
set the residual number of items of each type to be $(n_1, n_2, \ldots, n_{i^*-1}, n^*, n_{i^*+1},$
$\ldots, n_I)$;
set the residual number of containers of each type to be $(e_1, e_2, \ldots, e_K)$;
from the first $i^*$ item types, find the $d$ item types with the largest item volumes,
and set them to high priority and the remaining $i^* - d$ types to low priority;
while there exist residual containers and residual items do
    pack the first $i^*$ item types in the first residual container by using LOCP;
    // described in Section 4.1
    if $i^* < I$ then
        RPC;// reload the packed-container
    endif
    URNI;// update the residual number of items (described in Section 3.4.1)
    the residual number of containers of the corresponding type is decreased by 1;
endwhile
if $MaxV <$ the sum value of packed items then
    $MaxV :=$ the sum value of packed items;
endif
if the first $m^*$ items have been fully packed into the containers then
    $FulPack$ :=TRUE;
else
    $FulPack$ :=FALSE;
endif
```

3.4.2 The Approach for 3DKP

The item types are sorted according to the following criteria:

- *Main criterion*: decreasing order of μ_i (value per unit volume of the item).
- *Tie-breaker*: increasing order of item volume.

The main criterion is similar to the greedy criterion for the one-dimensional knapsack problem proposed by Dantzig (1957). The tie-breaker is to ensure that as many of the smaller items as possible are packed because they more easily make efficient use of the container space.

Because the available container space is not enough to pack all the items, it is necessary to choose a sub-set of items to be packed so that the total value of the packed items is as large as possible. The proposed approach is a greedy approach that searches for the greatest number m^* for which the first m^* items (note: not item types) can be fully packed into the containers. A binary search is used to decrease the search time.

The choosing of a container type is not considered because the available number of containers of each type is certain and finite. The given containers are sorted in decreasing order of container volume and are loaded one by one. The approach can be applied to MILOPP, MHLOPP, MIKP and MHKP, because it has no special requirement on the assortment of containers or items.

The priority-considering approach for the 3DKP (algorithm PRI_KP) is described by the pseudo-code in Table 13. Table 14 shows the procedure LMC_KP (load multiple containers), which is used in the algorithm PRI_KP.

The procedure RPC (reload the packed-container), which is used in the procedure LMC_KP, is described as follows: Reload the packed-container. Here the item types $i^* + 1, i^* + 2, \ldots, m$ can be packed, but the packed numbers of items of the first i^* types must be kept. In fact, it is a SCLPSP in which the first i^* item types are high priority, the last $m - i^*$ types are low priority, and the constrained numbers are the packed numbers of items of the first i^* types.

3.5 Computational Results

The proposed EXAC_MCLP algorithm was implemented in Visual C++ 2003 under Windows XP. All tests were performed on an Intel Core 2 U9300 PC (1.2 GHz, 2 GB RAM). In the next sections, the results of exact and heuristic algorithms for the SCLP and MCLP are shown, respectively.

The following test cases are used for benchmarking purposes.

- IV1: 47 test cases of 3DBPP with one container type (Ivancic et al., 1989)
- IV2: 17 test cases of 3DBPP with two or three container types (Ivancic et al., 1989)
- MO: 13 test cases of 3DKP with two or three container types (Mohanty et al., 1994)

A large number of computational experiments have been carried out for setting the *breadth* parameter (in the SCLP algorithm and SCLPSP algorithm) and the *leastU* parameter (Step 2 of the procedure LOCP, Section 3.4.1). For the sake of brevity, these computational experiments are not described in detail in this paper. Using these experiments, the *breadth* parameter was set to 50 and the *leastU* parameter was set to 90% because such values guarantee stable, high volume utilization in reasonable computing time for most test cases. The computing time of the EXAC_BPP algorithm for each case was limited to 30 minutes.

The proposed EXAC_MCLP algorithm has been compared with the following approaches:

Table 15: Comparative Results for IV1 data

Case	*IV_1989*	*BR_1995*	*BO_2000*	*EL_2002*	*EL_2003*	*TA_2008*	*EXAC_MCLP*
1	26	27	25	26	25	25	25*
2	11	11	10	10	10	10	10*
3	20	21	20	22	20	20	19*
4	27	29	28	30	26	26	26*
5	65	61	51	51	51	51	51
6	10	10	10	10	10	10	10*
7	16	16	16	16	16	16	16*
8	5	4	4	4	4	4	4*
9	19	19	19	19	19	19	19*
10	55	55	55	55	55	55	55*
11	18	19	18	18	17	16	16*
12	55	55	53	53	53	53	53*
13	27	25	25	25	25	25	25
14	28	27	28	27	27	27	27*
15	11	11	11	12	11	11	11*
16	34	28	26	26	26	26	26*
17	8	8	7	7	7	7	7*
18	3	3	2	1	2	2	2*
19	3	3	3	2	3	3	3*
20	5	5	5	2	5	5	5*
21	24	24	21	26	20	20	20
22	10	11	9	9	8	9	8*
23	21	22	20	21	20	20	20
24	6	6	6	6	6	5	5*
25	6	5	5	5	5	5	5
26	3	3	3	3	3	3	3*
27	5	5	5	5	5	5	4*
28	10	11	10	10	10	10	10
29	18	17	17	18	17	17	17
30	24	24	22	23	22	22	22
31	13	13	13	14	13	13	13
32	5	4	4	4	4	4	4*
33	5	5	5	5	5	5	4*
34	9	9	8	9	8	8	8
35	3	3	2	2	2	2	2*
36	18	19	14	14	14	14	14*
37	26	27	23	23	23	23	23*
38	50	56	45	45	45	45	45*
39	16	16	15	15	15	15	15
40	9	10	9	9	8	9	8*
41	16	16	15	15	15	15	15
42	4	5	4	4	4	4	4*
43	3	3	3	3	3	3	3*
44	4	4	3	4	4	3	3*
45	3	3	3	3	3	3	3
46	2	2	2	2	2	2	2*
47	4	3	3	3	3	3	3*
Total	763	763	705	716	699	698	693

Here'*' means the optimal solution.

- IV_1989: a sequential strategy (Ivancic et al., 1989).
- MO_1994: a sequential strategy (Mohanty et al., 1994).

Table 16: Results for IV1 data with Different Values of d

Case	$d = 0$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	*Min.*
1	27	25	-	-	-	25
2	11	10	-	-	-	10
3	21	24	21	19	-	19
4	29	28	26	28	-	26
5	62	51	51	59	-	51
6	10	10	10	-	-	10
7	16	16	16	-	-	16
8	4	5	4	-	-	4
9	19	19	-	-	-	19
10	55	55	-	-	-	55
11	16	16	-	-	-	16
12	53	53	53	-	-	53
13	25	25	25	-	-	25
14	28	28	27	-	-	27
15	11	11	11	-	-	11
16	28	26	26	-	-	26
17	8	7	8	-	-	7
18	2	2	2	-	-	2
19	3	3	3	-	-	3
20	5	5	5	-	-	5
21	22	20	20	21	22	20
22	8	10	10	9	8	8
23	21	21	21	21	20	20
24	5	6	6	6	-	5
25	5	5	5	5	-	5
26	3	3	3	3	-	3
27	4	5	5	-	-	4
28	10	10	10	-	-	10
29	17	17	17	17	-	17
30	23	24	23	22	-	22
31	13	13	13	13	-	13
32	4	4	4	-	-	4
33	5	5	4	-	-	4
34	9	8	8	-	-	8
35	2	2	-	-	-	2
36	18	14	-	-	-	14
37	26	23	25	-	-	23
38	47	45	45	-	-	45
39	15	15	15	-	-	15
40	9	9	8	9	-	8
41	17	15	15	17	-	15
42	4	4	4	-	-	4
43	3	3	3	-	-	3
44	3	4	3	-	-	3
45	3	3	3	3	-	3
46	2	2	2	2	-	2
47	3	4	3	3	-	3
Total	734	713	-	-	-	693

- BR_1995: a sequential strategy (Bischoff and Ratcliff, 1995).
- BO_2000: a sequential strategy (Bortfeldt, 2000).
- EL_2002: a sequential strategy (Eley, 2002).

- EL_2003: a bottleneck approach (Eley, 2003).
- TA_2008: a local search approach (Takahara, 2008).

Table 17: Comparative Results for IV2 data

Case	*IV_1989* *Ave(Cont)**	*BO_2000* *Ave(Cont)*	*EL_2003* *Ave(Cont)*	*TA_2008* *Ave(Cont)*	*EXAC_MCLP* *Ave(Cont)*
1	71.8(26/0)	74.7(25/0)	74.7(25/0)	74.7(25/0)	74.7(25/0)
2	97.6(7/13/6)	95.1(1/19/11)	99.9(2/13/17)	99.7(7/15/1)	99.1(2/22/3)
3	97.6(4/6/1)	99.7(4/1/2)	99.7(7/4/0)	99.6(7/1/0)	99.7(4/1/2)
4	85.8(10/1/7)	86.8(16/0/2)	87.4(2/0/14)	87.1(9/0/8)	87.4(2/0/14)
5	95.8(3/0/26)	97.9(7/0/23)	99.4(3/0/25)	98.7(1/1/25)	98.7(0/5/20)
6	92.2(7/6/1)	96.6(8/5/0)	96.6(6/9/0)	96.6(7/7/0)	96.6(7/7/0)
7	90.6(1/0/2)	90.6(1/0/2)	90.6(1/0/2)	90.6(1/0/2)	90.6(1/0/2)
8	81.2(3/3/11)	85.9(0/4/10)	88.4(9/2/5)	87.7(1/6/4)	**91.0(2/7/0)**
9	75.0(5/1/0)	90.2(2/1/1)	93.5(3/0/1)	92.7(5/0/0)	92.7(5/0/0)
10	87.3(2/5)	87.3(2/5)	88.5(1/7)	88.5(1/7)	**95.0(4/0)**
11	85.3(9/1/5)	87.8(14/1/1)	86.3(13/1/2)	85.0(5/11/2)	87.2(7/3/5)
12	88.7(0/2/4)	94.0(2/1/1)	94.0(2/1/1)	94.0(2/1/1)	94.0(2/1/1)
13	74.3(1/8)	92.2(2/0)	92.2(2/0)	92.2(2/0)	92.2(2/0)
14	76.3(3/2/11)	79.1(3/3/10)	79.2(2/3/11)	78.7(3/1/11)	**81.3(3/0/11)**
15	84.1(1/14)	89.0(0/15)	89.0(0/15)	89.5(2/11)	89.2(1/13)
16	82.7(4/0/0)	91.6(2/1/0)	91.6(2/1/0)	82.9(1/1/1)	91.6(2/1/0)
17	77.1(1/0/2)	84.7(0/0/3)	84.7(0/0/3)	91.6(0/1/1)	91.6(0/1/1)
Ave.	84.9	89.6	90.3	90.0	**91.3**

*Here *Ave* is the average volume utilization ratio (%) of containers; *Cont* is the number of containers used for each type.

Table 15 shows the comparative results for IV1. The numbers of used containers obtained from different approaches are shown in the table. In total, the proposed EXAC_MCLP algorithm required fewer containers to solve all 47 test cases than any other approaches. In addition, for the three test cases 3, 27 and 33, new best solutions were found. These results show that the proposed algorithms have high validity for the 3DBBP in comparison with other approaches. Moreover, within the given time limit, 34 results are proved to be optimal solutions by EXAC_MCLP algorithm.

Table 16 shows the results for IV1 with different values of the division parameter d. Note that if the number of item types was not larger than d, no result was obtained. It is difficult to determine which value of d was the most appropriate for all the cases. For most cases, the best result was obtained when $d > 0$, but this was not guaranteed for all the cases. For example, for cases 24 and 27, the result of $d = 0$ was better than that of $d > 0$. Therefore, the best result for all the values of d was selected as the final result.

Table 17 shows the comparative results for IV2. Because there is more than

Table 18: Comparative Results for MO data

Case	*Bound*	*MO_1994* *RB(Sum)*[*]	*BO_2000* *RB(Sum)*	*EL_2003* *RB(Sum)*	*TA_2008* *RB(Sum)*	*EXAC_MCLP* *RB(Sum)*
1	11112.0	77.7 (8640.0)	77.7 (8640.0)	77.7 (8640.0)	77.7 (8640.0)	77.7 (8640.0)
2	86016.0	97.1 (83494.4)	99.0 (85120.0)	99.3 (85376.0)	97.9 (84224.0)	99.3 (85376.0)
3	53500.0	99.6 (53262.5)	99.6 (53262.5)	99.6 (53262.5)	97.9 (52350.0)	99.6 (53262.5)
4	2720640.0	85.8 (2333440.0)	85.8 (2333440.0)	84.8 (2307840.0)	85.8 (2333440.0)	85.8 (2333440.0)
5	653750.0	75.8 (495500.0)	88.9 (581250.0)	89.3 (583750.0)	88.6 (579250.0)	88.6 (579250.0)
6	143424.0	96.4 (138240.0)	97.3 (139584.0)	98.5 (141216.0)	96.2 (137952.0)	97.6 (139968.0)
7	20203.2	82.5 (16668.0)	86.2 (17409.0)	84.2 (17004.0)	85.4 (17262.0)	85.3 (17226.0)
8	77986.8	84.3 (65741.0)	88.0 (68645.6)	88.6 (69121.2)	89.4 (69747.2)	**91.3 (71236.4)**
9	139356.0	85.9 (119772.0)	92.5 (128952.0)	95.9 (133632.0)	92.3 (128556.0)	93.9 (130860.0)
10	15360.0	100.0 (15360.0)	100.0 (15360.0)	100.0 (15360.0)	100.0 (15360.0)	100.0 (15360.0)
11	68353.2	73.1 (49995.0)	77.8 (53202.8)	77.4 (52873.6)	77.8 (53202.8)	77.8 (53202.8)
12	24964.0	94.3 (23529.0)	97.1 (24235.2)	94.8 (23673.0)	96.1 (23990.4)	96.1 (23990.4)
13	36556.8	100.0 (36556.8)	100.0 (36556.8)	100.0 (36556.8)	100.0 (36556.8)	100.0 (36556.8)
14	71552.0	78.9 (56492.8)	91.3 (65316.8)	96.0 (68723.2)	96.0 (68723.2)	96.0 (68723.2)
15	42922.8	87.5 (37558.8)	92.6 (39727.2)	91.8 (39382.2)	94.6 (40590.0)	94.6 (40590.0)
16	666829.6	83.5 (556458.0)	89.3 (595770.0)	88.7 (591535.0)	85.7 (571290.0)	**90.4 (603000.0)**
Ave.		87.6	91.4	91.8	91.3	**92.1**

[*]Here *RB* is the ratio (%) of the sum of the values of the packed items to the bound; *Sum* is the sum of the values of the packed items.

one container type for all the test cases, the number of containers used as well as the volume utilization ratio is shown. The proposed algorithm obtained the highest average volume utilization over all 17 test cases. For test cases 8, 10 and 14, new best solutions were found.

Table 19: Running Time (Sec.) for One Test Case

Test Cases	*Max.*	*Min.*	*Ave.*
IV1	49	< 1	8
IV2	59	1	20
MO	110	2	42

Table 18 shows the comparative results for MO. Here, the objective was not to maximize the sum volume, but the sum value of the packed items. The second column is an upper bound that was calculated in Eley (2003) by using integer programming. The ratio of the sum of the values of the packed items to the upper bound and the sum of the values of the packed items is shown. Again, the proposed algorithm obtained the highest sum value over all 16 test cases. For test cases 8 and 16, new best solutions were found.

The running times of the proposed algorithm is shown in Table 19. The running time increased with the number of different item types and container types, and for each test case it was less than 2 minutes.

3.6 Conclusions

In the literature not much work has been done on the multiple container loading problem because of the larger number of complexities occurring as compared to packing problems in lower dimensions. An exact algorithm (EXAC_MCLP) has been proposed for the problem, which calls the exact algorithm (EXAC_SCLP) for the single container loading problem. Moreover, the distribution of items into containers has been treated as partition of multiset. In consequence repetitions caused by same size items were avoided and the computing time was reduced.

For reducing computing time, a heuristic algorithm is proposed and incorporated into algorithm EXAC_MCLP. This is the priority-considering approach. The assignment of some large or awkwardly formed items is crucial for the whole packing. These items have been set to high priority and preferentially assigned. Within the proposed approach algorithm EXAC_SCLPSP has been used for solving the single container loading problem with the priority constraint.

The proposed algorithms have achieved the best known results for the test cases suggested by Ivancic et al. (1989) and Mohanty et al. (1994) with reasonable computing time. Moreover, some results have been proved by algorithm EXAC_SCLP to be optimal solutions.

Further research is required to find a more efficient criterion to identify the large and small items for reducing the computational effort. The consideration of additional practical constraints, such as separation of items or complete shipment of items, is also an issue for future research.

In line-haul transportation, usually the route is simple and constant, and the loading of containers directly related to cost, fuel consumption and CO2 emissions. However in branch transportation, both the container loading and vehicle routes must be taken into account. Therefore in next chapter we address the three-dimensional loading capacitated vehicle routing problem.

Chapter 4

The Three-Dimensional Loading Capacitated Vehicle Routing Problem

4.1 Problem Description of 3L-CVRP

Let $V = \{0, 1, \ldots, f\}$ be a set of $f+1$ vertices corresponding to a depot (vertex 0) and f customers (vertices $1, \ldots, f$), and E a set of edges (i, j) connecting all vertex pairs. Let $G = (V, E)$ be the induced graph and denote by a_{ij} the distance of edge (i, j) $(i, j \in V)$. In most practical contexts it is assumed that the distances satisfy the triangle inequality $a_{ij} \leq a_{ik} + a_{kj}$ $(i, j, k \in V)$, which is easily imposed by defining each distance a_{ij} as the shortest path from i to j. Let Γ be the number of available identical vehicles. Each vehicle has a container whose loading space is defined by length L, width W and height H. The weight capacity of each vehicle is T. A set of items $I = \{1 \cdot n_1, \ldots, m \cdot n_m\}$ should be delivered by the vehicles from the depot to the customers, and especially items of type k should be delivered to customer $v_k \in V \setminus \{0\}$ $(k = 1, \ldots, m)$. For simplicity, we assume that each vehicle travels with a const speed.

The 3L-CVRP calls for finding a set of at most Γ vehicle routes (one per vehicle, called single route), each one starting and ending at the depot, such that the following constraints are satisfied:

- *Item Clustering Constraint*: Each customer is served by exactly one vehicle.
- *Completeness Constraint*: All the items demanded by the customer must be delivered to the customers.

- *Weight Constraint*: No vehicle carries a total weight exceeding its capacity.
- *Loading Constraint*: A three-dimensional feasible loading is ensured for each vehicle. The three-dimensional feasible loading has also been mentioned is section 3.2, however, in the vehicle routing context, considering the item (cargo) stability and practical policies in transportation, some additional constraints should also be satisfied. They will be mentioned in Section 4.2.

In the conventional VRP, the objective is to find the minimum distance, or the minimum cost, which is usually assumed to be proportional to the distance. In this paper, for dealing with green logistics issue, we consider the fuel consumption and GHG emissions.

A solution of the 3L-CVRP can be generated by the following steps:

- Divide the set customers (or items, if the item clustering constraint is ignored) into subsets.
- For each subset, call an CLP algorithm to check whether the items in the group can be packed into a vehicle.
- For each subset, call an TSP algorithm to find a optimal (minimum distance, minimum fuel consumption, or minimum GHG emissions) route.

The rest of this chapter is organized as follows. Sections 4.2 explains the NP-hard problem with NP-hard constraint. Sections 4.3 mentions the practical loading constraints in vehicle routing context. Section 4.4 provides exact algorithm for 3L-CVRP. Section 4.5 provides relaxation algorithm for 3L-CVRP, which is a heuristic algorithm based on the exact algorithm. Section 4.6 mentions the formulas for fuel consumption and GHG emissions. Section 4.7 provides algorithm for TSP with fuel consumption, which can be used in 3L-CVRP considering different objectives. Section 4.8 provides algorithms for generalized 3L-CVRPs for real-world instances. Section 4.9 presents the computational results of the algorithms, and Section 4.10 summarizes the paper.

4.2 NP-Hard problem with NP-Hard Constraint

Different from other variants of the VRP, in the 3L-CVRP the loading constraint itself is a NP-hard problem. Therefore, we call that the 3L-CVRP is a NP-hard problem with NP-hard constraint. More formally, for a problem A, given a feasible solution *sol* of A and a additional constraint B, define the decision problem D:

'Whether *sol* satisfies constraint B?'

If D is a NP-hard problem, the constraint B is called a NP-hard constraint.

For example, for the VRP problem, the constraint of time windows is not a NP-hard constraint, and the constraint of three-dimensional loading is. Obviously a NP-hard problem with NP-hard constraint is much more difficult than a conventional NP-hard problem.

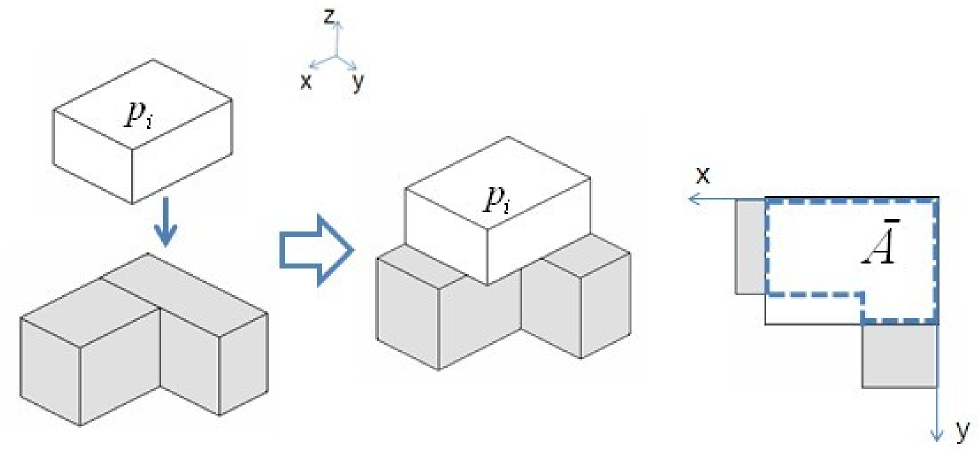


Figure 20: Supporting Area.

In the literature, some VRP instances with more than 100 customers have been solved optimally. However, for the 3L-CVRP, optimal solutions cannot be obtained even for instances with less than 20 customers. The reasons are the following:

- For each vehicle route, a SCLP algorithm should be used the test whether the items in the route can be loaded into a container, i.e., the SCLP algorithm is called very often for the 3L-CVRP.
- The SCLP algorithm must be flexible, since they should be adaptable to take into account all practical constraints in transportation.

Both reasons make the 3L-CVRP algorithm very time consuming and difficult to obtain good result. Most existing algorithms for the 3L-CVRP are heuristics or metaheuristics, which speed up the process of finding a satisfactory solution, however don't search for optimal solution in the global solution space. Therefore usually only some of the possible solutions are tried, and better solutions may be missed, even for small-scale instances.

4.3 Practical Loading Constraints in Vehicle Routing Context

In the definition of the SCLP in Section 2.1, only the maximum volume utilization of the container has been considered and practical constraints resulting from transportation have been ignored. However, in the vehicle routing context, considering the item (cargo) stability and practical policies in transportation, the following constraints should also be satisfied:

- Fragility constraint: Items are divided into two groups: fragile and non-fragile. Non-fragile items cannot be stacked on top of fragile ones, while fragile items can be placed on top of each other. Also non-fragile items can be placed on top of each other.

- Supporting area constraint: When an item p_i is placed on top of other items, its base must be supported by a minimum supporting area. This means that the items that are placed under p_i, and with their top touching directly the bottom of p_i, should form a cumulative area $\bar{A} \geq \alpha l_i w_i$, where $0 \leq \alpha \leq 1$ is a given parameter representing the minimum fraction of the area of p_i to be supported (Figure 20).

- LIFO (Last In First Out) policy: The items are loaded into some vehicles (containers) starting from the depot, and when the vehicles visit each customer a set of items required by the customer are unloaded. For the convenience of unloading, the loading order of items should be inverse to visiting order of customers.

Algorithm EXAC_SCLP is flexible and easy to modify for consideration of the additional constraints. When loading an item in a possible point, all the three constraints above should be tested. If one of the constraints is not satisfied, the item cannot be loaded and the next item is tested.

4.4 Exact Algorithm For 3L-CVRP

Optimization of the VRP is increasingly considered to be a more practical approach for real problems than it used to be in the past. This change of viewpoint is the result of the fact that rapidly decreasing computation costs are making higher quality solutions more desirable, even at the expense of more computation. Even if exact algorithms are not run to full optimality, the solutions obtained are likely to be better than what existing heuristics can provide, with an increasing robustness since a bound on the amount by which a particular solution differs from optimality can also be guaranteed.

Many exact algorithms for the VRP are based on the set partitioning formulation, which was first provided by Balinski and Quandt (1964). Let ω denote a single route, let $b_{i\omega}$ be a binary coefficient equal to 1 if and only if vertex $i \in V \setminus \{0\}$ belongs to route ω , let σ_ω be the optimal distance of route ω, and let ζ_ω be a binary variable equal to 1 if and only if route ω is used in the optimal solution. The problem is then

$$\text{Maximize} \quad \sum_{\omega} \sigma_\omega \zeta_\omega, \tag{4.1}$$

$$\text{subject to} \quad \sum_{\omega} b_{i\omega} = 1, i \in V \setminus \{0\}. \tag{4.2}$$

A direct application of this formulation is impractical because of the large number of potential single routes encountered in most nontrivial instances and of the difficulty of computing the σ_ω coefficients which requires solving an exponential number of instances of TSP. Especially, for the 3L-CVRP, a large number of instances of SCLP should be solved, which makes the 3L-CVRP algorithm very time consuming.

Table 20: Algorithm EXAC_3LCVRP

```
**Input**: items, vehicles, roads.
**Output**: $MinC$ (minimum distance of vehicles).
initialize $MinC$ by using the saving method;
initialize $t$ by using the MCLP algorithm;
while ($t > 0$)
    if $t = \Gamma$ then
        return;
    endif
    $t = t + 1$;
        partial_solution_list:= $\{(0, \ldots, 0)\}$;
        for $k = 0, \ldots, t-1$ do
            temp_partial_solution_list:= $\emptyset$;
            for all partial solution in partial_solution_list do
            for all possible values of $\omega_{k+1}$ which satisfy (4.6)-(4.8) do
                generate a new partial solution;
                if the sum volume of items represented by $\boldsymbol{\omega_{k+1}}$ is larger than
                the volume of container then
                    continue;
                endif
                if the items represented by $\boldsymbol{\omega_{k+1}}$ cannot be packed into a
                container by using the SCLP algorithm then
                    continue;
                endif
                calculate the $q$-route lower bound;
                if $MinC <$ the lower bound of the new partial solution then
                    continue;
                endif
                add the new partial solution to temp_partial_solution_list;
                calculate upper bound of the new partial solution by using
                saving method and TSP algorithm;
                if $MinC >$ the upper bound of the new partial solution then
                    $MinC :=$ the upper bound of the new partial solution;
                endif
            endfor
        endfor
        if temp_partial_solution_list= $\emptyset$ then
            return;
        endif
    partial_solution_list:=temp_partial_solution_list;
    endfor
endwhile
```

In this section we generate set partitions in a branching tree. A solution of the 3L-CVRP is denoted by a set of single routes $\{\boldsymbol{\omega_1}, \ldots, \boldsymbol{\omega_t}\}$ (t is the number of vehicles). Each single route is denoted by $\boldsymbol{\omega_j} = (\omega_{1j}, \ldots, \omega_{fj})$, where ω_{ij} is a binary variable equal to 1 if and only if customer i is included in the j-th single route ($i = 1, \ldots, f; j = 1, \ldots, t$). Let g_i and η_i denote the total weight and total volume of items demanded by customer i, respectively. The following constraints should be satisfied:

$$\sum_{j=1}^{t} \omega_{ij} = 1, \tag{4.3}$$

$$\sum_{i=1}^{f} \omega_{ij} g_i \leq T, \forall j = 1, \ldots, t, \tag{4.4}$$

$$\text{and each vehicle is feasibly loaded.} \tag{4.5}$$

Constraint (4.3) requires that each customer is served by exactly one vehicle, and constraint (4.4) prevents the weight of items in each vehicle from exceeding the weight capacity of the vehicle. Note that if constraint (4.5) is not considered, the problem is the general VRP.

Without loss of generality, we assume that the loaded vehicles are sorted in nonincreasing order of volume utilization. Obviously the following constraints hold, which greatly reduce the search scope:

$$\sum_{i=1}^{f} \omega_{i(k+1)} \eta_i \leq \sum_{i=1}^{f} \omega_{ik} \eta_i, \forall k = 1, \ldots, t-2, \tag{4.6}$$

$$\sum_{i=1}^{f} \omega_{i(k+1)} \eta_i \geq (\sum_{i=1}^{f} \eta_i - \sum_{j=1}^{k} \sum_{i=1}^{f} \omega_{ij} \eta_i)/(t-k),$$
$$\forall k = 0, \ldots, t-2, \tag{4.7}$$

$$\sum_{i=1}^{f} \omega_{it} \eta_i = \sum_{i=1}^{f} \eta_j - \sum_{j=1}^{t-1} \sum_{i=1}^{f} \omega_{ij} \eta_i. \tag{4.8}$$

The 3L-CVRP algorithm (algorithm EXAC_3LCVRP) iteratively calls a branch-and-bound algorithm. First we initialize the number of containers t by using the MCLP algorithm mentioned in Chapter 3. Then the number t is increased by one until $t = \Gamma$. For each value of t, we use a branching tree to assign items to t vehicles. The root node is $(0, \ldots, 0)$, which means that no item has been assigned to the vehicles. Each node in depth k of the tree denotes a partial solution $\{\boldsymbol{\omega_1}, \ldots, \boldsymbol{\omega_k}\}$, and all of the possible values of $\boldsymbol{\omega_{k+1}}$ that satisfy (4.6) - (4.8) are generated in its child nodes.

For decreasing the computing time, we calculate the upper bound of each node by using the saving method to assign the residual customers. The saving method was first proposed in Clarke and Wright (1964). It starts with an initial solution made up of back-and-forth single routes. At each iteration, it merges one single route with another single route, maximizing the saving of distance, and provided the merge is feasible. The process stops when it is no longer possible to merge routes.

The lower bound is the q-route lower bound. A q-routes is the minimum-cost single single route whose loaded weight is equal to a fix value q. In detail, let Λ be the ordered set of all possible loaded weight of a single route, starting with the smallest value. Let $q(j)$ be the value of the j-th element of Λ. Christofides et al. (1981) proved that that all the q-routes $q(1), \ldots, q(|\Lambda|)$ can be generated in pseudo-polynomial time. Let $\psi(i)$ be the value of a least distance single route passing through i, and having a loaded weight equal to $q(j)$, it is proved that

$$\sum_{i}^{f} \min_{j=1,\ldots,|\Lambda|} \{\psi(i) g_i / q(j)\} \tag{4.9}$$

is a valid lower bound on the cost of an optimal VRP solution. This bound is the sum, over all customers i, of a lower bound on the contribution made by i to the routing cost.

The minimum value of the upper bounds serves as the current solution. If the lower bound of a node is more than the current solution, then the node is killed. Algorithm EXAC_3LCVRP is described by the pseudo-code in Table 20.

4.5 Relaxation Method For 3L-CVRP

Most VRP algorithms consist of solving the set partitioning formulation with a subset of promising single routes. Most algorithm works with a set partitioning formulation in which the columns correspond to q-routes. However, for the 3L-CVRP the number of q-routes is still too large to compute because for each single route we must check the loading constraint.

In this section, a relaxation method is proposed for the 3L-CVRP. We attempt to replace loading constraint (4.5) with the following constraint:

$$\sum_{i=1}^{f} \omega_{ij} \eta_i \leq \rho LWH, \forall j = 1, \ldots, t, \tag{4.10}$$

The constraint requires that the volume of items in each vehicle cannot exceed a given ratio ρ of the vehicle space.

The method hierarchically consists of a subordinated and a superior module. The subordinated module is the algorithm EXAC_3LCVRP, however in which constraint (4.5) is replaced by constraint (4.10).

The superior module is a heuristic algorithm (algorithm RELAX_3LCVRP) which varies parameter ρ (multiplied by a parameter $\Theta, 0 < \Theta < 1$) and iteratively calls algorithm EXAC_3LCVRP until all the items are loaded. For decreasing computing time, only a fixed number (parameter $M1$) best solutions are kept, and a fixed number (parameter $M2$) of feasible single routes are selected from the $M1$ best solutions. Then we use algorithm EXAC_3LCVRP to deal with the selected feasible single routes. Algorithm RELAX_3LCVRP is described as follows:

[RELAX_3LCVRP]

Input: items, vehicles, roads.

Output: Minimum cost.

Parameter: $\rho, M1, M2, \Theta$.

- **Step 0** (*Initialize*):
  Use the CLP algorithm to load the items into a vehicle, and initialize ρ to the volume ratio of the loaded vehicle.
- **Step 1** (*Solve Relaxed Problem*):
  Use the EXAC_3LCVRP to solve the relaxed problem. Keep the $M1$ best solutions.
- **Step 2** (*Select Feasible Single Routes*):
  Use the SCLP algorithm to check the $M1$ best solutions. Select the feasible routes in which the items can be wholly loaded into a vehicle. If the number of feasible routes is equal to $M2$, go to Step 4.
- **Step 3** (*Reduce Volume Ratio*):
  Let $\rho := \rho\Theta$. If $\rho > 0$ go to Step 2.
- **Step 4** (*Deal with Feasible Single Routes*):
  Use algorithm EXAC_3LCVRP to solve the 3L-CVRP, in which only the selected $M2$ feasible routes are allowed.

4.6 Fuel Consumption and GHG Emissions

In the conventional VRP, the objective is to find the minimum distance, or the minimum cost, which is usually assumed to be proportional to the distance. In this paper, for dealing with green logistics issue, we consider the fuel consumption and GHG emissions. In late 2008, European Committee for Standardization (CEN) founded a working group CEN/TC 320 Transport - Logistics and Services WG10 Methodology for calculation, declaration and reporting on energy consumption and GHG emissions in transportation (http://lipasto.vtt.fi/indexe.htm). As mentioned in the report, for a certain vehicle travelling with a constant speed, the fuel consumption is approximately proportional to the travelling distance and linear correlate to the weight of loaded items, i.e.,

$$Q(K, R) = (Q_0 + \beta K)R, \tag{4.11}$$

$$M(K, R) = (M_0 + \gamma K)R, . \tag{4.12}$$

Where $F(K, R)$ and $M(K, R)$ are the fuel consumption (g) and CO2 emissions (ml), respectively, of an vehicle whose loaded weight (tone) is K, and travel distance is R (kilometre); Q_0 and M_0 are the fuel consumption and CO2 emission per kilometre of an empty vehicle, respectively; β is the unit fuel consumption (g/tone-kilometre) and γ is the unit CO2 emissions (ml/tone-kilometre). For a given truck in a certain condition, Q_0, M_0, β and γ are constant. For example, for an Euro 4 15T truck in delivery driving, the fuel consumption and CO2 emissions are the following:

$$\begin{aligned}Q(K,R) &= (154+7K)R,\\ M(K,R) &= (483+22K)R,.\end{aligned}$$

Emissions of the other GHGs have similar function as CO2. Therefore all the GHGs emissions can be represented by the CO2 emissions.

Note that the fuel consumption or CO2 emissions are not only determined by the distance. Therefor many effective algorithms for conventional VRP or TSP cannot be used directly.

4.7 The Traveling Salesman Problem with Fuel Consumption

The classical TSP is to find a route of minimum distance. Let $V\prime = \{0, 1, \ldots, \delta\}$ be a set of $\delta + 1$ vertices corresponding to a depot (vertex 0) and δ customers (vertices $0, 1, \ldots, \delta$), and $E\prime$ a set of edges (i, j) connecting all vertex pairs. Let $G\prime = (V\prime, E\prime)$ be the induced graph and denote by a_{ij} the distance of edge (i, j). A route starting and ending at the depot, and visiting each custom in $V\prime$ can be denoted by $0, c_1, \ldots, c_\delta, 0$ according to visiting order, where $\{c_1, \ldots, c_\delta\}$ is a permutation of $\{1, \ldots, \delta\}$.

From (4.11), the total fuel consumption of the route is

$$Q = \sum_{i=0}^{\delta} a_{c_i c_{i+1}}(Q_0 + \beta(D - \sum_{j=1}^{i} g_{c_j})).$$

Where g_{c_i} is the weight of items demanded by customer c_j, $D = \sum_{j=1}^{\delta} g_{c_j}$ is the weight of all the items demanded by the customers, and $c_0 = c_{\delta+1} = 0$.

The TSPFC is formulated as follows:

$$\begin{aligned}&\text{Maximize} && Q, && (4.13)\\ &\text{subject to} && \{c_1, \ldots, c_\delta\} \text{ is a permutation of } \{1, \ldots, \delta\}. && (4.14)\end{aligned}$$

Note that if the objective function is replaced by $\sum_{i=0}^{\delta} a_{c_i c_{i+1}}$, the problem is the classical TSP.

Example 4.1: Four vertices (depot 0, customer 1, 2 and 3) and the distance between every two vertices are shown in Figure 21 (a). Assume the distance matrix is symmetric. The weight of items demanded by each customer is 3 tones. In Figure 21 (b) the route is {0, 1, 2, 3, 0} and in (c) that is {0, 1, 3, 2, 0}. Assume the vehicle is Euro 4 15T truck. The distance of (b) is

$$2 + 10 + 10 + 2 = 24.$$

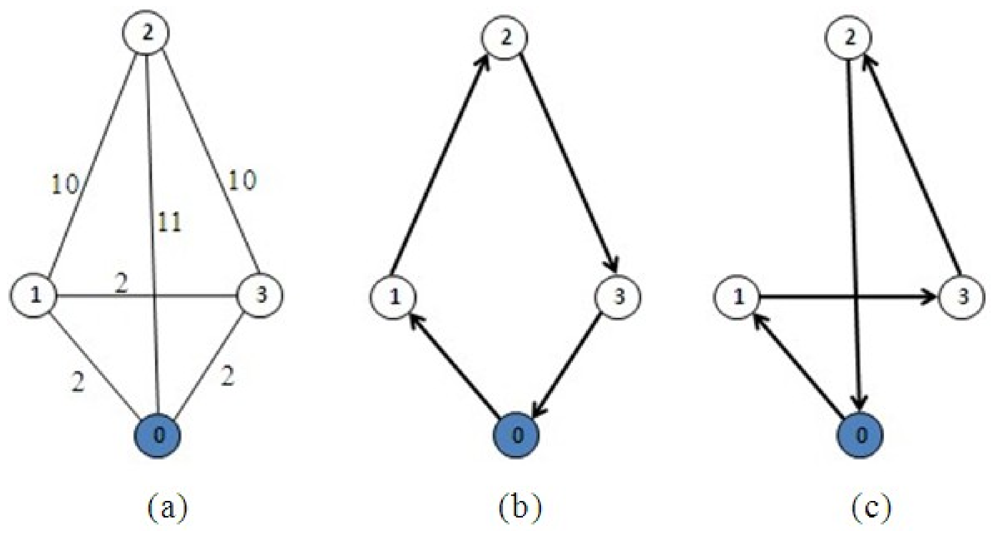


Figure 21: The TSP Considering Distance or Fuel Consumption.

And the distance of (c) is

$$2 + 2 + 10 + 11 = 25.$$

The fuel consumption of (b) is

$$(154 + 7 \times 9) \times 2 + (154 + 7 \times 6) \times 10 + (154 + 7 \times 3) \times 10 + 154 \times 2$$
$$= 4452.$$

And the fuel consumption of (c) is

$$(154 + 7 \times 9) \times 2 + (154 + 7 \times 6) \times 2 + (154 + 7 \times 3) \times 10 + 154 \times 11$$
$$= 4270.$$

The route of (b) has the minimum distance, but does not have minimum fuel consumption.

The 3L-CVRP is especially relevant for the cases that deal with large items. There are not many items are loaded in each vehicle, therefore each vehicle will not travel many customers. Therefore we expected to solve the TSPFC by using exact method, but not some metaheuristic such as genetic algorithm, ant colony algorithm or particle swarm algorithm, because those algorithms are more suitable for large-scale instances. There are some high-efficiency exact algorithms have been proposed for the TSP, such as Carpaneto algorithm (Carpaneto et al., 1995). However, they are not suitable for the TSPFC, because the cost (fuel consumption) between every two vertices is not constant and therefore it is difficult to estimate the upper bound or lower bound.

We propose a branch and bound algorithm (algorithm EXAC_TSPFC) to solve the TSPFC. First at all we use the Lin-Kernighan algorithm (Lin and Kernighan, 1973) to get a minimum-distance route, which serves as the current solution. Then we process a breadth-first search in a branching tree. The root node of the tree is 0, which means that each route starts form the depot 0.

Each node in depth k represents the first k visited customers ($0 < k < \delta$), which denoted by $\{c_1, \ldots, c_k\}$. The upper bound and lower node of each node in depth k are calculated as follows.

Let

$$Q_1 = \sum_{i=0}^{k-1} a_{c_i c_{i+1}} (Q_0 + \beta(D - \sum_{j=1}^{i} g_{c_j})),$$

which means that the fuel consumption of visiting the first k customers.

For any permutation of the residual $\delta - k$ customers $c_{k+1}, \ldots, c_\delta$, the fuel consumption of visiting the residual customers and ending at the depot is

$$Q_2 \;=\; \sum_{i=k}^{\delta} a_{c_i c_{i+1}} (Q_0 + \beta(D - \sum_{j=1}^{i} g_{c_j})).$$

For any $r \in \{k+1, \ldots, \delta\}$, we have

$$\begin{aligned} Q_2 &= \sum_{i=k}^{r-1} a_{c_i c_{i+1}} (Q_0 + \beta(D - \sum_{j=1}^{i} g_{c_j})) + \sum_{i=r}^{\delta} a_{c_i c_{i+1}} (Q_0 + \beta(D - \sum_{j=1}^{i} g_{c_j})) \\ &\geq \sum_{i=k}^{r-1} a_{c_i c_{i+1}} (Q_0 + \beta g_{c_r}) + \sum_{i=r}^{\delta} a_{c_i c_{i+1}} Q_0 \\ &\geq a_{c_k c_r} (Q_0 + \beta g_{c_r}) + a_{c_r 0} Q_0. \end{aligned}$$

Note that $Q = Q_1 + Q_2$. Therefore a lower bound of Q is

$$\begin{aligned} Q^{L1} &= Q_1 + \max_{c_r \in \{c_{k+1}, \ldots, c_\delta\}} (a_{c_k c_r} (Q_0 + \beta g_{c_r}) + a_{c_r 0} Q_0) \\ &= Q_1 + \max_{i \in V\prime \setminus \{0, c_1, \ldots, c_k\}} (a_{c_k i} (Q_0 + \beta g_i) + a_{i0} Q_0) \end{aligned}$$

On the other hand,

$$\begin{aligned} Q_2 &= a_{c_k c_{k+1}} (Q_0 + \beta(D - \sum_{j=1}^{k} g_{c_j})) + \sum_{i=k+1}^{\delta} a_{c_i c_{i+1}} (Q_0 + \beta(D - \sum_{j=1}^{i} g_{c_j})) \\ &\geq a_{c_k c_{k+1}} (Q_0 + \beta(D - \sum_{j=1}^{k} g_{c_j})) + \sum_{i=r}^{\delta} a_{c_i c_{i+1}} Q_0 \\ &\geq a_{c_k c_{k+1}} (Q_0 + \beta(D - \sum_{j=1}^{k} g_{c_j})) + \Pi(c_{k+1}, 0, \{c_{k+2}, \ldots, c_\delta\}) Q_0. \end{aligned}$$

Here $\Pi(c_{k+1}, 0, \{c_{k+2}, \ldots, c_\delta\})$ is the minimum-distance route staring from c_{k+1}, visiting customers $c_{k+2}, \ldots, c_\delta$, and ending at depot 0, which can be obtained by Carpaneto algorithm (Carpaneto et al., 1995).

Therefor

$$\begin{aligned} Q_2 \;\geq\; & \max_{i \in V\prime \setminus \{0, c_1, \ldots, c_k\}} (a_{c_k i} (Q_0 + \beta(D - \sum_{j=1}^{k} g_{c_j})) \\ & + \Pi(c_k + 1, 0, V\prime \setminus \{0, c_1, \ldots, c_k, i\}) Q_0). \end{aligned}$$

And another lower bound of Q is

$$\begin{aligned} Q^{L2} &= Q_1 + \max_{i \in V\prime \setminus \{0,c_1,\ldots,c_k\}} (a_{c_k i}(Q_0 + \beta(D - \sum_{j=1}^{k} g_{c_j})) \\ &\quad + \Pi(c_k + 1, 0, V\prime \setminus \{0, c_1, \ldots, c_k, i\})Q_0). \end{aligned}$$

We use $Q^L = \min\{Q^{L1}, Q^{L2}\}$ as the lower bound.

Then use the Lin-Kernighan heuristic (Lin Kernighan, 1973) to get a route which visits the residual customers and ends at the depot. The upper bound Q^U is the fuel consumption of the route.

For each of the nodes, we get the lower bound and upper bound. The minimum value of the upper bounds serves as the current solution. If the lower bound of a node is more than the current solution, then the node is killed.

If the objective is minimum CO2 emissions, we can get a similar algorithm. Therefore three kinds of TSP algorithms can be used for 3L-CVRP according to different objectives: minimum distance, minimum fuel consumption and minimum CO2 emissions.

4.8 The 3L-CVRP with Item Partition and Shipment Priority

In the real world instance, the routing and loading problem may be much different with archetypal versions of the 3L-CVRP. Particularly, for some cases the item clustering constraint or completeness constraint may not be satisfied. In detail:

- The items demanded by one customer may exceed the capacity of one vehicle. Therefore one customer may be served by more than one vehicle, and the items belong to the same customer must be partitioned into different vehicles.
- The total volume or total weight of items may exceed the capacity of vehicles. Therefore not all the items demanded by the customer must be delivered. Usually some items are more urgently demanded than others, i.e., they are of high priority, and these items can be wholly loaded by the given Γ vehicles.

The problem is called the 3L-CVRP with item partition (3L-CVRPNIP), for the former case, and the 3L-CVRP with shipment priority (3L-CVRPNSP), for the latter case. In the residual part of this section, algorithms are proposed for the two problems, respectively. Note that the objective function is not specified. Therefore the algorithms can be applied for minimize either the total distance or the fuel consumption.

A exact algorithm is proposed for the 3L-CVRPNIP. Because one customer cannot be served by exactly one vehicle, the partition of items, but not of customers, is considered. The multiset I is partitioned into t subsets. Each subsets of items is loaded and delivered by a vehicle. For simplicity, the algorithm is not described in detail, because it is similar to algorithm EXAC_MCLP proposed in Section 3.3, except that a TSP algorithm is used to find the best single route of each vehicle.

However for large-scale instance the exact algorithm for the 3L-CVRPNIP is very time consuming, because there are much more possible partitions of items than that of customers. A heuristic algorithm (HEURI_3LCVRPNIP) is proposed, in which we use the following *direct-delivery criterion* to deduce the computational complexity: Use the MCLP algorithm to load the items demanded by each customer independently. If the volume utilization of a vehicle exceeds a given parameter ($Vdirec$), the vehicle is directly delivered, i.e., the route is back-and-forth single route, which visits only one customer. The items in other vehicles are dealt with by using algorithm EXAC_3LCVRPNIP.

Algorithm HEURI_3LCVRPNIP is described as follows.

[HEURI_3LCVRPNIP]

Input: items, vehicles, roads.

Output: Minimum cost.

- **Step 1** (*Load*):
  For each customer, use algorithm EXAC_MCLP to load the items demanded.

- **Step 2** (*Directly Deliver*):
  If the volume utilization of a vehicle is larger than $Vdirec$, the vehicle is directly delivered.

- **Step 3** (*Deliver Other Items*):
  Use algorithm EXAC_3LCVRPNIP to deliver items in other vehicles.

It is easy to solve the 3L-CVRPNSP by incorporating the SCLPSP algorithm. The algorithm (algorithm _3LCVRPNSP) is described as follows.

[HEURI_3LCVRPNSP]

Input: items, vehicles, roads.

Output: Minimum cost.

- **Step 1** (*Deliver High-priority Items*):
  Use the 3L-CVRP algorithm as mentioned before to deliver items.

- **Step 2** (*Add the Low-priority Items*):
  For each packed vehicle generated in Step 1, use the SCLPSP algorithm to add low-priority items to the vehicle, if the high-priority items of the same customer have already loaded into the vehicle. Note that the SCLPSP algorithm should be modified to satisfy the additional constraints mentioned in Section 4.3.

4.9 Computational Results

4.9.1 Computational Results for Distance

The test cases for the 3L-CVRP are the data suggested by Gendreau et al. (2006) (GE data). In these instances, the graphs, the weights demanded by the customers and the container weight capacities were taken from 27 Euclidean CVRP instances suggested by Toth and Vigo (2005) for a detailed description of CVRP test bed instances. For all the 27 test cases, the number of customers ranges from 15 to 100, and the number of item types ranges from 32 to 198. The container has dimensions $W = 25, H = 30$ and $L = 60$. For each customer, the number of requested items was uniformly and randomly generated between 1 and 3. Each item dimension was randomly generated according to a uniform distribution in the interval between 20% and 60% of the corresponding container volume.

The proposed RELAX_3LCVRP algorithms has been compared with the following approaches:

- GE_2006: a tabu search algorithm (Gendreau et al., 2006).
- FU_2010: an ant colony optimization algorithm (Fuellerer et al, 2010).

Similar as Gendreau et al. (2006) and Fuellerer et al. (2010) the input threshold α for the minimum supporting area (see Section 2.3) was set to 0.75.

In algorithm RELAX_3LCVRP, the parameter Θ is set to 0.95. The parameter $M1$ is set to $0.5f$ and $M2$ is set to $2f$ (f is the number of customers). A large number of experiments were carried out for the setting of parameters. For simplicity they are not mentioned here in detail.

The comparative results are shown in Table 21. The RELAX_3LCVRP algorithm obtained the minimum costs for almost all the cases, and, significant smaller average cost than that of other algorithms.

The average running time for all of the cases are shown in Table 22. The RELAX_3LCVRP algorithm obtained much better results, however with much more running time.

In Table 23 we examine the effect of the loading constraints discussed in Section 2.3, namely fragility, supporting area and LIFO policy. The RELAX_3LCVRP was used. Columns two refer to the results in which all constraints are imposed (i.e., they give the same values of the last column of Table 21). The four next pairs of columns report the results without the fragility constraint, without the supporting area constraint, without the LIFO constraint, and with none of these constraints. Strong reduction of average cost was obtained by removing the supporting area constraint (3.96%) and the LIFO constraint (3.81%). Removing the fragility constraint leaded to the lowest reduction of average cost (2.58%). The removal of all three constraints yielded an overall average cost reduction of 6.68%.

Table 21: Comparative Results for GE Data.

Case	*GE_2006*	*FU_2010*	*RELAX_3LCVRP*
1	291.00	291.00	291.00
2	334.96	334.96	334.96
3	447.73	409.79	393.98
4	448.48	440.68	441.44
5	464.24	453.19	466.20
6	504.46	501.47	500.49
7	831.66	797.47	776.17
8	871.77	820.67	801.39
9	666.10	635.50	621.73
10	911.16	841.12	803.25
11	819.36	821.04	768.91
12	651.58	629.07	616.11
13	2928.34	2739.80	2591.00
14	1559.64	1472.26	1377.02
15	1452.34	1405.48	1325.58
16	707.85	698.92	691.28
17	920.87	870.33	881.36
18	1400.52	1261.07	1069.71
19	871.29	781.29	739.53
20	732.12	611.26	550.03
21	1275.20	1124.55	1029.44
22	1277.94	1197.43	1061.20
23	1258.16	1171.77	1041.92
24	1307.09	1148.70	1090.91
25	1570.72	1436.32	1352.14
26	1847.95	1616.99	1430.15
27	1747.52	1573.50	1415.11
Ave.	1041.68	966.67	906.00

Table 22: Average Running Time.

GE_2006	*FU_2010*	*RELAX_3LCVRP*
2058.9	1746.6	6724.3

Table 23: Results for Different Loading Constraints.

	All constraints	*No fragility*	*No support*	*No LIFO*	*3D loading only*
1	291.00	291.00	291.00	291.00	291.00
2	334.96	334.96	334.96	334.96	334.96
3	393.98	386.77	372.56	364.28	364.28
4	441.44	441.44	441.44	441.44	430.88
5	466.20	442.45	409.89	409.89	391.69
6	500.49	500.49	498.65	496.40	496.40
7	776.17	776.17	740.27	740.27	721.09
8	801.39	746.14	746.14	801.39	741.59
9	621.73	621.73	607.65	607.65	607.65
10	803.25	803.25	705.24	705.24	705.24
11	768.91	710.92	710.92	710.92	703.73
12	616.11	610.37	610.37	610.37	610.37
13	2591.00	2314.88	2292.55	2292.55	2292.55
14	1377.02	1377.02	1347.78	1349.38	1168.11
15	1325.58	1325.58	1325.58	1325.58	1165.46
16	691.28	691.28	691.28	691.28	691.28
17	881.36	881.36	881.36	862.18	862.18
18	1069.71	1069.71	1046.95	1069.71	1046.95
19	739.53	703.28	739.53	703.28	689.36
20	550.03	550.03	516.96	550.03	516.96
21	1029.44	1029.44	974.03	1029.44	969.90
22	1061.20	1036.29	1036.29	1030.75	1023.77
23	1041.92	1041.92	1010.31	1010.31	991.70
24	1090.91	1090.91	1090.91	1058.10	1058.10
25	1352.14	1317.04	1352.14	1317.04	1215.00
26	1430.15	1411.40	1411.40	1401.17	1430.15
27	1415.11	1325.49	1308.02	1325.49	1308.52
Ave.	906.00	882.64	870.15	871.49	845.51
(%)*		2.58%	3.96%	3.81%	6.68%

*%: reduction of cost.

4.9.2 Computational Results for Fuel Consumption and CO2 Emissions

We still use the GE data, assuming the vehicle is Euro 4 15T truck. As mentioned in Section 4.6, the fuel consumption and CO2 emissions are the following:

$$\begin{aligned} Q(K,R) &= (154 + 7K)R, \\ M(K,R) &= (483 + 22K)R,. \end{aligned}$$

The results are shown in Table 24. Algorithm RELAX_3LCVRP was used, with different objectives. Column 2-4 are the distance, fuel consumption and CO2 emissions obtained when the objective is minimum distance. Column 5-7 are the results obtained when the objective is minimum fuel consumption. The results for the latter objective saved fuel consumption by about 3%, and CO2 emissions also by about 3%, compare to that of the former objective. The distance increased by about 2.2%. Especially, for case 6 the fuel consumption and CO2 emissions decreased by more than 10% and the distance increased by about only 1.4%. These results indicate that the algorithm can reduce fuel consumption and CO2 emissions, and especially for some cases there may be significant savings of fuel consumption and CO2 emissions, with very little increase in distance.

The minimum CO2 emission objective was also tested for the GE data. The results are exactly the same as that of minimum fuel consumption, because the CO2 emissions are nearly proportional to the fuel consumption.

4.9.3 Computational Result for an Real-World Instance

The real-world instance was provided by a third-part logistics company in Japan. The items are divided into groups according to their due date. This company makes transportation plan according to each date, and the instance came from one day's transportation plan. The vehicles have dimensions $L = 5.89, W = 2.35$ and $H = 2.2$. There are one depot and 31 customers. The demands of each customer consist of three-dimensional rectangular items. These items are divided into groups according to their due dates of delivery. Some items must be delivered in this day (high-priority items). Other items (low-priority items) also lie in the depot, but their due dates are later. The information of items is not described in detail. Only some statistic properties are shown in Table 25.

The distances between customers and depot were obtained by a GIS. For simplicity, only the distance between each customer and the depot are shown in the second column of Table 25.

In the last years, the company applied an existing TMS, which used a heuristic algorithm (old algorithm). In this paper we used algorithm HEURI_ 3LCVRPNIP and algorithm HEURI_3LCVRPNSP (new algorithms) for this instance. In algorithm HEURI_ 3LCVRPNIP the parameter $Vdirec$ is set to 80%. The comparative results are shown in Table 26. We can see that volume utilization was increased, and the number of vehicles was reduced. The total distance is reduced by about 5%. And particularly, the fuel consumption and CO2 emissions were reduced by about 8%.

Table 24 Results for Fuel Consumption and CO2 Emissions.

	Min-distance			*Min-fuel*		
Case	*di**	*fu**	*co**	*di*	*fu*	*co*
1	291.00	45302.63	142088.69	291.00	45302.63	142088.69
2	334.96	56711.33	177900.65	336.86	53314.51	167223.04
3	393.98	58835.02	184516.07	398.39	57968.27	181787.61
4	441.44	66135.50	207413.00	460.49	63602.07	199431.73
5	466.20	75404.75	236520.16	466.20	75404.75	236520.16
6	500.49	82764.00	259614.93	507.46	74247.34	232833.56
7	776.17	122085.39	382920.77	776.17	122085.39	382920.77
8	801.39	133771.79	419624.25	807.00	130888.60	410556.68
9	621.73	96867.51	303818.13	664.86	94862.37	297474.01
10	803.25	118988.04	373159.17	853.29	110144.69	345311.91
11	768.91	110367.07	346099.03	772.04	105181.56	329798.04
12	616.11	99078.26	310765.36	621.13	96987.37	304192.39
13	2591.00	359673.84	1127811.29	2591.00	359673.84	1127811.29
14	1377.02	219891.85	689711.60	1380.85	209457.44	656913.92
15	1325.58	206280.26	646983.81	1340.30	198881.30	623715.16
16	691.28	118580.11	371947.47	726.28	111358.46	349242.35
17	881.36	142193.20	446006.43	928.79	137768.97	432059.39
18	1069.71	172099.00	539812.86	1070.97	167438.63	525164.73
19	739.53	116851.97	366507.95	745.40	110889.11	347763.23
20	550.03	84544.37	265159.82	568.42	83341.88	261362.88
21	1029.44	164199.87	515025.63	1062.40	160862.22	504504.48
22	1061.20	165566.18	519287.13	1084.49	161431.73	506272.36
23	1041.92	154951.98	485950.03	1042.47	149191.01	467843.40
24	1090.91	170223.11	533894.32	1140.81	169387.26	531218.79
25	1352.14	213986.86	671177.01	1385.70	211910.99	664620.29
26	1430.15	229940.79	721240.89	1439.61	211720.54	663966.89
27	1415.11	233169.02	731399.23	1540.01	227734.90	714197.65
Ave.	906.00	141424.58	443568.73	926.01	137075.47	429881.31

**di*: distance; *fu*: fuel consumption; *co*: CO2 emissions.

Table 25 Real-world Instance.

		High-priority			*Low-priority*		
Customer	*Distance*	*nu**	*vo**	*wei**	*nu*	*vo*	*wei*
1	9.83	13	36.43	9338.78	9	30.24	8559.02
2	4.07	4	10.52	2572.62	2	3.48	581.60
3	20.55	5	21.61	4751.50	3	3.14	735.78
4	20.09	5	8.63	3059.02	2	2.62	590.16
5	12.78	14	17.29	3707.21	2	2.12	1875.17
6	12.33	2	3.75	559.86	0	0.00	0.00
7	44.44	3	2.57	582.82	0	0.00	0.00
8	4.17	17	33.52	7151.60	4	15.64	2511.60
9	21.80	5	5.47	1734.03	5	3.90	1150.00
10	28.16	27	15.70	3936.76	10	13.11	2432.40
11	29.71	11	2.03	521.70	0	4.01	1388.02
12	22.05	12	13.76	3292.01	0	0.00	0.00
13	8.96	38	33.11	12283.40	9	22.10	3692.41
14	8.80	56	41.12	14786.32	12	1.24	214.44
15	13.54	20	53.30	16040.00	0	0.00	0.00
16	9.66	24	57.79	15224.75	4	8.69	1353.00
17	27.30	10	35.20	5468.91	0	0.00	0.00
18	9.57	78	74.24	21797.42	8	12.00	2136.05
19	18.33	9	17.85	3977.92	0	0.00	0.00
20	11.05	9	8.28	2337.15	2	1.88	259.44
21	8.18	4	12.04	3673.14	4	8.69	1496.56
22	8.04	25	30.60	5528.42	5	20.06	2751.90
23	8.25	41	81.26	20477.58	79	67.53	11809.53
24	16.74	4	9.80	2941.47	10	21.72	8075.34
25	12.18	12	19.33	4042.73	24	47.55	8400.00
26	8.48	12	29.40	7285.40	10	24.39	6271.30
27	5.74	46	56.03	12439.69	6	9.48	1518.39
28	23.91	15	15.05	3653.33	0	0.00	0.00
29	4.80	24	36.87	8835.43	0	0.00	0.00
30	7.22	38	74.69	16438.93	10	11.63	2646.40
31	14.37	15	32.12	6265.82	12	13.66	3240.47
Total.	441.20	581.00	889.36	224705.72	221.00	348.88	73688.98
Ave.	14.36	18.87	28.69	7248.57	7.48	11.25	2377.06

**nu*: number; *vo*: volume; *wei*: weight.

Table 26 Comparative Results for Real-world Instance.

Algorithm	*nu**	*rv**	*di**	*fu**	*co**	*Time (m)*
Old	41	78.31	844.36	1370532.02	4303194.49	421.2
New	38	83.22	805.23	1260435.21	3957731.24	350.4

**nu*: vehicle number; *rv*: volume utilization(%); *di*: distance; *fu*: fuel consumption; *co*: CO2 emissions.

4.10 Conclusions

The 3L-CVRP is a combination of the VRP and CLP, and both are strongly NP-hard. Mostly heuristic or metaheuristic algorithms have been proposed for the 3L-CVRP. An exact algorithm (EXAC_3LCVRP) has been proposed for solving the 3L-CVRP, which is base on the model of set partition. A highly sophisticated heuristic algorithm (RELAX_3LCVRP) has proposed, which is based on the exact algorithm. It is time consuming but much better results are obtained. The validity of the proposed algorithms has been examined by computational experiments.

Considering the issue of green logistics, we have taken the fuel consumption and CO2 emissions into account. The traditional models and algorithms can not be applied directly. We have addressed the TSP with fuel consumption, and have proposed algorithms for solve them. Because the CO2 emission has similar function to the fuel consumption, the similar algorithms can also be proposed for minimizing the CO2 emissions. The TSP algorithms with different objectives can be used in the 3L-CVRP algorithm.

Moreover, in the real world instance, for improving the utilization of vehicles, the completeness constraint or cluster constraint may be violated. Algorithms for the generalized 3L-CVRPs have also been proposed. The computational results for real-world data show that proposed algorithms are fast and high-quality for solving practical problems.

In this paper, only rectangular items have been considered. Further research is required to deal with other shapes of items, such as circular column, sphere or irregular items. The consideration of additional practical constraints, such as the time windows, multiple depots, irregular items and variable speed, is also an issue for future research.

Chapter 5

Conclusions

Green Logistics is concerned with producing and distributing goods in a sustainable way, taking account of environmental and social factors. Transportation management is the main part of the logistics processes. Vehicle loading and routing optimization is core function of transportation management. In most transportation management systems the CLP and VRP were treated separately, and the green logistics issues were not been taken into account.

In this paper, we proposed algorithms for solving the single and multiple CLP. It can be used in many processes in green logistics, including material handling, waste management, packaging, warehousing and transportation. Furthermore, the combination of VRP and CLP, know as the 3L-CVRP has been addressed. Some well defined data structures are used, such as the staircase packing and the multiset partition. Simple exact, heuristic or metaheuristic algorithms can not be applied directly. Therefore we proposed different kinds of exact and heuristic algorithms. The integration of the exact and heuristic algorithms obtained excellent results, both for test data that come from the literature and for real-world instance.

Algorithms for routing and loading optimization is, in the real world, just part of the story. The algorithms have to be embedded in a system that enables the decision-maker to actually use it. The system has to be integrated into the information system of the enterprize, which can be a formidable task. Database and user interface are also important part of the system. The system usually has to interact with a number of different systems in an organization. It may receive information from a higher level system and provide information to a lower systems.

Robustness and reactive decision making is also an important issue in the real world. In practice, it often happens that soon after a vehicle schedule has been generated, an unexpected event happens that forces the decision-maker to make changes. Such an event may, for example, be change of the demand of items, or congesting in a road. In a reactive process, the decision-maker tries to accommodate the original objectives, and also tries to make the new schedule look, as much as possible, like the original one in order to minimize confusion. In order to do so, it is necessary for the original schedule to be robust so that the changes after a disruption are minimal.

Due to limited space, these issues are not mentioned in detail in this paper. They are issues for future research.

References

[1] Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W. J. (2007) *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, Princeton.

[2] Aprile, D., Egeblad, J., Garavelli, A., Lisi, S. and Pisinger, D. (2007) 'Logistics optimization: vehicle routing with loading constraints', *Proceedings of 19th International Conference on Production Research.*

[3] Baldacci, R., Christofides, N. and Mingozzi, A. (2008) 'An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts', *Mathematical Programming.*, Vol. 115, pp.351-385.

[4] Balinski, M. and Quandt, R. (1964) 'On an integer program for a delivery problem', *Operations Research.*, Vol. 12, pp.300-304.

[5] Bischoff, E.E. and Ratcliff, M.S.W. (1995) 'Issues in the development of approaches to container loading', *Omega - International Journal of Management Science*, Vol. 23, pp.377-390.

[6] Bischoff, E.E. (2006) 'Three dimensional packing of items with limited load bearing strength', *European Journal of Operational Research*, Vol. 168, pp.952-966.

[7] Blum, C. and Roli, A. (2003) 'Metaheuristics in combinatorial optimization: Overview and conceptual comparison', *ACM Computing Surveys*, Vol. 35, pp.268-308.

[8] Bortfeldt, A. (2000) 'Eine heuristik für multiple containerladeprobleme', *OR Spektrum*, Vol. 22, pp.239-262.

[9] Bortfeldt, A. and Gehring, H. (1998) 'Ein Tabu Search-Verfahren fúr Containerbeladeprobleme mit schwach heterogenem Kistenvorrat', *OR Spektrum*, Vol. 20, pp.237-250.

[10] Bortfeldt, A. and Gehring, H. (2001) 'A hybrid genetic algorithm for the container loading problem', *European Journal of Operational Research*, Vol. 131, pp.143-161.

[11] Carpaneto, G., Dell'Amico, M. and Toth, P. (1995) 'Exact Solution of Large Scale Asymmetric Travelling Salesman Problems', *ACM Transactions on Mathematical Software*, Vol. 21, pp.394-409.

[12] Christensen, S.G. and Rousøe, D.M. (2009) 'Container loading with multi-drop constraints', *International Transactions in Operational Research*, Vol. 16, pp.727-743.

[13] Christofides, N., Mingozzi, A and Toth, P. (1981) 'Exact algorithms for the vehicle routing problem, based on spanning tree shortest path relaxations', *Mathematical Programming*, Vol. 20, pp.255-282.

[14] Clarke, G. and Wright, J. W. (1964) 'Scheduling of vehicles from a central depot to a number of delivery points', *Operations Research*, Vol. 12, pp.568-581.

[15] Cook, S. (1971) 'The complexity of theorem proving procedures', *Proceedings of the third annual ACM symposium on Theory of computing*, pp.151-158.

[16] Dantzig, G. B. (1957) 'Discrete variable extremum problems', *Operations Research*, Vol. 5, pp.266-277.

[17] Davies, A.P. and Bischoff, E.E. (1999) 'Weight distribution considerations in container loading', *European Journal of Operational Research*, Vol. 114, pp.509-527.

[18] Dyckhoff, H. (1990) 'A typology of cutting and packing problems', *European Journal of Operational Research*, Vol. 44, pp.145-159.

[19] Edmonds, J. (1962) 'Covers and packings in a family of sets', *Bulletin of the American Mathematical Society*, Vol. 68, pp.494-499.

[20] Eley, M. (2002) 'Solving container loading problems by block arrangement', *European Journal of Operational Research*, Vol. 141, pp.393-409.

[21] Eley, M. (2003) 'A bottleneck assignment approach to the multiple container loading problem', *OR Spectrum*, Vol. 25, pp.45-60.

[22] Fanslau, T. and Bortfeldt, A. (2010) 'A tree search method for solving the container loading problem', *Informs Journal on Computing*, Vol. 22, pp.222-235.

[23] Gehring, H. and Bortfeldt, A. (2002) 'A Parallel Genetic Algorithm for Solving the Container Loading Problem', *International Transactions in Operational Research*, Vol. 9, pp.497-511.

[24] George, J.A. and Robinson, D.F. (1980) 'A heuristic for packing boxes into a container', *Computer and Operations Research*, Vol. 7, pp.147-156.

[25] Ivancic, N. J., Mathur, K. and Mohanty, B. B. (1989) 'An integer-programming based heuristic approach to the three-dimensional packing problem', *Journal of Manufacturing and Operations Management*, Vol. 2, pp.268-298.

[26] Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M. and Nagamochi, H. (2009) 'Exact algorithms for the two-dimensional strip packing problem with and without rotations', *European Journal of Operational Research*, Vol. 2, pp.268-298.

[27] Liang, S., Lee, C. and Huang, S. (2007) 'A Hybrid Meta-heuristic for the Container Loading Problem', *Communications of the IIMA*, Vol. 7, pp.73-84.

[28] Lin, L. (2011) 'Analasys to logistics cost as a percentage of GDP in China', *Economic and Trade Update*, Vol. 199, pp.27-28 (In Chinese).

[29] Lin, S. and Kernighan, B. W. (1973) 'An effective heuristic algorithm for the traveling-salesman problem', *Operations Research*, Vol. 21, pp.498-516.

[30] Jin, Z., Ohno, K. and Du, J. (2004) 'An efficient approach for the three-dimensional container packing problem with practical constraints', *Asia-Pacific Journal of Operational Research*, Vol. 21, pp.279-295.

[31] Mack, D., Bortfeldt, A. and Gehring, H. (2004) 'A parallel hybrid local search algorithm for the container loading problem', *International transactions in operational research*, Vol. 11, pp.511-533.

[32] Martello, S., Pisinger, D. and Vigo, D. (2000) 'The three-dimensional bin packing problem', *Operations Research*, Vol. 48, pp.256-267.

[33] Mohanty, B.B., Mathur, K. and Ivancic, N. (1994) 'Value considerations in three-dimensional packing - a heuristic procedure using the fractional

knapsack problem', *European Journal of Operational Research*, Vol. 74, pp.143-151.

[34] Moura, A. and Oliveira, J.F. (2005) 'A GRASP approach to the container-loading problem', *IEEE Intelligent Systems*, Vol. 20, pp.50-57.

[35] Moura, A. and Oliveira, J.F. (2009) 'An integrated approach to the vehicle routing and container loading problems', *OR Spectrum*, Vol. 31, pp.775-800.

[36] Parreño, F., Alvarez-Valdes, R., Tamarit, J.M. and Oliveira, J.F. (2008) 'A maximal-space algorithm for the container loading problem', *Informs Journal on Computing*, Vol. 20, pp.412-422.

[37] Pisinger, D. (1998) 'A tree search heuristic for the container loading problem', *Ricerca Operativa*, Vol. 28, pp.31-48.

[38] Pisinger, D. (2002) 'Heuristics for the container loading problem', *European Journal of Operational Research*, Vol. 141, pp.382-392.

[39] Raidl, R. and Puchinger, J. (2008) 'Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization', *Hybrid Metaheuristics*, Vol. 114, pp.31-62.

[40] Ren, J., Tian, Y. and Sawaragi, T. (2011) 'A tree search method for the container loading problem with shipment priority', *European Journal of Operational Research*, Vol. 214, pp.526-535.

[41] Rota, G. (2011) 'The number of partitions of a set', *American Mathematical Monthly*, Vol. 71, pp.498-504.

[42] Sbihi,A. and Eglese, R. W. (2007).'Combinatorial optimization and Green Logistics', *Omega - International Journal of Management Science*, Vol. 5, pp.99C116.

[43] Takahara, S. (2008) *A multi-start local search approach to the multiple container loading problem. In: Advances in Greedy Algorithms*, Witold Bednorz, I-Tech, Vienna, Austria, pp.586-599.

[44] Techanitisawad, A. and Tangwiwatwong, P. (2004) 'A GA-based Heuristic for the interrelated container selection and loading problems', *Industrial Engineering and Management System*, Vol. 3, pp.22C37.

[45] Terno, J., Scheithauer, G., Sommerwei, U. and Riehme, J. (2000) 'An efficient approach for the multi-pallet loading problem', *European Journal of Operational Research*, Vol. 123, pp.372-381.

[46] Toth, P., and Vigo, D. (2005) *The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications.*, Society for Industrial and Applied Mathematics, Philadelphia.

[47] Wallenburg, C., Cahill, D., Michael Knemeyer, A. and Goldsby, T. (2011) 'Commitment and trust as drivers of loyalty in logistics outsourcing relationships: cultural differences between the united states and germany', *Journal of Business Logistics*, Vol. 32, pp.83C98.

[48] Wang, Z., Li, K. and Levy, J.K. (2008) 'A heuristic for the container loading problem: a tertiary-tree-based dynamic space decomposition approach', *European Journal of Operational Research*, Vol. 191, pp.86C99.

[49] Wayne, D.B. (1989) 'Multiset theory', *Notre Dame Journal of Formal Logic*, Vol. 30, pp.36C66.

[50] Wächer, G., Haussner, H. and Schumann, H. (2007) ‘n Improved Typology of Cutting and Packing Problems‘, *European Journal of Operational Research*, Vol. 183, pp.1109-1130.

[51] Wirth, N. (1976) *Algorithms + Data Structures = Programs*, Englewood Cliffs, New Jersey.

[52] Woeginger, G. J. (2001) ‘A note on the depth function of combinatorial optimization problems’, *Discrete Applied Mathematics*, Vol. 108, pp.325C328.

[53] Yorgey, B. and Parker, C. (2007) ‘Generating multiset partitions’, *The Monad Reader*, Vol. 8, pp.5-20.

Published Papers

Journal Papers

1. Ren, J., Tian, Y. and Sawaragi, T. (2011) 'A tree search method for the container loading problem with shipment priority', *European Journal of Operational Research*, Vol. 214, pp.526-535.

2. Ren, J., Tian, Y. and Sawaragi, T. (2011) 'A priority-considering approach for the multiple container loading problem', *International Journal of Metaheuristics*, Vol. 1, pp.298-316.

3. Ren, J., Tian, Y. and Sawaragi, T. (2011) 'An exact algorithm for the three-dimensional loading capacitated vehicle routing problem', *International Journal of Business Performance and Supply Chain Modelling*, (Submitted and in review).

4. (Paper related to this research) Sawaragi, T., Xu, H., Tian, Y., Horiguchi, Y. and Ren, J. (2011) 'A TZBM-based algorithm for flow shop scheduling problem considering the decision makers preference', *Tetsu-to-Hagane*,The Iron and Steel Institute of Japan, Vol. 97, pp.352-359 (In Japanese).

International Conference Papers

1. Ren, J., Tian, Y. and Sawaragi, T. (2011) 'A priority-considering approach for the three-dimensional bin packing problem', *Proc. of the Conference for the International Federation of Operation Research Societies*, pp.14

2. Ren, J., Tian, Y. and Sawaragi, T. (2011) 'An relaxation method for the three-dimensional loading capacitated vehicle routing problem', *Proc. of the International Symposium on System Integration*, pp.45.

Japanese Conference Papers

1. Ren, J., Tian, Y. and Sawaragi, T. (2010) 'A tree search for container loading problem', *Proc. of the SICE (the Society of Instrument and Control Engineers) Kansai Chapter Young Researcher Conference*, pp.35-38.

2. (Paper related to this research) Tian, Y., Xu, H., Ren, J., Sawaragi, T. and Horiguchi, Y. (2010) 'TZBM-based decision making for production scheduling problem', *Proc. of the 37rd SICE Symposium on Intelligent Systems*, pp.303-308 (In Japanese).