UNIVERSITY OF GAZİANTEP
GRADUATE SCHOOL OF
NATURAL & APPLIED SCIENCES

SOLVING THE 3D CONTAINER LOADING PROBLEM WITH METAHEURISTICS

Ph.D THESIS
IN
INDUSTRIAL ENGINEERING

BY
GÜLESİN SENA DAŞ
JUNE 2010

Solving the 3D Container Loading Problem with Metaheuristics

PhD Thesis
in
Industrial Engineering
University of Gaziantep

Supervisor
Prof. Dr. Türkay DERELİ

by

Gülesin Sena DAŞ

Haziran 2010

ABSTRACT

SOLVING THE 3D CONTAINER LOADING PROBLEM WITH METAHEURISTICS

DAŞ, Gülesin Sena
Ph.D. in Industrial Eng. Dept.
Supervasor: Prof. Dr. Türkay DERELİ
June 2010, 121 pages

Container Loading (CL) is a quite interesting and very difficult problem to solve. Given a set of small rectangular items and a rectangular container with known dimensions, the aim is to load the items into the container in such a way that maximum volume utilization of the container is achieved. The Operations Research (OR) literature classifies this problem as NP-hard. Due to the complex nature of the problem, in the first part of this thesis two swarm intelligence (SI) based solution approaches namely Ant Colony Optimization (ACO) and Bees Algorithm (BA) are offered to solve the CL Problem. The results obtained with these approaches are compared with the available approaches in the literature and the performances of these approaches are discussed. Comparison of the proposed approaches in terms of utilization ratio revealed that BA is the best performing algorithm. In addition to this, a CL decision support system - to determine and visualize the packing pattern of a CL problem - is also designed.

In the second part, a multi-objective CL (MOCL) problem inspired from a real industrial problem is introduced. The main goal of the MOCL problem is to pack a group of items into the container without any overlap while maximizing the total weight of the packed items and the utilization rate of the container simultaneously. These two objectives are conflicting since the volume of an item is usually not proportional to its weight. The problem is solved via selected multi objective optimization methods (Goal Programming and Weighted-Sum) and the Simulated Annealing algorithm. The proposed algorithms are tested on real data provided by a distribution company and the positive impact of the obtained solution to the company's transportation policy is discussed.

Key Words: Container Loading, Swarm Intelligence, Ant Colony Optimization, Bees Algorithm, Multi- Objective Container Loading, Decision Support System

ÖZET

3 BOYUTLU KONTEYNER YÜKLEME PROBLEMİNİN METASEZGİSELLERLE ÇÖZÜLMESİ

DAŞ, Gülesin Sena
Doktora Tezi, Endüstri Müh. Böl.
Tez Yöneticisi: Prof. Dr. Türkay DERELİ
Haziran 2010, 121 sayfa

Konteyner Yükleme (KY) oldukça ilginç ve çözülmesi çok zor bir problemdir. Bir grup küçük dikdörtgen nesnenin, boyutları bilinen dikdörtgen bir konteynere; konteyner hacminden maksimum şekilde faydalanmak amacıyla yerleştirilmesidir. Yöneylem Araştırması (YA) yazını problemi NP-zor olarak sınıflamaktadır. Problemin karmaşık doğası gereği, bu tezin ilk kısmında sürü zekası tabanlı iki çözüm yaklaşımı ismen Karınca Kolonisi Optimizasyonu (KKO) ve Arı Algoritması (AA) KY problemini çözmek için önerilmiştir. Bu yaklaşımlarla elde edilen sonuçlar yazında mevcut diğer yaklaşımlarla kıyaslanmış ve bu yaklaşımların performansları tartışılmıştır. Önerilen algoritmalarla kıyaslandığında AA'nın performansının daha iyi olduğu görülmüştür. Bunlara ek olarak, bir KY problemine ait doldurma düzeninin belirlenmesi ve görüntülenmesi için bir KY karar destek sistemi de tasarlanmıştır.

Tezin ikinci kısmında ise, gerçek bir endüstriyel problemden esinlenen çok-amaçlı bir KY (ÇAKY) problemi tanıtılmıştır. ÇAKY problemin ana amacı; bir grup nesnenin herhangi bir çakışmada olmadan, yüklenen nesnelerin toplam ağırlığını ve konteyner kullanım oranını eş zamanlı maksimize ederek konteynere yüklemektir. Bu iki amaç, bir nesnenin hacminin ağırlığına orantılı olmadığında çoğunlukla birbirine zıttır. Problem seçilmiş çok-amaçlı optimizasyon metotları (Hedef Programlama ve Ağırlıklı-Toplam) ve Tavlama Benzetimi algoritması vasıtasıyla çözülmüştür. Önerilen algoritmalar bir dağıtım firması tarafından sağlanan gerçek veri üzerine test edilmiş ve elde edilen sonuçların firmanın ulaştırma politikasına olumlu etkisi tartışılmıştır.

Anahtar Kelimeler: Konteyner Yükleme, Sürü Zekası Algoritmaları, Karınca Kolonisi Optimizasyonu, Arı Algoritması, Çok-Amaçlı Konteynır Yükleme, Karar Destek Sistemi

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I am grateful to my advisor Prof. Dr. Türkay Dereli for his supervision, mentorship and foremost support. I would also like to thank the members of my thesis advisory committee, Prof. Dr. Rızvan Erol and Prof. Dr. Adil Baykasoğlu.

I would like to thank TUBİTAK BIDEB for their support under the PhD. Fellowship Programme.

My special thanks are to my colleagues and my intimate friends Tolunay Göçken and Mustafa Göçken from the Department of Industrial Engineering, University of Gaziantep for their valuable friendship and endless support. Without their motivation and support, it would be not able to complete this thesis.

I thank all my friends in Gaziantep who always made me feel at home. I am also grateful to my colleagues from TUBITAK, especially to Selda, Övgü, Tuba, Zeynep and Betül for always being very encouraging.

I am also grateful to Şiir Kılkış for proofreading this thesis and her invaluable comments on it and to Erdal Dayak for his great helps.

I would like to thank my mother and father, and also my siblings for their patience and their motivation. Last but not least, my very special thanks to my patient and sensible husband Taylan whom made me smile even in my worse times, and my sons, Ömer Görkem and Yiğit who had to grow up together with this PhD thesis. They are the best things in my life… I had an invaluable degree by being their mum…

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

3D - Three Dimensional

ACO – Ant Colony Optimization

BA – Bees Algorithm

BR – Bischoff and Ratcliff test cases

CL – Container Loading

CPP – Cutting and Packing Problems

DSS – Decision Support System

GA – Genetic Algorithm

LDB – Layer Determining Box

LN – Loh and Nee test cases

MOCL – Multiple Objective Container Loading

MOO - Multiple Objective Optimization

NP – Nondeterministic Polynomial-time

OR – Operations Research

PP – Packing Problem

SA – Simulated Annealing (Algorithm)

SI – Swarm Intelligence

SOO – Single Objective Optimization

TS – Tabu Search algorithm

CHAPTER 1

INTRODUCTION

1.1. Introduction

The globalization of the supply chains has significantly increased container shipment all around the world. %90 of all cargo moves in containers and approximately 250 million are shipped annually (van de Voort et al., 2003). One of the critical parts of the container shipment process is the loading phase. Packing a shipment into the containers or onto the pallet is a complex process. It often takes several days to allocate the pooled goods into the number of containers and then to pack the allocated goods into the containers. Occasionally, workers must unload some containers and then reload them in a different pattern to pack more goods in the containers (Chien and Deng, 2002). Thereby, the need for more efficient algorithms to ship and transport goods has become apparent. This study will make an attempt to offer alternative solution algorithms to solve Container Loading (CL) problems (both single objective and multi-objective) and a real-world case.

1.2. Motivation of the Thesis

With increasing global competition, organizations are forced to review their processes and their overall systems. At this stage, organizations have realized that it is no longer competitive to work alone but as a part of a certain network having the resources beyond the reach of an organization. This search for competitiveness has lead to a new structure called "**supply chains**" as well as its management.

The organizations that are linked together through the supply chain aim to supply goods and services to fulfill the demands of the end-customers. This is achieved by the material and information flow along the supply chain. An analogy to the flow of water in a river is often used to describe organizations near the source (raw material suppliers) as upstream, and those near the end customer (retailers) as downstream (Harrison et al., 2002). Harrison et al. (2002) defined the supply chain management as; the alignment of upstream and downstream capabilities of supply chain partners to deliver superior value to the end customers at less cost to the supply chain as a whole. In the literature, it is widely recognized that effective management of material and information along the chain is vital for the performance of the whole supply chain. Unless this is achieved, it is difficult to satisfy the customer demand on time. This makes "**logistics**" a critical element for the supply chain management.

The word "**logistics**" was originally used for military applications but today covers commercial activities as well (Wood et al., 2002). The Council of Logistics Management (Wood et al., 2002) defines logistics as; the process of planning, implementing and controlling the physical and information flows concerned with materials and final goods from point of origin to point of usage. A detailed definition of logistics can be given as the strategic management of the procurement, movement and storage of materials, parts and finished product inventory and the related information flows, through the organization and its marketing channel in such a way that the current and the future profitability of the organization are maximized through the cost-effective fulfillment of orders (Harrison et al., 2002).

Transportation is critical to logistical performance (Bowersox et al., 2002). The objective is to transport *goods* from one place to another, on time in an economic way. For organizations trying to decrease their operational costs, *"cheap transportation"* is a practical alternative. To take this alternative as an opportunity, *goods* should be transported from one place to another, on time in an economic way. With the aim of transporting more goods/items with low cost, less energy and time, especially in overseas logistics applications, goods should be packed optimally or at least near optimally. This introduces the question of the effective use of containers.

Containers are large boxes that are used to transport goods from one destination to another. Compared to conventional bulk, the use of containers has several advantages, namely less product packaging, less damage of goods and higher productivity (Agerschou et al., 1983). For transportation facilities, standard ISO containers are generally preferred. ISO shipping containers are provided in two basic sizes of length: 20ft (6.1 meters) also referred to as twenty-feet-equivalent-unit (TEU) containers and 40ft (12.2 meters) containers expressed by 2 TEU (web1). The overall (outside) width of the containers is a standard 2.438 meters and the heights vary between 8′6″ (2.59m) height and 9′6″ (2.896m). The most widely used type of container is the general-purpose (dry cargo) containers. General Purpose (GP) containers (Figure 1.1.a) are abbreviated as 20′DC (Dry Container) or 40'DC. Some special purpose containers such as open-top containers, flat racks etc. are also available (Figure 1.1.b& 1.1.c).

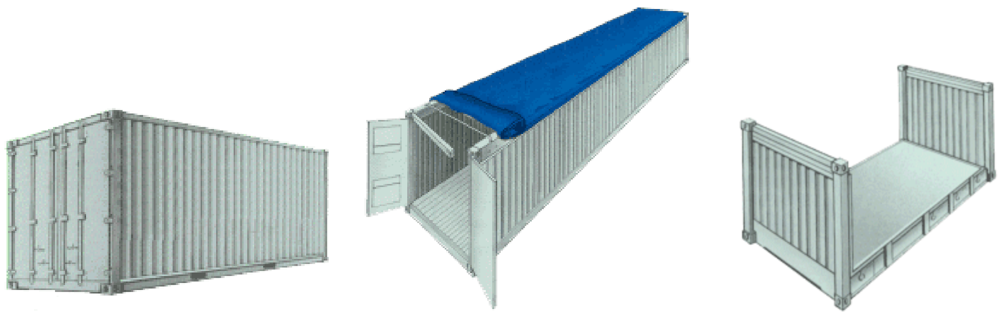


Figure 1.1 Different types of containers (a) General purpose container (b) Open-top container (c) Flat rack

Rising fuel cost now provide a strong incentive for container carriers to maximize available container space, thereby minimizing the number of required trips across the global container transportation system (Wang et al., 2008). The efficient loading of containers, that is, the minimization of empty spaces inside of them, is not only an economic requirement but also an ecological issue due to the adverse consequences of increased traffic on environmental resources (Parreno et al., 2010). This study will make an attempt to offer alternative solution algorithms to solve Container Loading (CL) problems (both single objective and multi-objective) and a real-world case.

CL is an interesting and difficult problem to solve. The problem can be described as follows: Given a set of small rectangular items and a rectangular container with known dimensions, load the items in to the container in such a way that maximum utilization of volume of the container is achieved.

The operations research literature classifies this problem as **NP-hard** (Pisinger, 2002). That is these problems are not solvable in polynomial time, which in turn means that it is not possible to find an exact solution for large sized problems. In general, only approximate solutions can be found for large sized problems. The use of special-purpose heuristics and meta-heuristic algorithms can be a good to provide good solutions for large sized problems.

1.3. Statement of the Thesis

In the first part of this thesis, two swarm intelligence based solution approaches namely Ant Colony Optimization and Bees Algorithm are offered to solve the CL problem. For the first part of the thesis, algorithms based on Swarm Intelligence (SI) are utilized. These algorithms, which are inspired by the behaviors of swarm of biological organisms, are preferred since they are previously applied to solve difficult and complex real-world problems. The detailed examination of the literature on the subject has revealed that approaches based on the application of SI techniques to the CL problems are quite limited. Having this in mind, the ultimate goal of this thesis is to offer some Swarm Intelligence (SI) based solution approaches to the CL Problem.

Another considerable contribution of this thesis is the definition of a new problem called *multi-objective CL problem.* The problem is mostly encountered in transportation and wholesaling industry. The main goal is to load the items (boxes) that would provide the highest total weight to the container in the best possible way. These two objectives (maximization of weight and maximization of volume utilization) are conflicting since the volume of a box is usually not proportional to its weight. Using some multi-objective optimization techniques such as goal programming and weighted-sum approach, the objectives are combined into a single objective. A Simulated Annealing (SA) algorithm accompanied by a heuristic filling procedure is then proposed to solve the model. The proposed algorithm has been tested on a set of benchmark problems available in the literature and also on real-world data provided by a distribution company.

1.4. Overview of the Thesis

This study includes eight chapters. Following this Introduction chapter, a detailed discussion about Cutting and Packing problems which also embraces the CL problems is presented in Chapter 2. In Chapter 3, the literature survey about the CL problems and the SI based technique are discussed. The literature survey about the CL problems is examined according to the type of the solution technique proposed: heuristic approaches, meta-heuristic approaches and exact approaches. In Chapter 4, the proposed heuristic filling procedure that is used together with all the proposed algorithms is presented. The developed BA is presented in Chapter 5. In Chapter 6, two algorithms based on ACO namely: hybrid–ACO-1 and hybrid-ACO-2 are presented. The computational results for both algorithms are also supplied within this chapter. In Chapter 7, the developed container loading support system is introduced and its functions are explained with examples. A new problem – *multi-objective CL problem*- is defined in Chapter 8. Finally, in Chapter 9 conclusions and future works are presented.

CHAPTER 2

CUTTING AND PACKING PROBLEMS

2.1. Cutting and Packing problems

Cutting and Packing Problems (CPP) are a set of widely studied problems. In the literature, they appear under various names such as cutting stock or trim-loss problems, bin or strip packing problems, vehicle, pallet or container loading problems, nesting problems, knapsack problems,...etc (Dyckhoff, 1990). In this study, a packing problem (PP), namely Container Loading problem is considered.

PP is concerned with finding a *good* arrangement of multiple *items* in larger containing regions (*objects*). The placement is described by a set of rules or constraints. The objective of the process is to maximize the volume utilization and hence, minimize the "wasted" area (Hopper and Turton, 1997).

These problems are encountered in many industries, with different industries incorporating different constraints and objectives. The wood, glass and paper industries are mainly concerned with the cutting of regular figures, whereas in the shipbuilding, textile and leather industries irregular, arbitrary shaped items are packed (Hopper and Turton, 2001).

Dyckhoff (1992) defines these problems as ***geometric - combinatorial*** problems. CPP are ***geometric-based*** because within each large object, one or more small items are arranged in such a way as to avoid overlapping and to fit into the object's geometric boundaries. They are also ***combinatorial-based*** since small items are to be assigned to the large objects. In other words, each large object is assigned a given set of small items and each item is assigned to at most one large object (Dyckhoff, 1992).

In his work “A typology for cutting and packing problems”, Dyckhoff presents several criteria to classify packing problems. These are:

- **Dimensionality** – This is the most important issue that should be stated in the problem definition. A problem can be stated as one- **(1),** two- **(2),** three- **(3)** or more dimensional **(N).**
- **Kind of Assignment -** Two categories can be presented under this heading: The assignment of all objects to a selection of items **(B)** or the assignment of all items to a selection of object **(V).**
- **Assortment of large objects** – The use of one **(O)** or more objects is mentioned under this heading. In case of using multi objects, these objects can be defined as identical **(I)** or different **(D)** in dimensions.
- **Assortment of small items** – Four types can be defined regarding the assortment of small items: few small items of different figures **(F),** many small items with most of them having different figures **(M),** many small items with relatively few different figures **(R)** and all small items with congruent figures **(C).**

According to this typology Container Loading problem is classified as a combined problem as **3/ V/ I** or **3/ B/ O**. An overview of CPP as summarized by Dyckhoff is given in Figure 2.1.

Following this work, an improved typology of cutting and packing problems is published by Wäscher et al. (2007), which is partially based on the Dyckhoff’s typology. This typology presents some new categorization criteria that is different from those of Dyckhoff.

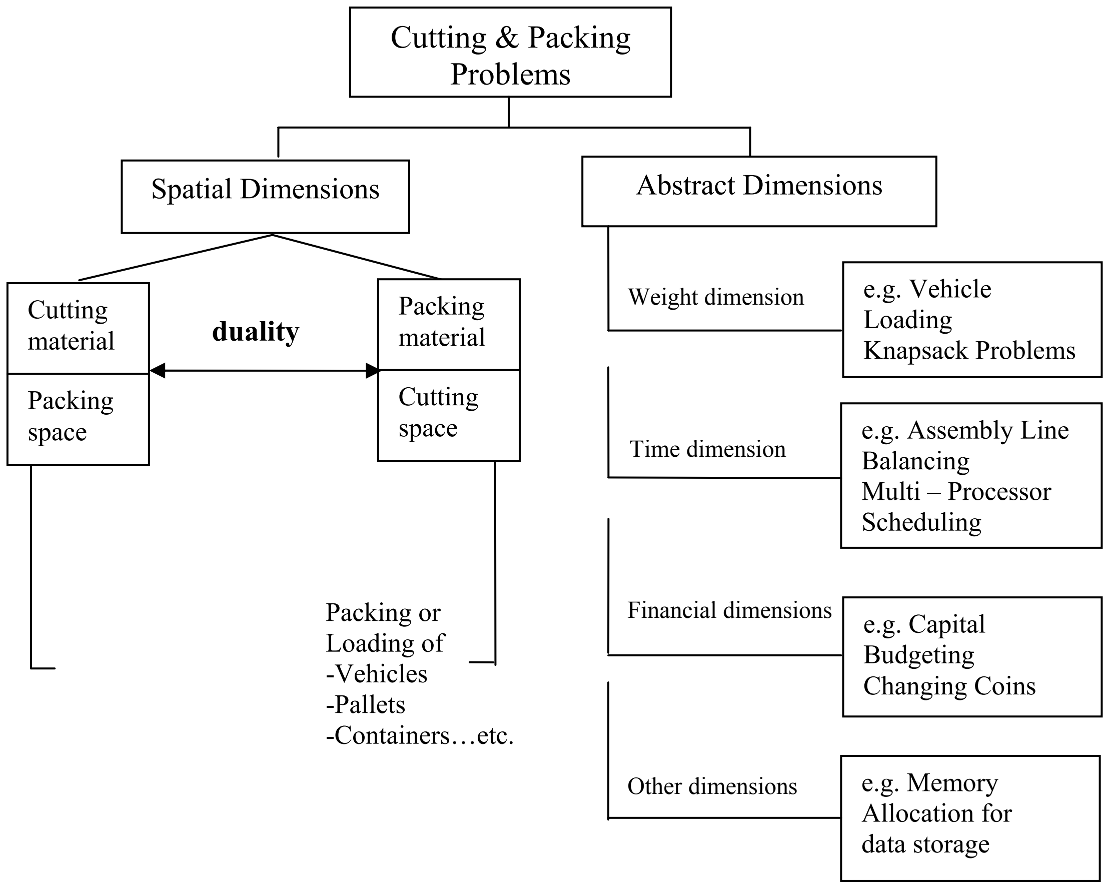

Cutting & Packing Problems
Spatial Dimensions
Abstract Dimensions
Cutting material
Packing space
duality
Packing material
Cutting space
Packing or Loading of -Vehicles -Pallets -Containers…etc.
Weight dimension
e.g. Vehicle Loading Knapsack Problems
Time dimension
e.g. Assembly Line Balancing Multi – Processor Scheduling
Financial dimensions
e.g. Capital Budgeting Changing Coins
Other dimensions
e.g. Memory Allocation for data storage

Figure 2.1. Overview of CPP (Dyckhoff, 1992)

Wäscher et al. (2007) gives a common definition for cutting and packing problems as:

Given are two sets of elements, namely

- A set of large objects (input, supply) and
- A set of small items (output, demand) which are defined exhaustively in one, two, three or an even larger number (n) of geometric dimensions. Select some or all small items, group them into one or more subsets and assign each of the resulting subsets to one of the large objects such that the geometric condition holds, i.e. the small items of each subset have to be laid out on the corresponding large object where,
- All small items of the subset lies entirely within the large object,
- The small items do not overlap,

and a given (single-dimensional of multi-dimensional) objective function is to be optimized.

In the light of this definition, they distinguish five sub-problems:

- Selection problem regarding the large objects,
- Selection problem regarding the small items,
- Grouping problem regarding the selected small items,
- Allocation problems regarding the assignment of the small items to large objects,
- Layout problem regarding the arrangement of the small items on each of the selected large objects with respect to the geometric condition.

In 2007, Wäscher and colleagues modified criteria for the definition of the problem types in the set of CPP. These are;

Dimensionality: one-, two-, three-, or more dimensional as previously defined by Dyckhoff.

Kind of assignment: based on the Dyckhoff's typology named as ***output (value) maximization*** and ***input (value) minimization***.

- ***output (value) maximization*** refers to the assignments of a set of small items to a set of large objects where the set of large objects is not sufficient to accommodate all items.

- ***input (value) minimization*** refers to the assignments of a set of small items to a set of large objects where the set of large objects is sufficient to accommodate all small items.

Assortment of small items: under this heading three cases are defined; **identical small items, a weakly heterogeneous assortment of small items and a strongly assortment of small items.**

Assortment of large objects: The use of **one large object** (all dimensions are fixed or one or more variable dimensions) or **several large objects** (identical large objects, weakly heterogeneous assortment, strongly heterogeneous assortment).

Shape of small items: **Regular small items** (rectangles, circles, boxes, cylinders, balls, etc.) and **irregular small items**.

According to these criteria, Wäscher et al. (2007) developed some basic, intermediate and refined problem types. Basic types of CPP are developed by taking into account **type of assignment** and **assortment of small items.** These basic types are illustrated in Figure 2.2.

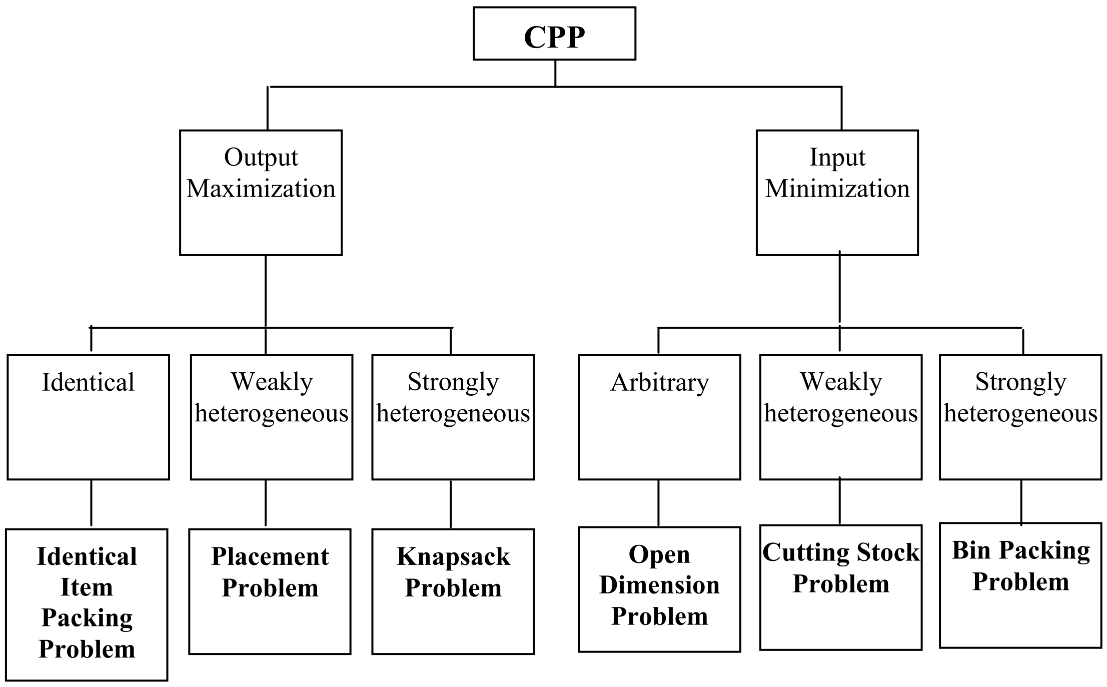


Figure 2.2. Basic problem types (Wäscher et al., 2007)

Intermediate problem types are developed by adding the criteria assortment of large objects to the basic problem types. **Figure 2.3** and **2.4** presents these intermediate problem types in terms of output maximization and input minimization.

Characteristics of the large objects / Assortment of the small items		identical	weakly heterogeneous	strongly heterogeneous
All dimensions fixed	one large object	**Identical Item Packing Problem** **IIPP**	**Single Large Object Placement Problem** **SLOPP**	**Single Knapsack Problem** **SKP**
	identical	▨	**Multiple Identical Large object placement problem** **MILOPP**	**Multiple Identical Knapsack Problem** **MIKP**
	heterogeneous	▨	**Multiple Heterogenous Large Object Placement Problem** **MHLOPP**	**Multiple Heterogenous Problem** **MHKP**

Figure 2.3. Intermedite problem types: output maximization (Wäscher et al., 2007)

Characteristics of the large objects / Assortment of the small items		weakly heterogeneous	strongly heterogeneous
All dimensions fixed	identical	**Single Stock Size Cutting Stock Problem** **SSSCSP**	**Single Bin Size Bin Packing Problem** **SBSBPP**
	weakly heterogeneous	**Multiple Stock Size Cutting Stock Problem** **MSSCSP**	**Multiple Bin Size Bin Packing Problem** **MBSBPP**
	strongly heterogeneous	**Residual Cutting Stock Problem** **RCSP**	**Residual Bin Packing Problem** **RBPP**
One large object variable dimension (s)		**Open Dimension Problem** **ODP**	

Figure 2.4. Intermediate problem types: input minimization (Wäscher et al., 2007)

According to this typology, Container Loading Problem (also called Single Container Loading) is a Placement Problem according to the Kind of Assignment and named

Single Large Object Placement Problem **(SLOPP).** Wäscher et al. (2007) states that this problem requires loading a fairly large, weakly heterogeneous consignment of boxes into a given container such that the volume or value of the packed boxes is maximized, or equivalently, the unused space of the container or the value of the unpacked boxes is minimized.

2.2. Container Loading Problems

CL and related problems such as the ones mentioned above have recently received considerable attention in the literature. There are several reasons of this popularity as reported by (Ertek and Kılıç, 2006). First of all, the CL problem is a NP-hard problem (Pisinger, 2002) and it has been recognized that it has a wide range of industrial applications. It is also possible to define new variants of CL problems by using different types of objective functions and the constraints as well. Therefore, there are many approaches (both heuristic and meta-heuristic based) proposed to solve CL problems, which are a sub-problem of Cutting and Packing problems.

CL problems can be defined as follows;

Given a set of n items with width (w_l), depth (d_i) and height (h_i) and a single container with known dimensions (W, D, H) where $w_l \leq W$, $h_i \leq H$ and $d_i \leq D$, the problem is to pack items into the container without overlapping while maximizing the utilization rate of the container. The utilization of the container is calculated in terms of the volumes of the allocated boxes. Suppose that;

U : Utilization of the container

V_p: Total volume of the allocated boxes into the container

V_c : Volume of the container where $V_c = W \times D \times H$

$$U = 100 \times \frac{V_p}{V_c} \qquad (2.1)$$

The problem is solved under the following assumptions:

(1) Items are rectangular boxes defined with known dimensions (w_l, d_i, h_i)
(2) Boxes are placed completely in the container
(3) Overlapping between the boxes is avoided
(4) Items can be rotated in any dimension if there is not a restriction defined (see Figure 2.5).

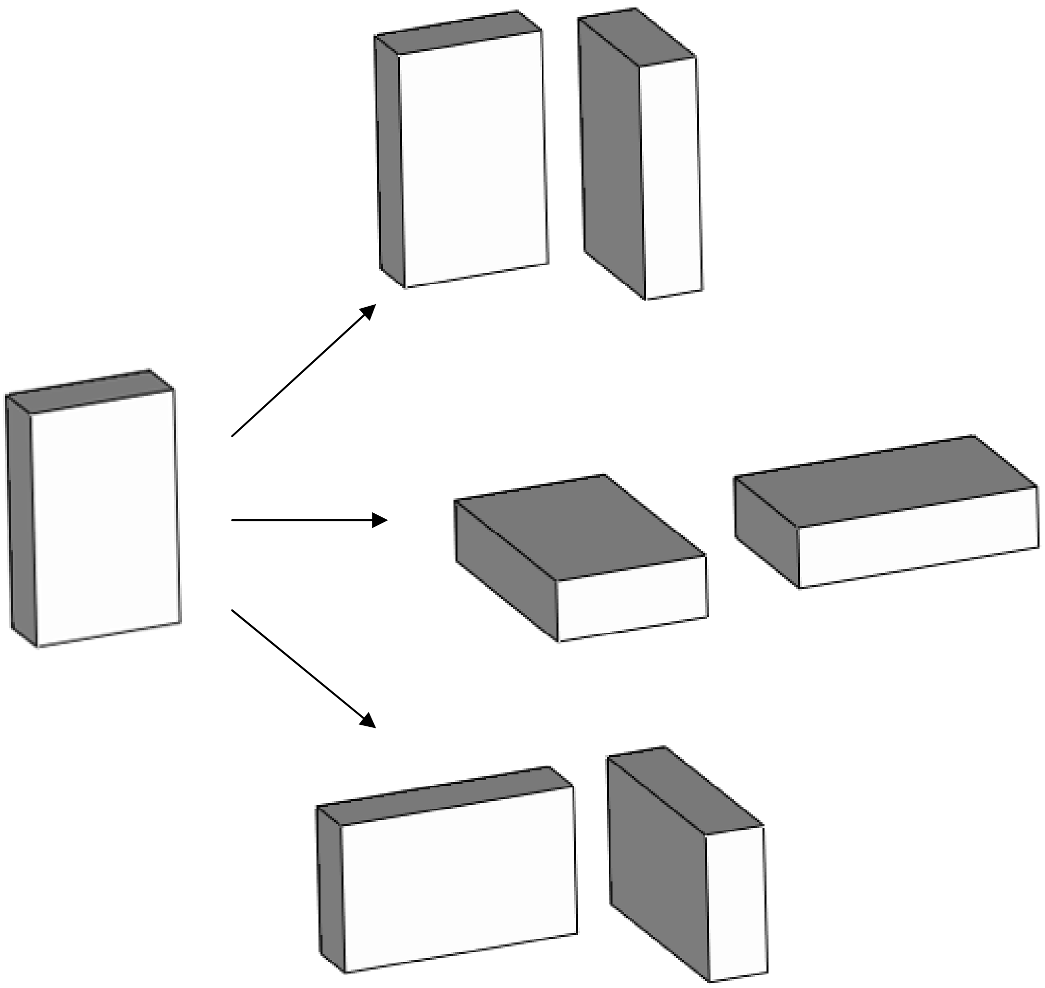


Figure 2.5. Six different rotation variants of a box

In addition to the above mentioned assumptions further constraints can be considered. Bischoff and Ratcliff (1995) discussed some factors that can be considered when solving the problem.

Orientation Constraints: Some items/boxes can have a transportation instruction such as “this way up”. Alternatively, some boxes can be placed in any of the six possible dimensions.

Load bearing strength of items: Another transportation instruction that can be encountered is “stack no more than x items high”.

Handling Constraints: The size or weight of an item and the loading equipment used may to some extent dictate the positioning within a container. For example, it may be necessary to put **large items on the** container floor or to restrict heavy ones to positions below a certain height. It may also be desirable from the viewpoint of easy/safe materials handling to place certain items near the door of the container.

Load Stability: The movement of the load should be avoided during the transportation if the cargo is of an easily damaged type. Straps, airbags and other devices can be used to prevent cargo movement.

Grouping of items: Items belonging to the same group for example, by a common recipient or the same type can be positioned in close proximity.

Multi-Drop situation: If a container is carrying cargo for a number of different destinations, items in the same consignment should be close together in the order of distribution to avoid unloading and reloading of a large part of the cargo several times.

Separation of items within a container: If the cargo compromises items that may adversely affect some of the other goods (both foodstuffs and perfumery articles), then the loading arrangement takes account of this.

Complete shipment of certain items: If the cargo is composed of sub-sets that may constitute functional entities (components for assembly into a piece of machinery) or may need to be treated as a single entity for administrative reasons, and if any part of such a sub-set is packed, then all the other items belonging to it are also to be included in the shipment.

Shipment priorities: If the shipment of some items is more important than all of the others, then this rating can represent a shipping priority such that no item in a lower priority class is shipped if this causes items with higher ratings to be left behind.

Complexity of the loading arrangement: Generally complex packing patterns results in a greater materials handling effort.

Container weight limit: If the total weight of the cargo is fairly high, the weight limit of a container may represent a more strict constraint than the loading space of the container.

Weight distribution within a container: From the transportation and handling point of view, it is desirable that its centre of gravity is close to the geometrical mid-point of the container floor. If the weight is distributed very unevenly, certain handling operations may be impossible to be carried out. In cases where a container is transported by road at some stage of its journey, the implications of its internal weight distribution for the axle loading of the vehicle can be an important consideration.

2.3. Conclusion

Lately, the CL problem is classified as the Single Large Object Placement Problem (SLOPP) in terms of the Kind of Assignment of the small items. Single Large Object Placement problems are defined under the type Output Maximization.

Single CL Problem is an example of the three-dimensional, rectangular SLOPP (Wäscher et al., 2007). According to Wäscher et al. (2007) published research concentrates on five problem types of CPP which are ODP (102 papers, 23%), SBSBPP (89 papers, 20%), SKP (86 papers, 19%), SLOPP (56 papers, 13%) and the SSSCSP (38 papers, 9%). Papers on these five problem types account for 371 out of 445 publications (83%). Among these papers focused on SLOPP, 19 out of 56 (34%) are concentrated on three-dimensional SLOPP problems.

In this study, the problem that is dealt is the Single CL problem (so called Container Loading Problem in the literature). In the first part of the study, the classical problem is handled without any additional constraints; however, in the second part a multi-objective CL problem is dealt.

CHAPTER 3

LITERATURE REVIEW ON CONTAINER LOADING PROBLEMS

3.1. Literature on Container Loading Problems

Many approaches have been proposed to solve CL problems along with many practical constraints and different objective functions. Most of the published work on the subject utilizes different types of data structures such as graphs or trees (Morabito and Arenales, 1994; Eley, 2002, Lim et al., 2005), heuristics algorithms (George and Robinson, 1980; Bischoff and Marriott, 1990; Gehring et al., 1990; Haessler and Talbot, 1990; Ngoi et al., 1994; Pisinger, 2002; Bischoff, 2003; Moura and Oliveira, 2005), meta-heuristic algorithms, such as Genetic Algorithms (GAs) (Gehring and Bortfeldt, 1997; Bortfeldt and Gehring, 2001;Yeung and Tang, 2005), Simulated Annealing (SA) (Faina, 2000, Mack et al., 2004) and Tabu Search (TS) (Bortfeldt and Gehring,1998) to solve different variants of the problem. Also a few parallel approaches, including a parallel GA (Gehring and Bortfeldt, 2002), a parallel TS (Bortfeldt et al., 2002, Mack et al., 2004), a parallel SA (Mack et al., 2004) and a parallel hybrid local search meta-heuristic (Mack et al., 2004) are available.

CL problems are NP-hard problems (Pisinger, 2002). Due to this fact, there are few exact approaches (Chen et al., 1995; Li et al., 2003). Most of the published work on the subject utilizes different types of data structures, such as graphs and trees, heuristics algorithms and meta-heuristic algorithms, such as Genetic Algorithms (GAs), Simulated Annealing (SA) and Tabu Search (TS) to solve different variants of the problem. An overview of solution approaches for the CL problems is presented in Table 3.1.

The publications on CL problems are examined according to specific criteria such as:

- **objective(s),**
- **constraint(s):** No Constraints ***(NC),*** Constraints included ***(CI)***
- **number of containers (bins)** that is used in the solution: Single ***(S),*** Multiple ***(M)***
- **type of the boxes** that are allocated: Homogeneous ***(H),*** Weak Heterogeneous ***(WHe),*** Strong Heterogeneous ***(SHe)*** and
- **the solution approach** that is employed.

Following Table 3.1., the published literature on CL problems is examined.

Table 3.1. Reviewed works in the literature

Authors (Year)	Type of boxes			No of bin(s)		Objective (s)	Constraint		Solution approach
	H	WHe	SHe	S	M		NC	CI	
George, Robinson (1980)		X		X		Maximize the box volume accommodated	X		Heuristic Algorithm
Bischoff, Marriott (1990)	X	X	X	X		Minimize the container length needed to accommodate the box	X		Heuristic Algorithms
Gehring et al. (1990)			X	X		Minimize inevitable space		X	Heuristic Algorithm
Haessler, Talbot (1990)		X		X		Maximize the box volume accommodated		X	Heuristic Algorithm
Morabito, Arenales (1994)		X	X	X		Maximize the box volume accommodated		X	AND/OR Graph approach
Ngoi et al.(1994)		X		X		Maximize the usage of space for a fully packed container	X		Heuristic Algorithm
Chen et al. (1995)		X			X	Minimize unused space by selecting a number of containers to pack all the boxes			Zero—One Mixed Integer Programming
Gehring, Bortfeldt (1997)		X	X	X		Maximize the box volume accommodated		X	Genetic Algorithm
Chien, Wu (1998)	-	-	-	X		Minimize waste of container space	X		Dynamic programming based recursive algorithm
Faina (2000)		X	X	X		Minimize the used container height used to pack the box	X		Global optimization algorithm based on Simulated Annealing
Martello et al. (2000)		X	X	X	X	Packing all items into the minimum number of bins	X		Branch and bound algorithm
Terno, et al. (2000)		X	X	X	X	To find the minimum number of pallets to load the whole consignment			Heuristic with branch and bound framework
Bortfeldt, Gehring (2001)			X	X		Maximize stowed box volume		X	Hybrid Genetic Algorithm
Lodi et al. (2002)		X	X		X	Minimize the number of bins that contains all the items	X		Heuristics and Hybrid Tabu Search Algorithm

constraints dealt: No Constraints ***(NC),*** Constraints included ***(CI)***
number of containers: Single Container ***(S),*** Multiple Containers ***(M)***
type of the boxes: Homogeneous ***(H),*** Weak Heterogeneous ***(WHe),*** Strong Heterogeneous ***(SHe)***

Table 3.1. Reviewed works in the literature (continues)

Authors (Year)	**Type of boxes**			**No of bin(s)**		**Objective (s)**	**Constraint**		**Solution approach**
	H	**WHe**	**SHe**	**S**	**M**		**NC**	**CI**	
Eley (2002)		**X**	**X**	**X**	**X**	Multiple objectives (treated singularly) - volume utilization - load stability - weight distribution	**X**		Tree search
Pisinger (2002)	**X**	**X**	**X**	**X**		Maximize the box volume accommodated	**X**		Heuristic Algorithm
Gehring, Bortfeldt (2002)			**X**	**X**		Maximize the box volume accommodated	**X**	**X**	Parallel Genetic Algorithm
Bortfeldt et al. (2003)		**X**		**X**		Maximize stowed box volume		**X**	Parallel Tabu Search Algorithm
Li et al. (2003)		**X**		**X**		Packing box into a rectangular container having minimal space	**X**		Zero—One Mixed Integer Programming
Bischoff (2004)		**X**		**X**		Maximize the box volume accommodated		**X**	Heuristic Algorithm
Mack et al. (2004)		**X**		**X**		Maximize the stowed box volume	**X**		Parallel Hybrid Local Search
Moura, Oliveira (2005)		**X**	**X**	**X**		Minimize wasted space in the container		**X**	GRASP Algorithm
Lim et al. (2005)	**X**	**X**	**X**	**X**		Maximize the box volume accommodated		**X**	Tree search heuristic
Yeung, Tang (2005)		**X**	**X**	**X**		Minimize the used container height used to pack the box		**X**	Hybrid Genetic Algorithm
Nepomuceno et al. (2007)		**X**	**X**	**X**		The maximum volume of the loaded boxes	**X**		Integer Linear Programming and Genetic Algorithm
Liang et al. (2007)		**X**		**X**		Determine the arrangement of objects with the best utilization ratio in the container	**X**		A hybrid meta-heuristic based on Ant Colony optimization and Genetic Algorithms
Wang, Li (2007)	X			**X**		Maximizing the number of boxes that can be loaded into the single container.	**X**		Heuristic Algorithms
Wang et al. (2008)		**X**		**X**		Determine a loading scheme that will maximize the space usage of the container	**X**		Tree based Heuristic Algorithm
Huang, He (2009)		**X**	**X**	**X**		Maximize the volume of the packed items	**X**		Heuristic Algorithm
Parreno et al. (2010)		**X**	**X**	**X**		Maximization of space usage		**X**	Heuristics, Variable Neighborhood Search
Kang et al. (2010)		**X**	**X**	**X**	**X**	Single Container Problem: maximizing the use of container's volume Multiple Container: minimizing the number of containers	**X**		Heuristic Algorithm

constraints dealt: No Constraints ***(NC),*** Constraints included ***(CI)***
number of containers: Single Container ***(S),*** Multiple Containers ***(M)***
type of the boxes: Homogeneous ***(H),*** Weak Heterogeneous ***(WHe),*** Strong Heterogeneous ***(SHe)***

3.1.1. Exact Approaches

As mentioned before, the studies on exact solution approaches are very limited. The first analytical work on CL problems is presented by Chen et al. (1995). They developed a zero – one mixed integer programming model for the general three dimensional CL problems. This model is presented here in order to demonstrate the complexity of the problem

If *x*, *y,* and *z* is denoted as the width, length, and height of the container (x>0, y>0, z>0), then the packing optimization problem is stated as follows (Chen et al., 1995):

Minimize xyz
subject to;
(1) All of *n* boxes are non-overlapping.
(2) All of *n* boxes are within the range of *x*, *y,* and *z*.
(3) $x^m \leq x \leq x^M$, $y^m \leq y \leq y^M$, and $z^m \leq z \leq z^M$
(x, y, and z are integers and x^m, y^m, z^m, x^M, y^M, *and* z^M are constants).

The related terminologies notations used in the packing model are;

(p_i, q_i, r_i) : Parameters indicating the length, width, and height of carton *i*.
(x, y, z) : Continuous variables indicating the length, width, and height of the container.
(x_i, y_i, z_i) : Continuous variables (for location) indicating the coordinates of the front-left-bottom corner of carton *i*.

(l_{xi}, l_{yi}, l_{zi}): Binary variables indicating whether the length of carton *i* is parallel to the *X-, Y-, or Z*-axis. For example, the value of l_{xi} is equal to 1 if the length of carton *i* is parallel to the *X*-axis; otherwise, it is equal to 0. It is clear that $l_{xi}+ l_{yi}+ l_{zi} =1$.

(w_{xi}, w_{yi}, w_{zi}) : Binary variables indicating whether the width of carton *i* is parallel to the *X-, Y-, or Z*-axis. For example, the value of w_{xi} is equal to 1 if the width of carton *i* is parallel to the *X*-axis; otherwise, it is equal to 0. It is clear that $w_{xi}+ w_{yi}+ w_{zi} =1$.

(h_{xi}, h_{yi}, h_{zi}) : Binary variables indicating whether the height of carton i is parallel to the *X*-, *Y*-, *or Z*-axis. For example, the value of h_{xi} is equal to 1 if the height of carton i is parallel to the *X*-axis; otherwise, it is equal to 0. It is clear that $h_{xi}+ h_{yi}+ h_{zi} =1$.

For a pair of cartons (i,k) where $i<k$, there is a set of 0–1 vectors ($a_{ik}, b_{ik}, c_{ik}, d_{ik}, e_{ik}, f_{ik}$) defined as;

a_{ik}= 1 if carton i is to the left of carton k, otherwise $a_{ik}=0$.
b_{ik}= 1 if carton i is to the right of carton k, otherwise $b_{ik}=0$.
c_{ik}= 1 if carton i is behind carton k, otherwise $c_{ik}=0$.
d_{ik}= 1 if carton i is in front of carton k, otherwise $d_{ik}=0$.
e_{ik}= 1 if carton i is below carton k, otherwise $e_{ik}=0$.
f_{ik}= 1 if carton i is above carton k, otherwise $f_{ik}=0$.

The front-left-bottom corner of the container is fixed at the origin. The packing problem can then be formulated as follows (Chen et al., 1995):

Minimize xyz (3.1)

subject to;

$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \le x_k + (1 - a_{ik})M$ for all $i, k, i<k$ (3.2)

$x_k + p_k l_{xk} + q_k w_{xk} + r_k h_{xk} \le x_i + (1 - b_{ik})M$ for all $i, k, i<k$ (3.3)

$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \le y_k + (1 - c_{ik})M$ for all $i, k, i<k$ (3.4)

$y_k + p_k l_{yk} + q_k w_{yk} + r_k h_{yk} \le y_i + (1 - d_{ik})M$ for all $i, k, i<k$ (3.5)

$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{zi} \le z_k + (1 - e_{ik})M$ for all $i, k, i<k$ (3.6)

$z_k + p_k l_{zk} + q_k w_{zk} + r_k h_{zk} \le z_i + (1 - f_{ik})M$ for all $i, k, i<k$ (3.7)

$a_{ik} + b_{ik} + c_{ik} + d_{ik} + e_{ik} + f_{ik} \ge 1$ for all $i, k, i<k$ (3.8)

$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \le x$ for all $i, k, i<k$ (3.9)

$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \leq y$ for all $i, k, i<k$ (3.10)

$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{z\,i} \leq z$ for all $i, k, i<k$ (3.11)

$l_{xi} + l_{yi} + l_{zi} = 1$ for all i (3.12)

$w_{xi} + w_{yi} + w_{zi} = 1$ for all i (3.13)

$h_{xi} + h_{yi} + h_{zi} = 1$ for all i (3.14)

$l_{xi} + w_{xi} + h_{xi} = 1$ for all i (3.15)

$l_{yi} + h_{yi} + w_{yi} = 1$ for all i (3.16)

$l_{zi} + w_{zi} + h_{zi} = 1$ for all i (3.17)

where $l_{xi}, l_{yi}, l_{zi}, w_{xi}, w_{yi}, w_{zi}, h_{xi}, h_{yi}, h_{zi}, a_{ik}, b_{ik}, c_{ik}, d_{ik}, e_{ik}$ and f_{ik} are 0–1 variables, $M = \max\{ x^M, y^M, z^M \}$, $x_i, y_i, z_i \geq 0$, $0 < x^m \leq x \leq x^M$, $0 < y^m \leq y \leq y^M$, $0 < z^m \leq z \leq z^M$, x, y, and z are integers, and $x^m, y^m, z^m, x^M, y^M,$ *and* z^M are constants. The objective of this model is to minimize the volume of the container. Constraints (3.2)–(3.8) are non-overlapping conditions used to ensure that none of these *n* boxes overlap each other. Constraints (3.9)–(3.11) guarantee that all boxes are within the enveloping container. Constraints (3.12)–(3.17) describe the allocation restrictions among logic variables. For instance, constraint (3.12) implies that the length of carton *i* is parallel to one of the axes. Constraint (3.15) implies that only one of length, width and height of carton *i* is parallel to *X*-axis.

The developed model takes into account the issues of carton orientations, overlapping of cartons, multiple carton sizes and multiple container sizes. They also extended the model for some special container loading problems. Although the model reaches an optimum solution, it takes fifteen minutes to solve a small scale problem in which the objective is to allocate six non-identical boxes to three non-identical containers. Unfortunately, using this model for the real-world problems is not practical since the number of variables increase greatly as the number of boxes increase.

Martello et al. (2000) proposed an exact branch and bound method for the 3D BPP. The algorithm iteratively solves sub problems which have to fill a single bin. For this purpose, a procedure called *main branching tree* assigns items to the bin and a branch and bound algorithm called *onebin* checks whether those items can fit in a single bin. If the items can fit in a single bin, the algorithm tries to obtain the best filling. The computational work that is presented shows that all the problems up to 30 items and %84 of problems up to 50 items are solved to optimality. However, the performance of the algorithm decline as the number of items that should be located into a single bin increases.

The later study of Li et al. (2003) extended the zero – one mixed integer programming model of Chen et al. (1995) by reducing the number of variables in the mathematical model. The original model proposed by Chen et al. (1995) uses $3n(n-1)+9n$ 0-1 variables where as the reformulated model uses $\frac{3}{2}n(n-1)+9n$ 0-1 variables. In spite of this improvement in the model, it is not clear how many boxes can be allocated to a container using this model within a reasonable amount of time.

3.1.2. Heuristic Approaches

Most of the approaches proposed so far are heuristic algorithms. The most common heuristic approaches can be classified as;

- Wall building algorithms (utilizing layers),
- Stack building algorithms,
- Guillotine cutting algorithms and
- Cuboid arrangement algorithms (Pisinger, 2002).

One of the earliest publications on CL was published by George and Robinson (1980). They proposed a wall building algorithm. Their heuristics packing algorithm pack a set of non- identical boxes into a container where the total volume of boxes is little less than the volume of the container. The proposed algorithm fills the container by building layers across the container width. The depth of each layer is determined by the box that has the highest rank Layers are produced from boxes of the same type. The empty spaces that are occurred in layers and in between the layers are filled with the remaining items. Empty spaces in between layers are combined and filled with an unused box. However, later work indicated that this algorithm does not produce very efficient patterns (Bischoff and Ratcliff, 1995).

Bischoff and Marriott (1990) developed fourteen heuristic algorithms by combining six ranking rules and three filling methods based on George and Robinson's and Bischoff and Dowsland's approach. The developed algorithms (namely B1 and B2) have two main differences from the George and Robinson's approach. Each layer is built from a single type of box and each layer is filled through a two dimensional packing procedure. The depth of each layer is determined by a heuristic approach. Each dimension of a box in turn is accepted as a potential layer depth. With this depth fixed, the number of rectangles that fills this layer is calculated. If a full layer cannot be formed with the number of boxes that should be loaded, this depth is unsuitable. The algorithm checks the other alternatives. If more than one possible layer depth occurs (a complete layer can be formed with this dimension), then a choice needs to be done. Either the dimension that yields the maximum percentage fill of the layer (B1) or the dimension that leaves the least number of items (B2) can be selected. The comparison of fourteen heuristics suggests that the performance of such heuristics is problem dependent. That is, each algorithm performs different for each set of problems.

The approach proposed by Haessler and Talbot (1990) is based on the idea of forming stacks from boxes. Their purpose is to arrange order quantities and form a loading plan for ordered products. For this purpose, they first estimate the number of stacks that can be loaded into a vehicle. Then, they form the stacks with the suitable boxes (low density products). Finally, they place these stacks across the container.

Later, Gehring et al. (1990) developed a heuristic algorithm utilizing the wall building philosophy of George and Robinson's approach and Haessler and Talbot's (1990) approach. Similar to George and Robinson's approach, the container is filled by building layers across the container width. The depth of each layer is decided by the layer determining box. Alternative loading patterns are obtained by chancing the dimensions of the layer determining box or changing the layer determining box.

Empty spaces in each layer are filled by the suitable box having the highest volume. Different from the George and Robinson’s approach, empty spaces in between layers cannot be combined. The algorithm produces alternative loading plans that can be selected by the decision-maker.

Morabito and Arenales (1994) proposed a guillotine cutting algorithm which uses a slicing tree. The algorithm slices the container into smaller parts using guillotine cuts. Thus, the initial node of the tree corresponds to the container and the leaf nodes correspond to the boxes. They argued that solving the problem by stacks was better than solving it by layers but their algorithm produced better solutions than these two.

The paper presented by Ngoi et al. (1994) utilizes spatial representation techniques to solve the 3D container loading problem. Their algorithm determines the empty spaces and compares the volume of unpacked boxes within these spaces. The box that gives the least amount of leftover space is selected. Then the packed boxes are updated into the spatial representation system. (the approach uses some ranks to determine the best suitable box).

Chien and Wu (1998) proposed a (dynamic programming based) computational procedure for the 3D container loading problem. The procedure reduces the problem to two dimensional, and one dimensional case, respectively. For this purpose, they first cut the container volume into a layer along length, width and height. Then they cut each of the layers into horizontal and vertical strips. The best solution is one of the three different cutting patterns. However, their work is completely theoretical. They did not present any computational work to evaluate the performance of the proposed approach.

Terno et al. (2000) developed a branch and bound based heuristics for the multi pallet loading problem. They utilized layers to pack the cargo but they used vertical layers which does not seem very practical for container loading. First, they developed a splitting procedure to partition the whole cargo into *k* pallets. Afterwards they used some loading strategies to load the pallets. These are G4 heuristic to load identical pieces, M4 heuristic to load pieces with same heights or height combination (at most 4 pieces) or M4 and M8 to load the rest of the items. The results presented in the study reveals that the performance of this approach outperforms some previously offered CL algorithms by Ngoi et al. (1994) and Gehring et al. (1997). Although offered for multi-pallet loading problem, the authors argued that their algorithm is also suitable for the CL problems.

Eley (2002) used a different filling approach to deal with the single and multiple container loading problems under stability and weight distribution constraints. Eley neither used layers nor towers but used homogeneous blocks (similar to towers) made up of identical items. His approach is based on a greedy heuristic to form the blocks and a tree search procedure to improve the solution. The results of the benchmark problems indicate that the algorithm can compete with CBGAT and CBGAS but remains poor compared to TS approach of Bortfeldt and Gehring. He also proposed a simple methodology in order to obtain an even weight distribution within the container. First the container is filled with nonstraddling walls across the width of the container. Then an even weight distribution is obtained by chancing the places of the walls along the length of the container.

Pisinger (2002) proposed a heuristic algorithm based on wall building approach. First, he formed layers and strips in each layer. He determined the layer depth and strip width using a tree search algorithm. Then, he filled each strip using a knapsack algorithm. He also searched the effects of different ranking rules for the selection of layer depths and strip widths. He observed that a compromise between the largest box dimension and the most frequent dimension leads to a high solution quality. He pointed out that the filling ratio of his algorithm is about %95 which is high compared to Gehring, Bortfeldt (1997) (%87.7), Morabito and Arenales (1994) (around %95).

An algorithm based on greedy randomized search procedure (GRASP) is presented by Moura and Oliveira (2005). The newly proposed algorithm; first builds a solution using the improved version of the George and Robinson’s heuristic which is based on wall building algorithm, then the solution is improved with a local-search algorithm. More, the authors tackle the cargo stability issues in the algorithm as a constraint in the construction phase of the algorithm. They obtained an average of 86.74% for the test problems.

Lim et al. (2005) used a different approach to deal with the container loading problems. They developed a basic heuristic and two augmenting heuristics based on the tree search. The basic heuristic packs a box into the container. Then, the generated empty spaces as a result of this packing are accepted as the root of a tree and each are packed with the suitable boxes.

Wang, Li (2007) proposed two heuristic approaches to pack homogeneous boxes into a single container. Both algorithms are based on layers on the faces of the container. In the first approach, layers are built on the selected layer face whereas in the second approach the algorithm selects the layer face dynamically according to the rameining container space as a result of the previously filled layer. Boxes are filled to the layers by a block-based 2D packing procedure.

Wang et al. (2008) used a tertiary-tree model for weakly heterogeneous CL problems. First, they placed block of homogeneous boxes into the container. Afterwards, they applied a dynamic space decomposition method to the remaining container space. To reach a high ratio, they used an optimal-fitting sequencing and an inner-right corner occupying action.

Huang and He (2009) proposed a heuristic approach in which the key issue is to pack an item into a corner or even a cave in the container such that the item is packed as compactly and closely to the other items as possible. Tested on some of the test cases from the literature, the proposed heuristics performs quiet well compared to the other approaches in the literature.

Recently, Parreno and colleagues (2010) proposed a new heuristic algorithm based on variable neighborhood search. The heuristic uses several new neighborhoods based on the elimination of layers, insertion of columns or boxes and a stronger move based on emptying a region of the container. The experiments with the test cases showed that the VNS algorithm competes favorably with the best performing algorithms. They also dealt with cargo stability aspects and compared their algorithm with some works dealing with cargo stability such as Eley (2002), Moura and Oliviera (2005)…etc.

A new block strategy is proposed both the Single Container and the Multiple Container Loading Problem by Kang et al. (2010). They used blocks made up of homogeneous boxes to fill the container. Using these blocks first the container is build recursively until all the boxes are stowed or no empty space is left behind. By replacing the previously placed blocks with the alternative blocks they generated alternative packing patterns and compared these patterns with each other to find the best packing patterns. The performance of this strategy is quite high even compared to the algorithm offered by Parreno and colleagues (2010).

3.1.3. Meta-heuristic Approaches

Gehring and Bortfeldt (1997) presented a Genetic Algorithm (GA) for the 3D container loading problem (CBGAT) in which a set of constraints related with weight and stability aspects are taken into account. The algorithm fills the container in two steps. First, a set of stable box towers is generated by a greedy algorithm. Then the container floor is covered by this box towers using GA. They suggested that this algorithm achieves high container utilization for both weak and strong heterogeneous problems.

Faina (2000) introduced a geometric model which reduces the general 3D packing problem to a finite enumeration scheme. He developed a Simulated Annealing based algorithm that uses the method of zones. Different from the previous algorithms, the proposed algorithm neither uses a ranking rule nor a packing strategy, such as wall building, stack building, etc. In this method each box is defined as zones. The algorithm starts packing by placing the first box to the origin. Then the second box moves in the direction of decreasing z dimension, than in the direction of decreasing x dimension and finally in the direction of decreasing y dimension, until it touches the border of the zone of the first box. After obtaining an initial solution in this manner, the algorithm performs a small perturbation on this initial solution and constructs a new solution. The algorithm provides high quality results up to 32 boxes, but the solution quality gets worse as the number of boxes increases.

In 2001, Bortfeldt and Gehring proposed another GA (named CBGAS) for the same problem but this time utilized layers to fill the container. Similar to Gehring et al's

(1990) approach, each solution is composed of non-overlapping vertical layers along the container. Using GA and problem-specific GA operators, the best sequence of each of the formed layer is determined. However, the results of the benchmark problems does not indicate a slight performance difference between the two GA developed by Bortfeldt and Gehring. The presented algorithm (CBGAS) seems more suitable for the strong heterogeneous problems. This finding supports the idea that the quality of heuristic algorithms for 3D container loading problems is usually problem dependent.

Gehring and Bortfeldt (2002) offered the parallel version of the GA which was published previously in 1998 by the same authors. Since the relevant article is written in German, it is not possible to discuss the details of this algorithm. Different designs for the parallelization were offered and one of them was chosen. With the parallelized algorithm an improvement of 0.7% is obtained compared to the original algorithm.

Lodi et al. (2002) proposed a two phased constructive heuristic for the 3D Bin Packing Problems (3D BPP). 3D BPP have relevant practical interest in industrial applications such as, e.g., cutting foam rubber in arm-chair production, container and pallet loading and packing design (Lodi et al., 2002). Therefore, this study is also reviewed here. The proposed two-phased heuristic packs the items by layers. The layers are filled either by items which are sorted in non-increasing height (Phase 1) or by items which are sorted by non-increasing area (Phase 2). One important point is that, Phase 2 use the layers produced by Phase 1. Two solutions are obtained as a result of the two phases. To obtain both solutions, produced layers are combined into finite bins using the 1D BPP algorithm. Finally, the better of two solutions is chosen. Later, this constructive heuristic called HA is embedded into the TS algorithm. When compared to H1 and H2 heuristics and exact algorithm BB of Martello et al. (2000) and constructive heuristic HA, TS with HA produce better solutions. When compared to GLS algorithm by Fareo et al.(), for some cases TS with HA performs better and for some cases GLS performs better.

Bortfeldt et al. (2003) used Tabu Search (TS) algorithm to solve the weakly heterogeneous 3D container loading problem. They load the container with what they called ***local arrangements,*** which are predefined box arrangements. They mainly investigated the effect of parallel computing on solution quality. For this purpose, they programmed a modular algorithm with two TS algorithms, one of which is a sequential one and the other being a parallel one.

Mack et al. (2004) proposed a hybrid meta-heuristic for the CL problem which combines a SA algorithm with a TS algorithm. The SA algorithm has been transformed into a hybrid meta-heuristic by post-proccesing the final solution obtained from SA with TS. The authors had also offered the parallel versions of all these algorithms. When the offered SA and TS algorithms are compared it is found out that SA algorithm yields better solutions. However, the average computational time of SA increases significantly compared to TS. The results revealed that parallelization and hybridization gives the best solution quality. With a 93.78% filling ratio (for Bischoff and Ratcliff cases which will be introduced later) their results dominated the results of the other authors in the literature.

Yeung and Tang (2005) hybridized the GA with a new heuristic filling strategy that is able to produce stable solutions. The heuristic filling algorithm packs a sequence of boxes using vertical layers. These layers can be regular or irregular in shape. Using GA they obtained the best placement sequence.

Nepomuceno et al. (2007) proposed a hybrid approach based on Integer Linear Programming and Genetic Algorithms. The hybrid approach has two components; Generator of Reduced Instances and the Decoder of Reduced Instances. The first component is in charge of producing reduced problem instances of the problem while the second component is responsible to interpret and solve any generated problem instances coming out of the first component. The optimal solution is achieved with solving the sub problems of the original problem in an iterative manner.

Lately, Liang et al. (2007) proposed a hybrid meta-heuristic algorithm based on Ant Colony Optimization and Genetic Algorithms. This study is probably one of the few approaches using Ant Colony Optimization for CL problems. In the first phase of the method, tower sets made up of objects are constructed with the pheromone updating structure of the ant colony optimization algorithms. Following this construction phase, towers are assigned to the container's bottom plane with Genetic Algorithm. The utilization rate obtained with the method was satisfactory however it was not better than results obtained by Mack et al. (2004).

3.2. Literature on Swarm Intelligence Techniques

SI as a term was first used in 1988 by Gerardo Beni (Beni, 1988) in the context of cellular robotic systems. As Bonabeau (1999) stated, SI describes any attempt to design algorithms or distributed problem-solving devices inspired from the collective behavior of social insect colonies and other animal societies. SI indicates a recent computational and behavioral metaphor for solving distributed problems that originally took inspiration from the biological examples provided by the social insects and by swarming, flocking and herding behaviors in vertebrates (Zhao et al., 2006).

Social insects have lived on Earth for millions of years, building nests and more complex dwellings, organizing production and procuring food (Teodorovic, 2003). SI algorithms draw inspiration from the problem-solving ability of social insects that live in colonies, such as ants, bees, wasps, termites. These insects interact with each other in various ways including bee dancing for food foraging, ants laying pheromone to the path, etc. This kind of communication systems between individual insects shows the connection between '***individual insect behaviour'*** and '***collective intelligence'*** of social insect colonies (Teodorovic, 2003; Bonabeau, 1999). Another interesting feature of social insects is their ***self organization capability***. When acting as a community, these insects even with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. Problems like finding and storing foods, selecting and picking up materials for future usage require a detailed planning and are solved by insect colonies without any kind of supervisor or controller (Abraham et al., 2008).

Although there are many animals or colonies available in the real world in order to mimic, two main types of SI algorithms can be found in the literature, namely; Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) (Engelbrecht, 2005; Kennedy et al., 2001; de Castro, 2002). An increasing number of researchers have also implemented Bee(s) Algorithms (BA) which imitates the foraging behavior of swarms of honey bees to solve a variety of diverse real-world problems (Dereli et al., 2009).

It is worth noting that some algorithms like Genetic Algorithms (GAs) and Stochastic Diffusion Search (SDS) are occasionally considered in the family of SI, although they are not inspired by the behavior of social insect colonies and other animal colonies. Other algorithms that can occasionally be classified under SI; like: (Dereli et al., 2009)

- Stochastic Diffusion Search (SDS)
- Bacteria Swarm Foraging Optimization (BSFO) or Bacteria Foraging Optimization Algorithm (BFAO)
- Artificial Immune System (AIS)
- Carabid Beetle Foraging
- Wasp or Wasp Colony Algorithm
- Physarum Solver

The most commonly used technique among these methods inspired by social insects is the ACO algorithm which was conceived by Dorigo et al. (1991). The algorithm can be described as an evolutionary search procedure based on the way that ant colonies cooperate in locating shortest routes to food sources. Ants are social insects, that is, insects that live in colonies and whose behavior is directed more to the survival of the colony as a whole than to that of a single individual component of the colony (Dorigo et al., 1999). The specific interest of researchers on ant colonies is their foraging behaviour and how they can find the shortest path between food sources and the nest. Ants communicate among themselves through a chemical substance called “pheromone”, which they lay on the ground along the path they traverse. It has been observed that the more ants use a particular path, the more pheromone is deposited on that path and the more it becomes attractive to the other ants seeking food. If an obstacle is suddenly placed on an established path leading to the food source, ants will initially go right or left in a seemingly random manner. Those choosing the side that is in fact shorter will reach the food more quickly and will make the return journey more often. The pheromone that is deposited on the shorter path will eventually become the preferred route for the stream of ants (Gravel et al., 2002).

Besides ACO, another population based optimization technique, which is relatively new, is the Particle Swarm Optimization (PSO) algorithm. It was developed by Eberhart and Kennedy in 1995 (Kennedy and Eberhart, 1995), inspired by the social behavior of bird flocking or fish schooling. The particle swarm concept originated as the simulation of a simplified social system. The original intent was to graphically simulate the choreography of a bird of a bird block or a school of fish. However, it was found that particle swarm model can be used as an optimizer. The PSO algorithm includes a swarm of particles moving in the n- dimensional problem space where each particle represents a potential solution having a fitness function that is to be optimized. Each particle in the swarm has a *position* and a *velocity* which is updated both by its own (*pbest*) and neighbours experience (*gbest*) in the search space. In analogy with evolutionary computation paradigms, a swarm is similar to a population, while a particle is similar to an individual (Engelbrecht, 2006).

Bee(s) algorithm (BA) is the youngest algorithm compared to ACO and PSO. Numerous researchers have recently been inspired from the interesting features of honey bee colonies. It is a well-known fact that if only some of the nature or behaviour of honeybees can be exploited and some new characteristics could be added, a class of algorithms can be devised (Yang, 2005). Due to this fact, considerable research has been conducted to develop algorithms that mimic the foraging, learning, mating and dancing behaviours of the honeybees. It has been also reported that these algorithms - namely bee(s) or bee colony algorithms are mainly inspired from two behaviors: (Abbass, 2001; Afshar et. al, Teo and Abbass, 2003; Koudil et. al, 2007) and food foraging (Lucic, 2002; Yang, 2005; Pham et. al, 2006) (Dereli and Daş, 2007). Most of the works in this field of research have been mainly affected from (or based on) the pioneering works of von Frisch (1967), Seeley (1995) and Dyer (2002). In utilization of the proposed models in this field, a number of algorithms based on the behaviors of honey-bee colonies have been developed by different researchers. It has been recently recognized that honeybees can manage to efficiently collect the best nectar without any central command and the swarm intelligence of these amazingly organized bees can also be used for optimization problems (Nakrani and Tovey, 2007).

Despite its relatively short history, these approaches that have been inspired from the behaviors of the honey bees have been applied to job shop scheduling (Chong et.al, 2006), transportation problems (Lucic, 2002), partitioning and scheduling problems (Koudil et. al, 2007), training of multi-layered perception networks (Pham et. al, 2006a), recognizing patterns in control charts (Pham et. al, 2006b), optimization of continuous functions (Karaboğa and Baştürk, 2007; Pham et. al, 2006c), water resources management problems (Bozorg and Afshar, 2004), data mining problems (Fathian et al., 2007; Benatchba et al., 2005) and to generalized assignment problems (Baykasoglu et al., 2007).

Among the above introduced SI based algorithms, two swarm-based techniques namely; ACO and BA are applied to CL problems in this study. In fact, PSO is not preferred compared to ACO and BA. The reason is that both PSO and some BA are originally proposed for continuous optimization problems (Yang and Simon, 2005; Pham et al., 2006). The use of these two population based optimization techniques for the solution of CL problems is one of the original contributions of this thesis.

3.3. Conclusion

The literature review on CL problems revealed that many different solution approaches have been offered. In addition to exact approaches mostly heuristics and meta-heuristic approaches are proposed due to the complexity of the problem. Meta-heuristic approach is the most common method (Liang et al., 2007).

Among the proposed heuristic approaches, the most popular ones that are widely utilized by the researchers are *"wall building"* and *"stack building"* approaches. In this thesis, a heuristic filling procedure based on the *"wall building"* approach is used. The wall-building and layering approach, first introduced by George and Robinson (1980), is most commonly used and modified by later researchers for its high efficiency and high quality (He and Huang, 2009). It should be noted that, for each meta-heuristic approach, a heuristic filling approach (a kind of decoder algorithm) is needed. Without this decoder algorithm, it is not possible to compute the objective function value of a solution which is volume utilization in the classical CL problem. This approach is explained in detail in Section 4.2.

The literature survey revealed that the most popular meta-heuristic algorithm used so far was GA. In addition to this, SA and TS which are widely used in combinatorial optimization have also been utilized for the CL problem. Interestingly, the algorithms based on SI techniques, such as ACO, PSO and BA algorithms have not been widely applied to these problems. A few studies in the literature are based on these approaches. Therefore, this study is concentrated on the use of some SI techniques to solve CL problems. For this purpose, the use of BA and ACO is considered in this study. In the following chapters, the findings related to the applicability of these methods to CL problems are investigated in detail.

CHAPTER 4

THE HEURISTIC FILLING PROCEDURE

4.1. Introduction

The heuristic filling procedure that is used together with BA and ACO to solve the CL problem and with SA algorithm to solve the multi-objective CL problems is presented in this chapter. First, the nature of the procedure which is based on wall building approach is introduced in the following section. Then, details about determining the dimension of a layer and filling a layer is shared. Finally, the test cases which are used in the literature to test the performance of a CL algorithm are introduced.

4.2. Heuristic Filling Procedure

The proposed heuristic filling procedure is a *"wall-building"* approach that loads the container *layer-by-layer* in a recursive manner. Before filling each layer, its dimensions are determined as it will be explained in detail in Section 4.2.1.

Layers are filled one at a time. If it is not possible to fill a layer with the boxes in the set of available boxes, the current layer is closed and a new one is started. The procedure is repeated until it is not possible to locate a new layer to the remaining container width or when the set of available boxes is empty. As a result of this packing process, the container is filled with the isolated vertical layers where spanning of the boxes between layers is avoided.

4.2.1. Determination of the layer dimensions

Before starting the filling process, it is important to determine the dimensions of a layer since the width of a layer must be carefully selected to obtain a good performance (Pisinger, 2002).

In this study, the width of each layer w_L is set equal to the width of the Layer Determining Box (LDB) (Gehring et al., 1990). In order to determine the LDB, first the boxes among the set of available boxes are sorted by width dimension in non-increasing order. Thus, the box with the greatest width dimension is given the highest priority. In case of a tie among the boxes with the same width dimension, the box with the smallest depth dimension d_i is given a higher priority. Finally, the highest priority box in the set of available boxes is chosen as the LDB.

After the determination of the LDB, the layer having width w_L equal to the width w_i of the LDB, height h_L and depth d_L equal to those of the container is filled. As a result, the dimensions of the layer are determined as seen in Equation (4.1);

$$\begin{aligned} w_L &= \max\ (w_i)\,, \quad \textit{where} \quad i = 1, ..., n \\ h_L &= H \\ d_L &= D \end{aligned} \tag{4.1}$$

4.2.2. Filling the layer

Following the determination of the layer dimensions, it is possible to fill the layers. The layers are filled in a recursive manner. The main advantage of using *recursive algorithms* is that they reduce the solution to a problem with a particular set of input to the solution of the same problem with smaller input values (Rosen, 2003). Besides, the recursive algorithms are simple and easy to implement.

To explain better how the recursion works, a recursive procedure developed for a two-dimensional case is presented below. This procedure which is the primitive of the one that is presented later in this section is as follow;

- Pack the first item into the bottom left corner ((0, 0) coordinate) of the object (This operation also divides the packing *"space S"* into two subsequent subspaces).
- Pack the next item into the *"subspace S1*". If packing to the *"subspace S1"* is not possible, then pack the item to the *"subspace S2"*. Call this procedure recursively until all the items are packed.

In Figure 4.1, the working principle of the recursive filling procedure is illustrated. The first item is packed in the bottom-left corner of the larger object. As a result of this packing, two empty subspaces (S1 and S2) are generated. The algorithm tries to pack the next item into the bottom-left corner of the empty space - subspace S1. If this is not possible, the item is packed to the bottom-left corner of the empty space - subspace S2. This packing will again divide the subspace in which packing is done into two subspaces. The algorithm will try to pack a new item firstly to subspace S11, then to subspaces S12, S21 and to S22. The procedure will be called recursively until all the items are allocated into the large object (Dereli and Daş, 2007).

For the CL problem, this heuristic procedure is adopted to the three dimensional case. The large object in the previously described procedure can be considered as a layer and items can be considered as boxes. Similar to the empty subspaces that are produced every time an item is placed on the large object, empty spaces in a layer are produced every time a box is allocated to the layer having predetermined dimensions.

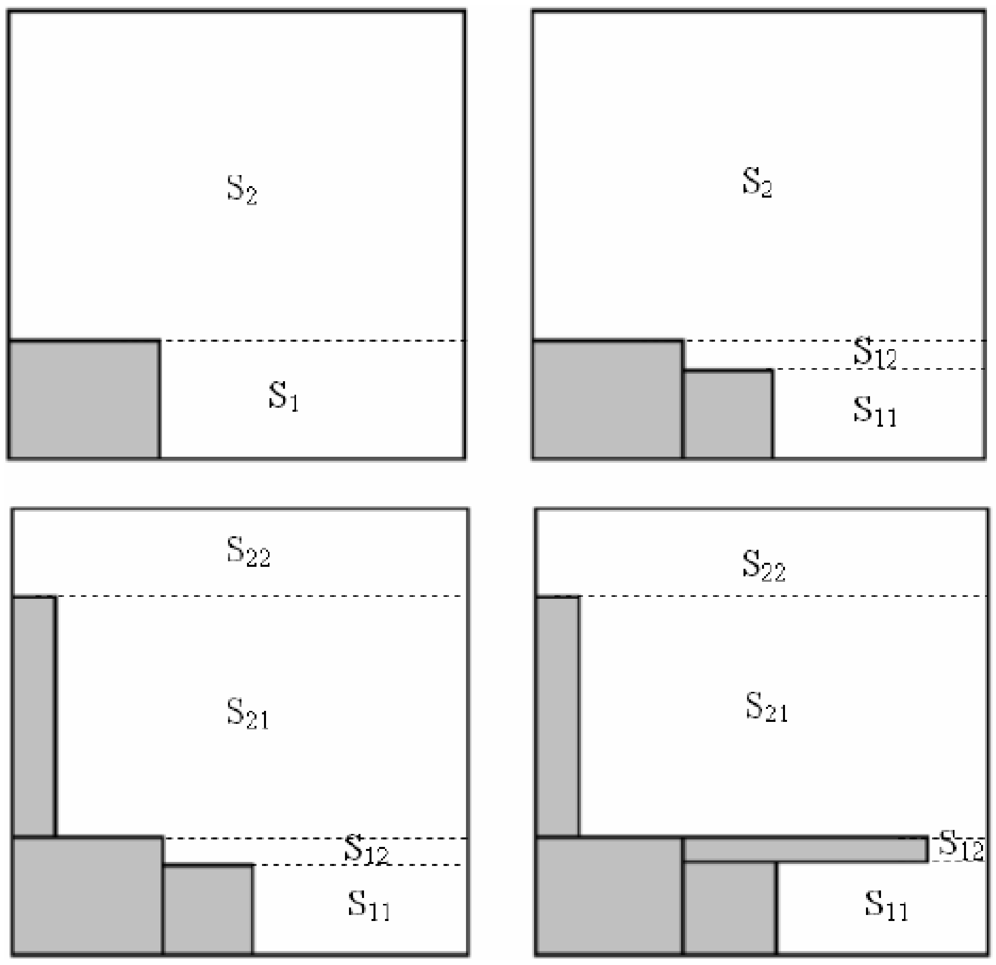


Figure 4.1 The working principle of the recursive algorithm developed for the two dimensional case

When the first box (that is LDB) is allocated into the layer, three empty new-spaces namely "*beside", "in front"* and *"above"* of the packed box are produced. In the given situation (see Figure 4.2 and 4.3), only two of these spaces "*in front"* and *"above"* occurs as a result of the allocation of the first box into the layer. Thus, only these spaces are shown in Figure 4.2**.** However, when a box having a width smaller than the LDB is packed into the layer, an empty space beside the packed box occurs as seen in Figure 4.3. In the proposed filling procedure, the empty spaces are filled in the following order: first, the empty space *"in front"* of the packed box is filled, then the empty spaces *"beside"* and *"above"* of the packed box is filled, respectively.

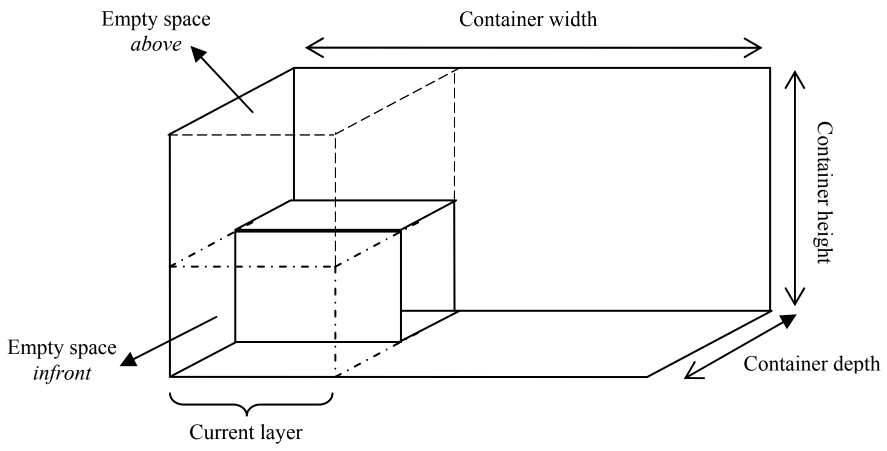


Figure 4.2 Empty spaces “in front” and “above” of the LDB

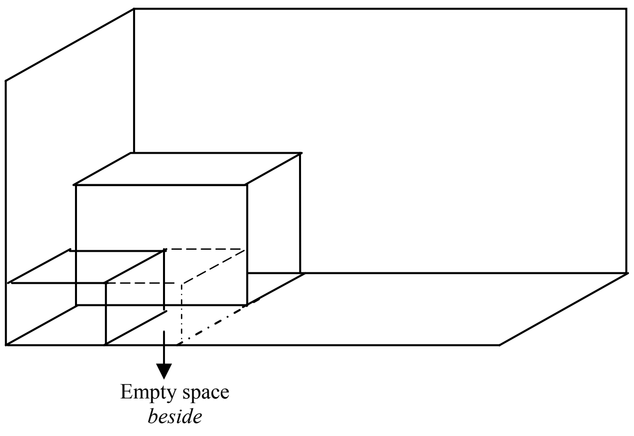


Figure 4.3 Empty space “beside” a packed box in the layer

Now suppose that there is a packed box in the layer as shown in Figure 4.2 and it is desired to pack the next highest priority box in the set of available boxes into the container. First, the space *“in front”* of the packed box is checked. If it is possible to allocate this box into this space, then it is packed there and removed from the set of available boxes since there is not an empty space *“beside”* the packed box in the current layer the space *“above”* the packed box is checked. If it is possible to allocate this box into this space, then the box is packed to this space and the packed box is removed from the set of available boxes. Otherwise, the suitability of the next highest priority box in the set of available boxes to the available empty spaces in the layer is investigated. The process is repeated in a recursive manner for each box in the set of available boxes and for all of the empty spaces available in current layer until it is not possible to fill the empty spaces with a box in the set of available boxes.

After this detailed introduction of the basic features of the heuristic filling procedure, the main steps of the procedure is presented in Figure 4.4.

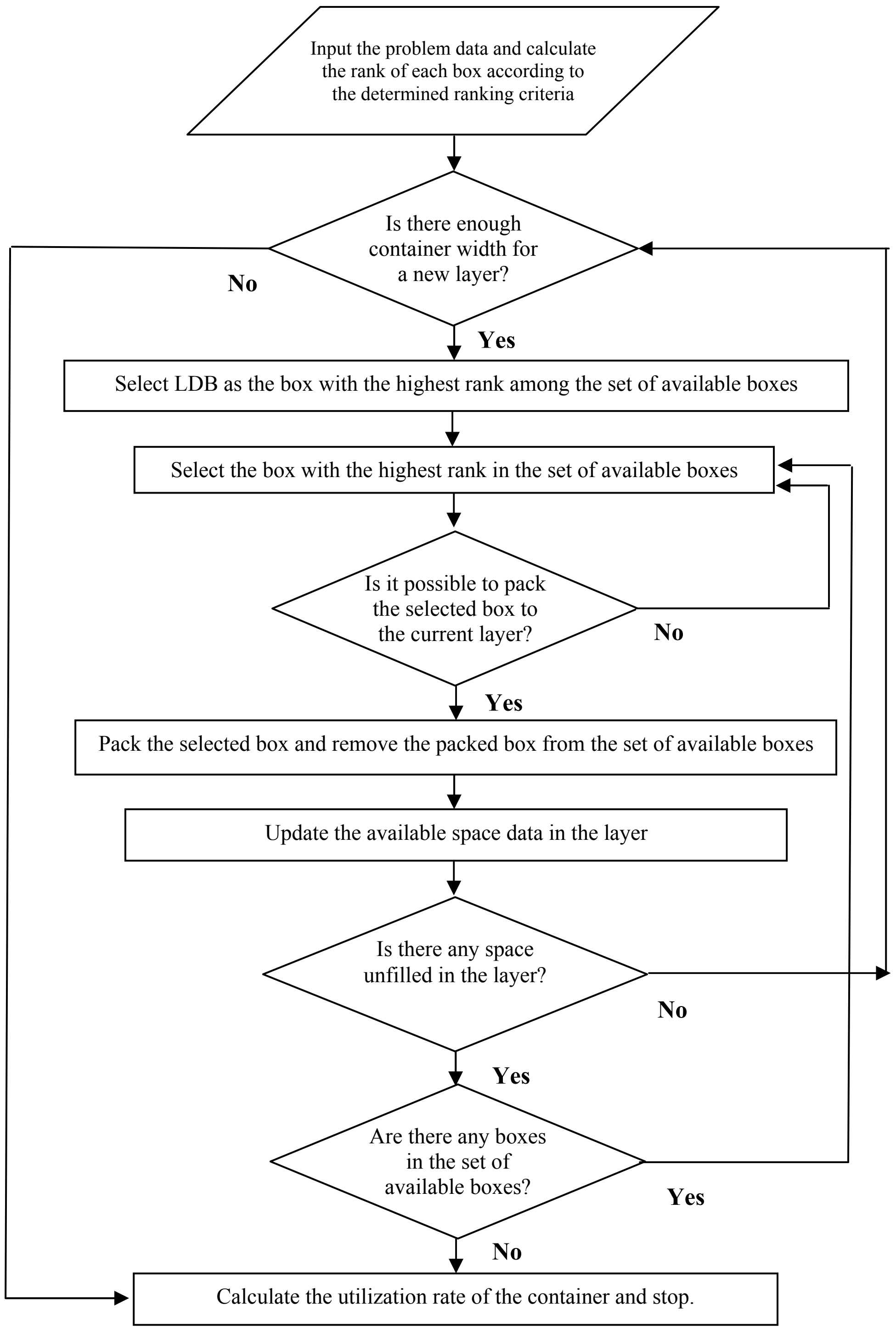

Input the problem data and calculate the rank of each box according to the determined ranking criteria
Is there enough container width for a new layer?
No
Yes
Select LDB as the box with the highest rank among the set of available boxes
Select the box with the highest rank in the set of available boxes
Is it possible to pack the selected box to the current layer?
No
Yes
Pack the selected box and remove the packed box from the set of available boxes
Update the available space data in the layer
Is there any space unfilled in the layer?
No
Yes
Are there any boxes in the set of available boxes?
Yes
No
Calculate the utilization rate of the container and stop.

Figure 4.4. The proposed heuristic filling-procedure

4.3. Test Cases

All throughout the study, the performance of the proposed approaches are tested using the well known test cases from the literature known as Loh and Nee (LN) (1992) test cases and Bischoff and Ratcliff (BR) (1995) test cases. Each class in BR test cases includes 100 problems whereas each LN test case represents a single problem. In both test cases the aim is to allocate a set of boxes with varying dimensions into a container without any overlap to maximize the volume utilization of the container for each problem.

Each test problem is run three times with different seed values and the average of these runs is used to test the performance of the algorithm.

When comparing the performance of the proposed algorithms it should be noted that figures computed by Loh and Nee (named as *packing density*) are not directly comparable to the *volume utilization figures* in the other columns, as they are quoted only on the basis of the smallest rectangular enclosure of the loaded boxes, rather than the actual container dimensions (Bischoff and Ratcliff ,1995).

The proposed algorithms are coded in C++ language and the problems are run on a computer with 2.4 GHz. Intel Pentium IV.

CHAPTER 5

A BEES ALGORITHM FOR SOLVING CONTAINER LOADING PROBLEMS

5.1. Introduction

The main goal of this chapter is to discuss the usability of Bees algorithm (BA) to find a "good" solution to CL problems, which are difficult combinatorial optimization problems. In the next section details related to the implementation of BA to the CL problems is described. Computational results and conclusions are presented in Section 5.3 and 5.4, respectively.

5.2. A Bees Algorithm (BA) for Container Loading

A search algorithm for this part of the study is considered, since it is essential to find out the dimensions of a layer for a good container loading performance as discussed in Section 4.2. As it was mentioned before, *the width of the layers* is set equal to *the width of the boxes,* which has the highest priority in the set of available boxes. If the priority of the boxes in the set of available boxes can be altered, alternative widths for the layers can be considered to reach a good container-loading performance. In our algorithm, the priorities of the boxes (in the set of available boxes) are changed by enabling or disabling the rotation of the boxes.

In this study, motivated by the algorithm proposed by Pham et al. (2006) a BA for CL problems is proposed to reach the above mentioned goal. The algorithm makes an analogy to a colony of honey bees that tries to find promising food sources in the nature. The natural food foraging process of a honey bee colony starts with a number of scout bees from the colony searching the food sources. When the scout bees find a rich food source they begin a so called *"waggle dance"* in the hive (Dyer, 2002) as shown in Figure 5.1.

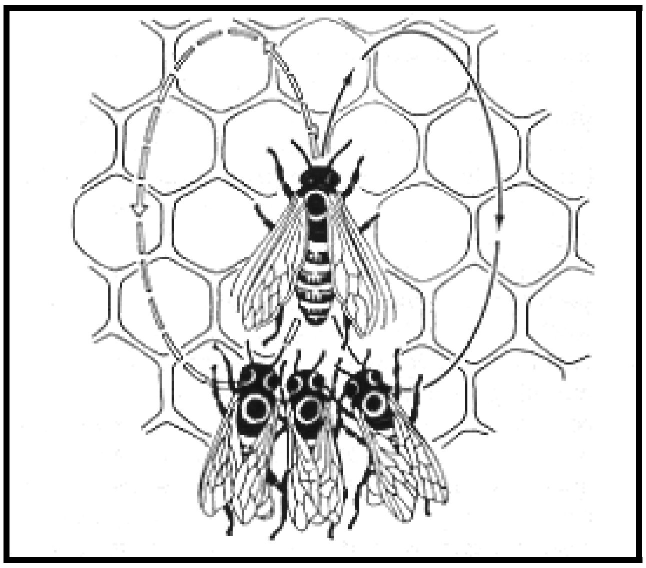
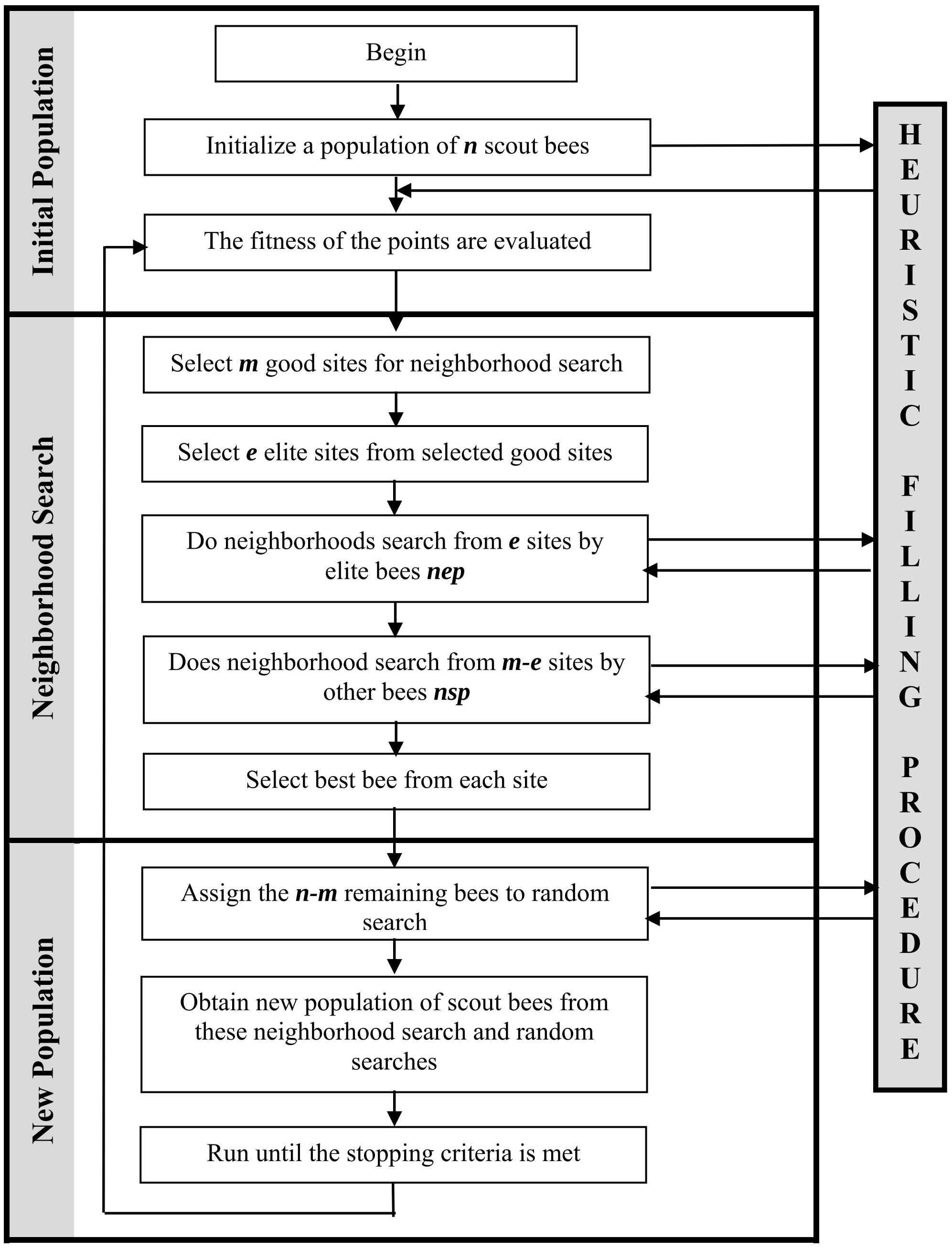


Figure 5.1. Waggle dance of bees (web2)

This dance which is a form of communication between the bees in the colony includes information about the distance, the direction and the quality of the food source. Equipped with this important knowledge the colony sends follower bees - more follower bees are sent to more promising food sources - to these food sources to collect the food. While collecting the food, bees evaluate the food level of the source and collect the needed information for the next waggle dance. The employed BA mimics the food foraging process of the honey bee colony. The main steps of the BA for CL (***hybrid-BA***) problems, which is hybridized with the heuristic filling procedure presented in the previous section, are schematized in Figure 5.2.

In implementing the ***hybrid-BA*** for CL several parameters should be determined. These key parameters are number of scout bees ***n***, number of selected sites ***m,*** number of elite sites ***e*** chosen from ***m*** sites, number of bees recruited to search ***e*** elite sites ***nep***, number of bees recruited to search ***m-e*** other sites ***nsp,*** and the termination criteria. These parameters of the algorithm are adjusted by trial and error since there is not a defined procedure to help the users choose the most appropriate set of parameters.



Initial Population
Neighborhood Search
New Population
Begin
Initialize a population of ***n*** scout bees
The fitness of the points are evaluated
Select ***m*** good sites for neighborhood search
Select ***e*** elite sites from selected good sites
Do neighborhoods search from ***e*** sites by elite bees ***nep***
Does neighborhood search from ***m-e*** sites by other bees ***nsp***
Select best bee from each site
Assign the ***n-m*** remaining bees to random search
Obtain new population of scout bees from these neighborhood search and random searches
Run until the stopping criteria is met
HEURISTIC FILLING PROCEDURE

Figure 5.2. Algorithm of the ***hybrid-BA*** (Daş and Dereli, 2007)

As can be seen in Figure 5.2., the algorithm starts with ***n*** scout bees being placed randomly in the search space. These ***n*** scout bees represent the initial population. Following the acquiring of random initial solutions, solutions found by scout bees are evaluated by the proposed heuristic filling procedure. Bees that have good fitness among this initial population are selected so that ***m*** sites are chosen for neighborhood search. The search is primarily around the best sites among these ***m*** sites known as elite sites ***e*** and other selected sites ***m-e.*** Here, elite sites (***e***) represent the more promising solutions which are searched with ***nep*** bees greater than ***nsp*** bees searching the other ***(m-e)*** sites.

To obtain the next bee population, the bees (evaluated with the heuristics filling procedure) with the highest fitness value are selected from each ***m*** sites. In order to complete the population to ***n*** bees, the remaining ***n-m*** bees are assigned randomly to the search space in order to find new solutions. These steps are repeated until the *stopping criterion* is met or the solution converges.

At this point, it is meaningful to explain why a *"hybridization"* of the algorithm is required. The hybridization of the BA algorithm with the heuristic filling-procedure is essential, since the *objective function values* corresponding to each solution is needed all through the algorithm. Without these values of objective function, the neighborhood search is not started in the algorithm. In Figure 5.2.**,** the interactions between the BA algorithm and the heuristic procedure are also illustrated. As it is clear from Figure 5.2, the heuristic procedure is called upon whenever the objective value of a solution is needed by the BA algorithm.

5.2.1. Representation of a solution and neighborhood search for the BA

Each bee in the population represents a bit string of length equal to the number of box types of a given CL problem. Each bit in this string shows an alternative orientation of a box type (there can be different rotation orientations for different problems). Suppose that, a CL problem having the relevant data like the one presented in Table 5.1 is being dealt with.

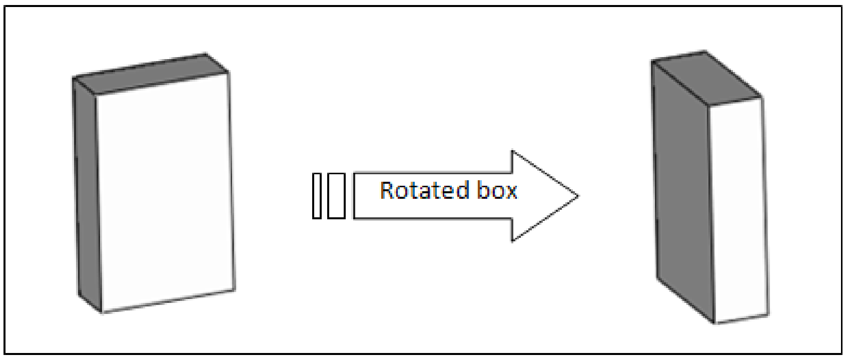
Table 5.1. Data for a CL problem

Box Type	Width	Depth	Height	Total number
Type 1	108	65	55	45
Type 2	95	52	45	55
Type 3	70	62	35	20
Type 4	83	40	20	30
Type 5	90	70	40	18
Type 6	55	48	37	27
Type 7	68	20	10	34
Type 8	100	83	44	41
Type 9	60	32	23	50

The problem contains a total of 320 boxes of 9 different box types (boxes in the same type has the same dimensions) and the bit string representation of this problem is as shown in Figure 5.3**.** Suppose that these boxes can only be base rotated. That is there is a ***"this way up"*** constraint for these boxes. Each bit in this string represents a box type and the numbers "0" and "1" represents whether the boxes of a type are rotated or not. The rotation of an ordinary box is shown in Figure 5.4.

0	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**1**
Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9

Figure 5.3. Bit string representation of a solution (only for a base-rotated box)



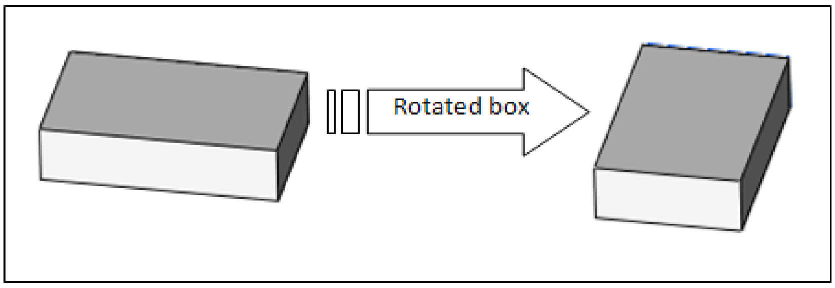


Figure 5.4. The rotation of a box (only for a base-rotated box)

The solution shown in Figure 5.3 tells the program to rotate (to exchange width and depth dimensions in case only rotation on the base is allowed) all the boxes of the ***type 2, 3, 5, 7, 8, 9*** and then allocate these boxes into the container in this new rotation-orientation. This structure is preferred to the structure in which each bit in the string represents a box in the problem. In this case, the resulting bit string will be of length 320 which will be a very inconvenient and time-consuming structure for the large sized problems.

Three operators, namely; "**1-flip**" and **"k-flip"** are defined in this work in order to reach the neighborhood solutions. In case of "**1-flip"** type of operator, the value of a randomly selected bit is flipped from "1 to 0" or "0 to 1. The operation of this operator is illustrated in Figure 5.5.

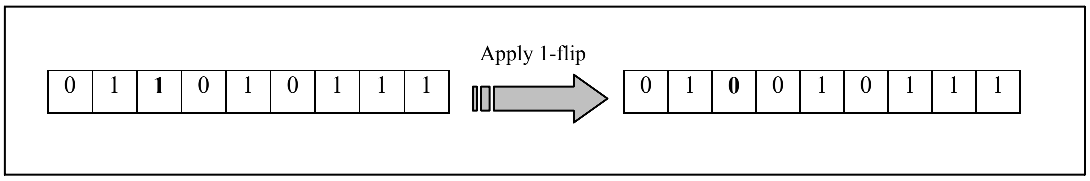


Figure 5.5. The use of "1-flip" operator

The second operator is designed with the motivation from the work of Kong et al. (2007). They designed a simple random **4-flip** method as the local search. This operator randomly selects four variables from the solution and flips their values from "1 to 0" or "0 to 1". If the newly generated solution is better, they replaced the original solution with the new one. They applied this method 1000 times for each solution. They selected the number of flips and the number of execution through a set of experiments. Similar to this structure, an operator named "**k-flip"** is designed in this work. This operator rotates a corresponding number of boxes (**k** *times* "**total box number"**) that are selected randomly. It is possible to rotate boxes of a *specific box-type* as well as the boxes from *alternative box-types* by the use of operators one after another. For example, for the sample problem provided in Table 5.1**,** the neighborhood search process through the use of operators discussed above is illustrated and explained in Figure 5.6. For the example provided above, firstly 1-flip operator is applied and the 1-flip operator is applied to the randomly selected bit. Accordingly, randomly selected bit representing all the boxes of Type 3 is rotated. Following the 1-flip operator, also k-flip operator is applied to the example bit string. The k-flip is applied to a number of selected bit position which is equal to the k times the total box number in the position (where k < 1). If the k is selected 0.5, then a total of 320 x 0.5 individual boxes that are selected randomly should be rotated. For example, randomly selected boxes; box number 15 having Type 1, box number 123 having Type 3,…etc.

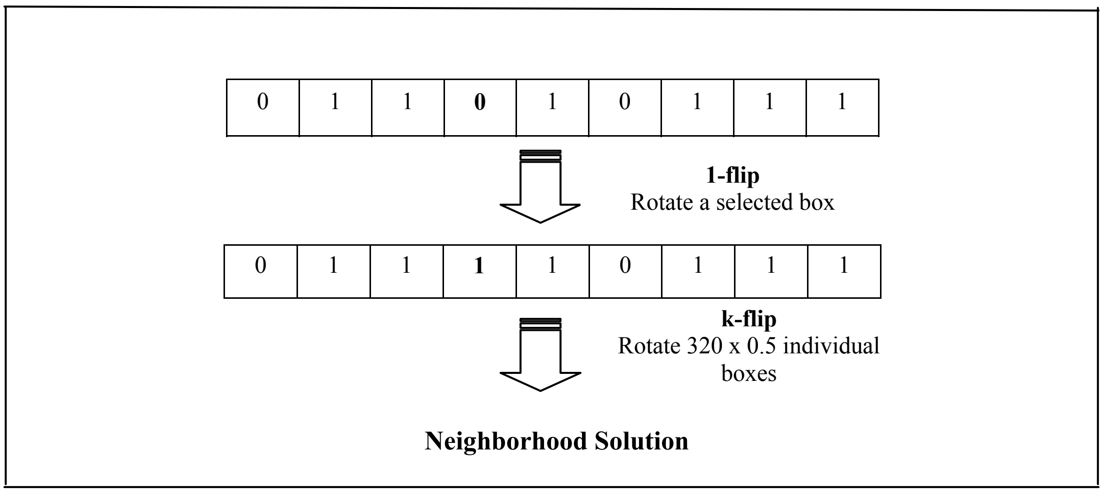


Figure 5.6 The operation of the defined operators

If the obtained neighborhood solution is better than the current solution, then the neighborhood solution is saved. Otherwise, the current solution is saved. Using both operators, it is possible to evaluate a large number of solutions.

5.3. Computational Results for the ***hybrid-BA***

The results of the LN and BR test cases are presented as a single column in Table 5.3 and Table 5.5 where *volume-utilization ratios* obtained by each study in the literature are also presented in Table 5.2, 5.3, 5.4 and 5.5.

The parameters of the ***hybrid-BA*** algorithm for both test cases are set to the following; number of bees (population) ***n*** = 20; number of selected sites ***m*** = 4; number of elite sites ***e*** = 2; number of bees send to elite points ***nep*** = 4 and number of bees send to other selected points ***nsp*** = 2 for 1000 iterations.

The **k-** coefficient is defined as a ratio of *the total number of boxes* in the problem (thus defined between 0 and 1) and the value of this coefficient is determined through a set of experiments. For both test cases this coefficient is set to 0.6.

Table 5.2. Comparative results with test cases of LN– heuristic approaches

Problem	Loh and Nee (1992)*	Ngoi et al. (1994)	Bischoff et al. (1995)	Bischoff and Ratcliff (1995)	Eley (2002)	Bischoff (2003)	Lim et al. (2005)
LN01	78.1	62.5	62.5	62.5	62.5	NC*	62.5
LN02	76.8	80.7	89.7	90.0	90.8	NC	80.4
LN03	69.5	53.4	53.4	53.4	53.4	NC	53.4
LN04	59.2	55.0	55.0	55.0	55.0	NC	55.0
LN05	85.8	77.2	77.2	77.2	77.2	NC	76.7
LN06	88.6	88.7	89.5	83.1	87.9	NC	84.8
LN07	78.2	81.8	83.9	78.7	84.7	NC	77.0
LN08	67.6	59.4	59.4	59.4	59.4	NC	59.4
LN09	84.2	61.9	61.9	61.9	61.9	NC	61.9
LN10	70.1	67.3	67.3	67.3	67.3	NC	67.3
LN11	63.8	62.2	62.2	62.2	62.2	NC	62.2
LN12	79.3	78.5	76.5	78.5	78.5	NC	69.5
LN13	77.0	84.1	82.3	78.1	85.6	NC	73.3
LN14	69.1	62.8	62.8	62.8	62.8	NC	62.8
LN15	65.6	59.5	59.5	59.5	59.5	NC	59.5
Mean	**74.2**	**69.0**	**69.5**	**68.6**	**69.9**	**-**	**67.0**

Table 5.3. Comparative results with test cases of LN– metaheuristic approaches

Problem	Gehring et al. (1997)	Bortfeldt et al. (1998)	Bortfeldt et al. (2001)	Gehring and Bortfeldt (2002)	Bortfeldt, et al. (2002)	Moura and Oliveira (2005)	***hybrid-BA***
LN01	62.5	62.5	62.5	NC*	NA**	NA	62.5
LN02	90.7	96.7	89.8	NC	NA	NA	86.3
LN03	53.4	53.4	53.4	NC	NA	NA	53.4
LN04	55.0	55.0	55.0	NC	NA	NA	55.0
LN05	77.2	77.2	77.2	NC	NA	NA	77.2
LN06	91.1	96.3	92.4	NC	NA	NA	89.2
LN07	82.7	84.7	84.7	NC	NA	NA	83.2
LN08	59.4	59.4	59.4	NC	NA	NA	59.4
LN09	61.9	61.9	61.9	NC	NA	NA	61.9
LN10	67.3	67.3	67.3	NC	NA	NA	67.3
LN11	62.2	62.2	62.2	NC	NA	NA	62.2
LN12	78.5	78.5	78.5	NC	NA	NA	78.5
LN13	85.6	85.6	85.6	NC	NA	NA	83.6
LN14	62.8	62.8	62.8	NC	NA	NA	62.8
LN15	59.5	59.5	59.5	NC	NA	NA	59.5
Mean	**70.0**	**70.9**	**70.1**	**-**	**70.9**	**70.3**	**69.46**

*These values are not computed in this study, ** These values are not available separately.

The results indicate that the proposed algorithm - ***hybrid-BA*** - is capable of solving CL problems. The algorithm reaches 11 out of 15 best-known solutions for LH test cases. On the other hand, for harder problems LN02, LN06, LN07 and LN013 the algorithm found feasible solutions. The ***hybrid-BA*** algorithm is the second best performing algorithm after the heuristic approach of Eley (2002) which is the best performing heuristic algorithm among the compared heuristic approaches and a quiet well performing algorithm among the compared meta-heuristic approaches. When the performance of the ***hybrid-BA*** is compared to other meta-heuristic approaches available in the literature, there is a performance gap of 2.03% between the best performing meta-heuristic algorithm and the proposed algorithm.

Table 5.4. Comparative results with the test cases of BR– heuristic approaches

Problem (box type)	Bischoff et al. (1995)	Bischoff and Ratcliff (1995)	Eley (2002)	Bischoff (2003)	Lim et al. (2005)
BR1(3)	81.76	83.79	NA*	89.39	87.40
BR2(5)	81.70	84.44	NA*	90.26	88.70
BR3(8)	82.98	83.94	NA*	91.08	89.30
BR4(10)	82.60	83.71	NA*	90.90	89.70
BR5(12)	82.76	83.80	NA*	91.05	89.70
BR6(15)	81.50	82.44	NA*	90.70	89.70
BR7(20)	80.51	82.01	NA*	90.44	89.40
Mean	**81.97**	**83.50**	**88.75**	**90.55**	**89.13**

*These values are not available separately.

Table 5.5. Comparative results with the test cases of BR – metaheuristic approaches

Problem (box type)	Gehring et al. (1997)	Bortfeldt et al. (1998)	Bortfeldt et al. (2001)	Gehring, Bortfeldt, (2002)	Bortfeldt, et al. (2003)	Mack et al. (2004)	Moura, Oliveira (2005)	***hybrid-BA***
BR1(3)	85.80	92.63	87.81	88.10	93.52	93.70	89.07	83.41
BR2(5)	87.26	92.70	89.40	89.56	93.77	94.30	90.43	84.60
BR3(8)	88.10	92.31	90.48	90.77	93.58	94.54	90.86	85.42
BR4(10)	88.04	91.62	90.63	91.03	93.05	94.27	90.42	85.19
BR5(12)	87.86	90.86	90.73	91.23	92.34	93.83	89.57	85.11
BR6(15)	87.85	90.04	90.72	91.28	91.72	93.34	89.71	84.69
BR7(20)	87.68	88.63	90.65	91.04	90.55	92.5	88.05	83.99
Mean	**87.50**	**91.26**	**90.10**	**90.43**	**92.70**	**93.78**	**89.73**	**84.63**

The performance of the ***hybrid-BA*** for the test cases of BR is also comparable with the other heuristic and meta-heuristic approaches. There is a performance gap of 9.7% between the best performing meta-heuristic algorithm and the proposed algorithm and a performance gap of 6.54% between the best performing heuristic algorithm and the proposed algorithm.

Each problem set in BR test cases contains different number of box types. The results summarized in Table 5.5 reveals that the performance of the algorithm first shows sharp increase as the number of box type increase, but then shows a gradual decrease. This can also be seen in Figure 5.7. A similar behavior can also be observed in some of the algorithms in the literature (refer to Table 5.5).

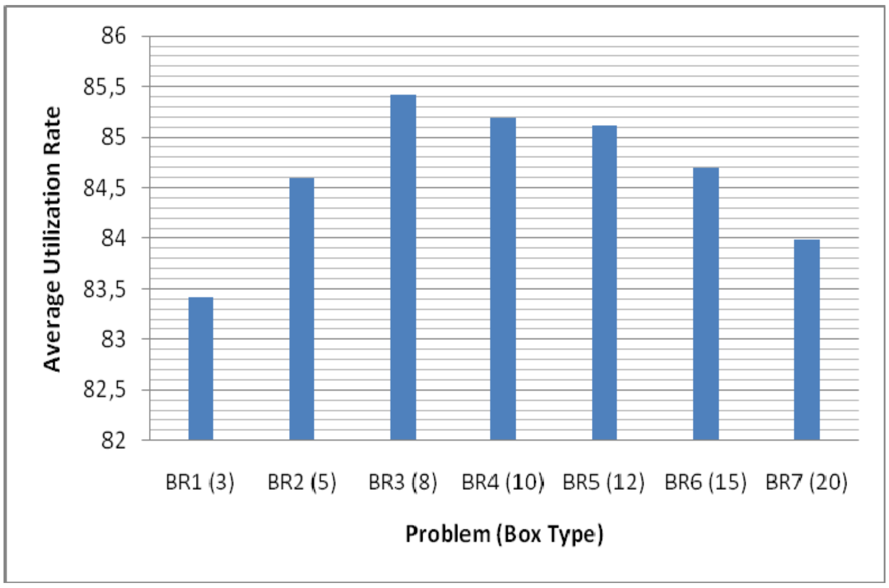


Figure 5.7. Graphical representation of the obtained results for BR test cases

The convergence graph of the ***hybrid-BA*** algorithm for a problem from BR7 is presented in Figure 5.8. It is clear from the convergence graph in Figure 5.8 that the algorithm converges after a reasonable number of iterations.

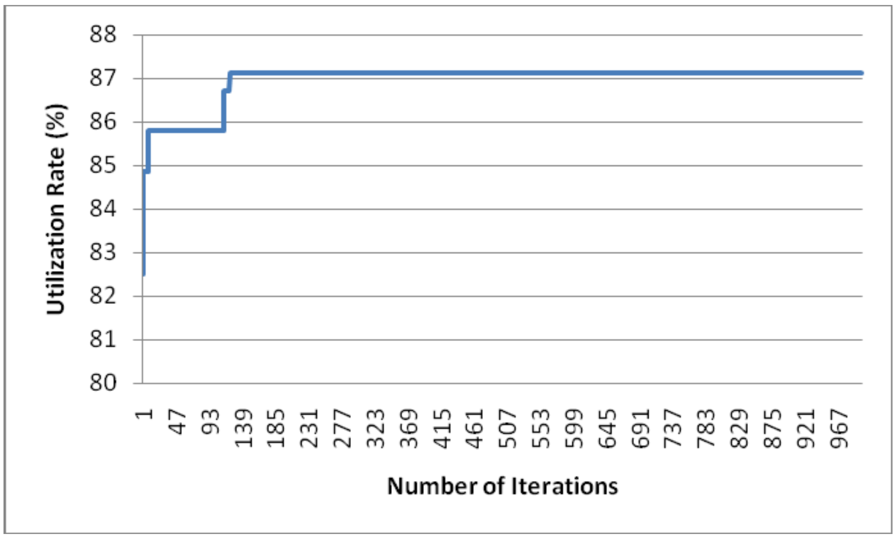


Figure 5.8. Convergence graph of a problem from test case BR7

5.3.1. Determination of the parameters of the hybrid-BA algorithm

The results for the proposed algorithm are already discussed in the previous section. Nevertheless, an attempt to determine the parameters of this algorithm and to discuss the contribution of this newly determined set of parameters to the performance of the algorithm is made in this section. For this purpose, factorial design is preferred. Factorial designs are efficient tools for problems where the study of two or more factors is needed (Montgomery, 1991).

To study the effects of the BA's parameters on the ***hybrid-BA's*** solution quality, a factorial analysis with four control parameters; number of scout bees ***n,*** number of selected sites ***m,*** number of bees send to elite points ***nep*** and number of iterations ***iter*** is designed. This design with four control parameters of the ***hybrid-BA*** algorithm is given in Table 5.6 and 5.7. In order to test the effects of these control parameters (factors), 10 problems from the BR7 test cases with 2*2*2*2 (= 16) different setting is run 3 times with different seed and totally 48 experiments are conducted. Minitab statistical software is used for the analysis.

Table 5.6. Levels of factors for the factorial design

Factors	**Levels**	
number of scout bees *n(nob)*	10	20
number of selected sites *m*	2	4
numbers of bees send to elite points *nep*	2	4
number of iterations *iter*	500	1000

The results of the analysis obtained from the Minitab have revealed that all the factors (*n, m, nep* and *iter*) are significantly affecting the performance of the proposed algorithm (relevant p-values for these parameters are smaller than 0.05). As the number of scout bees, numbers of bees send to elite points and number of iterations increases, the solution quality of the algorithm improves. On the other hand, as the number of selected sites decreases, the solution quality of the algorithm increases. There are no significant two-way, three-way and four-way interactions between the parameters. Main Effects plots and Interaction plots of this analysis are also shown in Figure 5.9 and 5.10.

Table 5.7. Factorial design on ***hybrid-BA*** response (obtained from Minitab)

```
Estimated Effects and Coefficients for C5 (coded units)

Term               Effect       Coef   SE Coef        T      P
Constant                    0,835130  0,000289  2886,75  0,000
n                0,003170   0,001585  0,000289     5,48  0,000
m               -0,001636  -0,000818  0,000289    -2,83  0,008
nep              0,001546   0,000773  0,000289     2,67  0,012
iter             0,003063   0,001531  0,000289     5,29  0,000
n*m             -0,000245  -0,000122  0,000289    -0,42  0,675
n*nep            0,000515   0,000258  0,000289     0,89  0,380
n*iter          -0,001090  -0,000545  0,000289    -1,88  0,069
m*nep           -0,000337  -0,000169  0,000289    -0,58  0,564
m*iter           0,000633   0,000316  0,000289     1,09  0,282
nep*iter         0,000154   0,000077  0,000289     0,27  0,791
n*m*nep         -0,000772  -0,000386  0,000289    -1,33  0,192
n*m*iter         0,000262   0,000131  0,000289     0,45  0,654
n*nep*iter       0,000754   0,000377  0,000289     1,30  0,202
m*nep*iter      -0,000113  -0,000056  0,000289    -0,20  0,847
n*m*nep*iter    -0,000631  -0,000316  0,000289    -1,09  0,283


S = 0,00200431   PRESS = 0,000289244
R-Sq = 72,47%    R-Sq(pred) = 38,05%   R-Sq(adj) = 59,56%


Analysis of Variance for C5 (coded units)

Source              DF      Seq SS      Adj SS      Adj MS      F      P
Main Effects         4  0,00029399  0,00029399  0,00007350  18,30  0,000
2-Way Interactions   6  0,00002462  0,00002462  0,00000410   1,02  0,429
3-Way Interactions   4  0,00001494  0,00001494  0,00000373   0,93  0,459
4-Way Interactions   1  0,00000478  0,00000478  0,00000478   1,19  0,283
Residual Error      32  0,00012855  0,00012855  0,00000402
  Pure Error        32  0,00012855  0,00012855  0,00000402
Total               47  0,00046688
```

As it can be seen from the main effects plot (Figure 5.9), the best results are obtained for the parameters; *n* equals to 20, *m* equals to 2, *nep* equals to 4 and *iter* equals to 1000. Furthermore, the value of the parameters ***e*** (number of elite sites ***e*** chosen from ***m*** sites) and ***nsp*** (number of bees recruited to search ***m-e*** other sites) is chosen equal to 50% of the value of the parameters ***m*** and ***nep***, respectively.

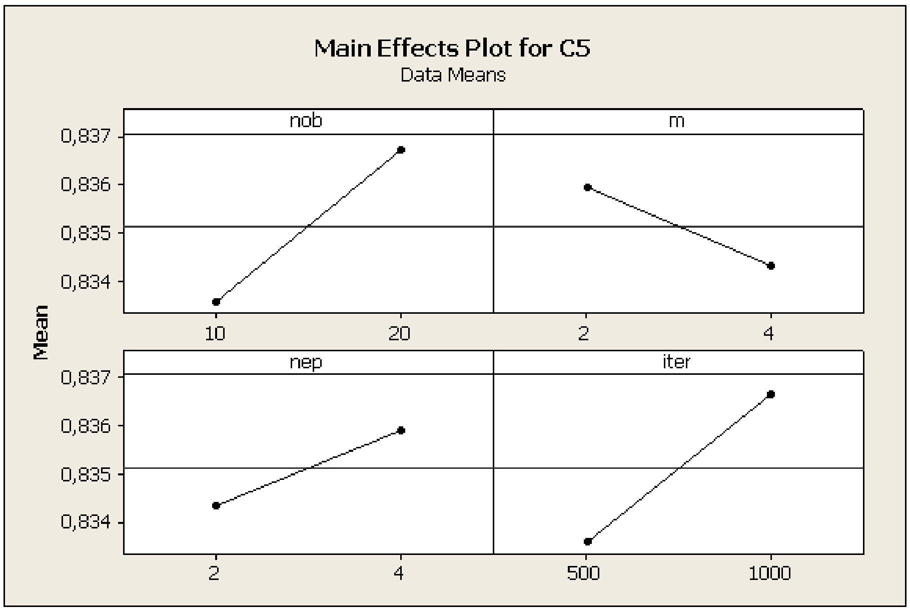


Figure 5.9. Main effects plots

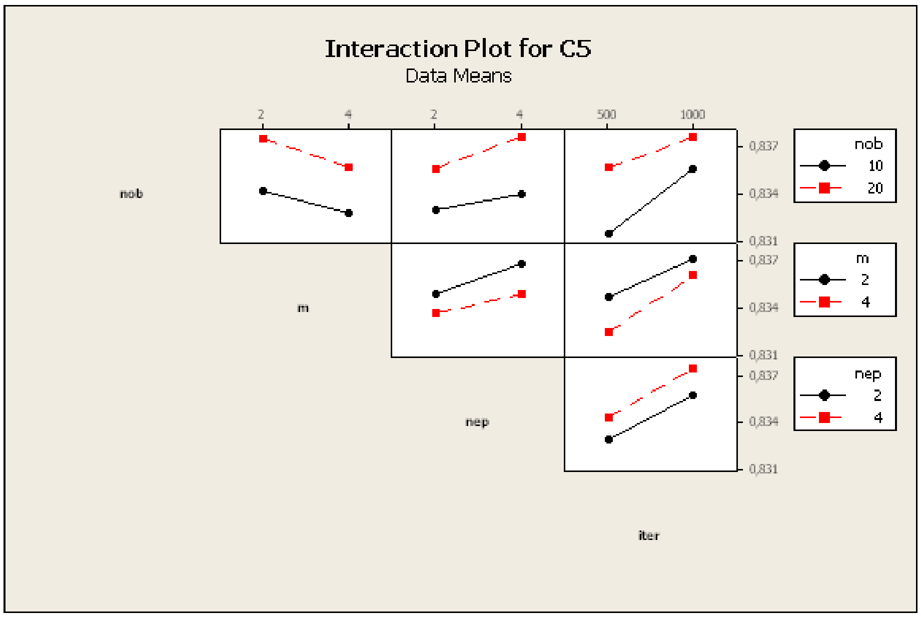


Figure 5.10. Interaction plots

In the proposed algorithm, random numbers are used in different part of the algorithm especially for the neighborhood search. Results presented previously were obtained by three runs. In order to study both the effect of randomness and the effect of the new set of parameters, the algorithm is run 10 times using the new set of parameters for the first 10 problems of the BR test cases. The computational results obtained from this analysis and relevant run times (shown in second column) are summarized in Table 5.8. The third column of Table 5.8 shows the results obtained over three runs, whereas column number 4 to 8 shows the results obtained over 10 runs.

Table 5.8. Analysis of the randomness in ***hybrid-BA*** over 10 runs

Problem (box type)	***Avg. Elapsed Time (sec) (100 problems)***	***hybrid-BA (3 runs)***	***hybrid-BA (10 runs) (10 problems)***				
			Min	***Max***	***Mean***	***SD***	***Range***
BR1 (3)	276,77	83.41	83.13	83.63	83,37	0.16	0,5
BR2 (5)	203,19	84.60	85.10	85.77	85,47	0.22	0,67
BR3 (8)	201,19	85.42	85.84	86.06	85,96	0.07	0,22
BR4 (10)	195,82	85.19	85.67	86.20	85,39	0.15	0,53
BR5 (12)	193,10	85.11	84.92	85.29	85,06	0.13	0,37
BR6 (15)	188.57	84.69	84.64	85.14	84.80	0.15	0,5
BR7 (20)	184.42	83.99	83.75	84.38	84.04	0.22	0,63
Mean	**214,014**	**84.63**	**84,72**	**85,21**	**84,87**	**0,16**	**0,48**

The results of ten runs are higher than the results of three runs. This can be due to the variability available in the algorithm and/or to the new set of parameters. The values of the average standard deviation and average range also support this finding. Therefore, running the algorithm several times in spite of longer computational time, can lead to an increase in volume.

5.4. Conclusion

In this chapter, a relatively new algorithm namely 'bee(s) algorithm' is offered as an alternative solution approach for the CL problem. Hybridized with a heuristic filling procedure based on *"wall building"* approach, the proposed algorithm (so called ***hybrid-BA***) is proved to be successful in solving these problems. The performance of the proposed algorithm against the performances of the other heuristics and meta-heuristics approaches (proposed for the same problems) is compared based on the volume utilization. The ***hybrid-BA*** produced comparable results with those of the other approaches.

CHAPTER 6

AN ANT COLONY ALGORITHM FOR SOLVING CONTAINER LOADING PROBLEMS

6.1. Introduction

As it is obvious from the literature review presented in Chapter 3, up to the present study efforts on applying ant colony algorithms to CL problems are very limited. With this motivation, a new approach to CL problems by using Ant Colony Optimization is proposed.

6.2. Ant Colony Optimization

6.2.1. Behaviors of real ants

Ants are social insects, that is, insects that live in colonies and whose behavior is directed more to the survival of the colony as a whole than to that of a single individual component of the colony (Dorigo et al., 1999). Real ants are capable of finding the shortest path from a food source to their nest without using visual cues by exploiting pheromone information (Dorigo and Gambardella, 1997). Ants communicate among themselves through this pheromone. ***Pheromone***, is a chemical substance that they lay on the ground along the path they traverse. This way they form pheromone trails on the ground. If no pheromone on the ground is available, then ants move randomly. Otherwise, ants observe the pheromone trail and are attracted to it while the path is marked again and will attract even more ants to follow the trail (Zhao et al., 2006).

This mechanism can be explained as follows; Figure 6.1(a) illustrates the shortest way between the nest and food source.

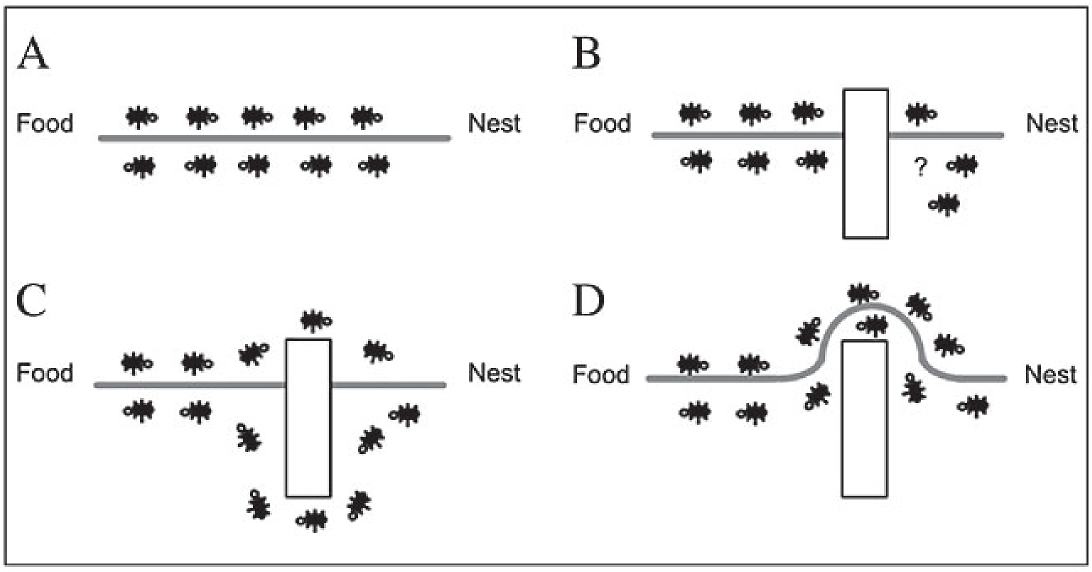


Figure 6.1. (a) Ants in a pheromone trail between nest and food **(b)** an obstacle interrupts the trail **(c)** ants find two paths to go around the obstacle **(d)** a new pheromone trail is formed along the shorter path (Peretto and Lopes, 2005)

As it is mentioned before, the ants have the ability to find the shortest way in case of any changes in the path that they follow. In case of putting an obstacle to their path between the nest and the food source as shown in Figure 6.1(b), they randomly choose the upper path or the lower path as can be seen in Figure 6.1(c). If it is assumed that the ants move at approximately the same speed, the ants which choose the upper path (shorter path) return the nest faster. As the time passes, the amount of the pheromone on the shortest path increases much more quickly than the longer path. Thus, more ants choose the shortest way which has a greater amount of pheromone on it and follow the shorter path as illustrated in Figure 6.1(d).

6.2.2. Simple ACO and Ant Colony System

Algorithms based on the foraging behavior of ants have been first introduced by Dorigo and were formalized as a new meta-heuristic termed Ant Colony Optimization (ACO) in 1999 (Zhao et al., 2006). ACO is a technique for hard combinatorial optimization problems. Early implementations of the algorithm focused on the traveling salesman and other routing problems but nowadays, it is being applied to an increasingly diverse range of combinatorial optimization problems including shortest common super sequence, generalized assignment, multiple knapsack, constraint satisfaction problems, among others (Cordon et al, 2002).

Up to date, several ACO algorithms have been proposed in the literature. The original Ant System and some successful variants are the Ant Colony System, MAX-MIN Ant System, Rank- based Ant system and Best-Worst Ant System (Cordon et al, 2002). The most successful of these are Ant System (AS), Ant Colony System (ACS) and MAX-MIN Ant System (web3). In this study, Ant Colony System which is proposed by Dorigo and Gambardella (1997) is made to become the focused algorithm.

Prior to the introduction of ACS, the simple ACO algorithm is presented below;

Consider the problem of finding the shortest path between two nodes on a graph, $G = (V, E)$, where V is the set of vertices (nodes) and E is a matrix representing connections between nodes (Engelbrecht, 2005). Here,

$n_G =$ is the number of nodes in the graph

$L_k =$ is the path constructed by ant k

$\tau_{ij} =$ is the total pheromone concentration in the edge (i,j)

$\tau_{ij}(0) =$ is the initial pheromone in the edge (i,j)

At the beginning, a number of ants, $k = 1,...,n_k$, are randomly placed on the source node. In each iteration, each ant incrementally constructs a path to the destination node (Engelbrecht, 2005). At each node, each ant tries to determine which node to visit next. If ant k is currently at node i, it selects the next node $j \in N_i^k$ using a transition probability;

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}(t)} & if \ j \in N_i^k \\ 0 & if \ j \notin N_i^k \end{cases} \tag{6.1}$$

where N_i^k is the set of feasible nodes connected to the node i, that ant k can visit. In Equation (6.1), α is a positive integer which magnifies the influence of pheromone concentrations.

When all ants complete their tour from the source to destination, each ant retraces its path to the source node deterministically and deposits a pheromone amount of;

$$\Delta\tau_{ij}^{k}(t) \ \ \alpha \ \ 1/L^{k}(t) \tag{6.2}$$

to each link *(i,j),* of the corresponding path; $L^k(t)$ is the length of the path constructed by ant *k* at time step *t* (Engelbrecht, 2005).

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau^{k}{}_{ij}(t) \tag{6.3}$$

According to Equation (6.3), the total pheromone intensity of a link is proportional to the desirability of the path in which the link occurs, based on the length of the path.

At each iteration of the algorithm, pheromone intensities on the links are evaporated to force ants to explore more and to prevent premature convergence. For each link (Engelbrecht, 2005),

$$\tau_{ij}(t) \quad \leftarrow \ (1-\rho).\tau_{ij}(t) \ \ where \ \ p \in [0,1] \tag{6.4}$$

The constant p here determines the rate of evaporation of the pheromone, which also cause ants to forget previous decisions.

ACS differs from AS and the simple ACO in four aspects: (1) a different transition rule is used, (2) different pheromone update rule is defined, (3) local pheromone updates are introduced and (4) candidate lists are used to favor specific nodes (Engelbrecht, 2005).

ACS uses *pseudo-random-proportional* transition rule given in Equation (6.5). This rule balances exploration and exploitation abilities of the algorithm. According to this rule an ant k located in node i selects the next node j as (Engelbrecht, 2005);

$$j = \begin{cases} \arg\max_{u \in N_i^k(t)} \{\tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)\} & \text{if } \mathrm{r} \le \mathrm{r}_0 \\ J & \text{if } \mathrm{r} > \mathrm{r}_0 \end{cases} \tag{6.5}$$

In Equation (6.5), τ_{iu} is the amount of pheromone from node *i* to node *u*, η_{iu}^{β} is the heuristic information from node *i* to node *u*, β is a parameter between 1 and 10 that determines the relative importance of the heuristic information, *r* is a random number between 0 and 1 and r_0 is a user defined parameter. According to this equation, if the random number *r* is grater that *r0* ($r > r_0$), then the best edge is chosen according to the Equation (6.6) (Engelbrecht, 2005);

$$p_{iJ}^k(t) = \frac{\tau_{iJ}^{\alpha}(t)\eta_{iJ}^{\beta}(t)}{\sum_{u \in N_i^k} \tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)} \tag{6.6}$$

In addition, it should be noted that ACS transition rule uses $\alpha = 1$ and therefore can be omitted from the equation.

In ACS, only the globally best ant is allowed to deposit pheromone. The globally best ant refers to the ant that completes the tour with the minimum distance. The aim of this updating rule is to make the search more directed. The global updating is applied after all ants complete their tours (that is why it is called global updating). The pheromone for the global updating is updated according to the Equation (6.7) and (6.8) (Engelbrecht, 2005);

$$\tau_{ij}(t+1) = (1-p_1)\tau_{ij} + p_1 \Delta\tau_{ij}(t) \tag{6.7}$$

$$\Delta\tau_{ij}(t) = \begin{cases} 1/f(x^+(t)) & \mathit{if}\ (i,j) \in x^+(t) \\ 0 & \mathit{otherwise} \end{cases} \tag{6.8}$$

where p_1 is pheromone decay parameter and valued between 0 and 1, and in Equation (6,8), $x^+(t)$ is the shortest path and $\Delta\tau_{ij}$ is the inverse of the tour length of the global best ant.

The other pheromone updating rule is local pheromone updating. The difference of this rule from the global one is the time of the application. This local pheromone update is applied just after a node is visited according to Equation (6.9) (Engelbrecht, 2005);

$$\tau_{ij}(t) = (1-p_2)\tau_{ij} + p_2\ \tau_0 \tag{6.9}$$

where p_2 is the local pheromone decay parameter and between 0 and 1, and τ_0 is a small positive constant. The essential aim of the local updating is to wide the neighborhood of the previous tours by changing the desirability of the edges visited previously.

6.3. The hybrid-ACS-1 and hybrid-ACS-2 algorithms for solving container loading problems

CL is an optimization problem which is not defined on a graph. Due to this, first the problem is restated as a graph/ network search problem. It is assumed that a CL problem containing *n* boxes is similar to a graph having *n* nodes. Each ant has an empty container and at each node there is a virtual box. As an ant travels the graph, it collects the boxes from the nodes that it visits. That is, when the tour of an ant is completed, it tries to pack the boxes into the container.

After ants build solutions, the fitness of each solution that is built by each ant is calculated with the use of the heuristic filling procedure and only the best ant is allowed to make the global updating. Local updating is done by each ant every time an ant selects a node to visit. The algorithm is run until the predetermined number of iterations is reached.

In the ant algorithms pheromone trail should be determined according to the typeof the problem. In the proposed algorithms the pheromone trail is defined in the similar way to that of Levine and Ducatelle (2004). In their study for the one-dimensional BPP, they defined the $\tau(i, j)$ as the favorability of packing items *i* and *j* in the same bin (only one dimension, that is weight is considered). In the proposed algorithms $\tau(i, j)$ shows the favorability of choosing item *j* after item *i* to allocate into the container.

Another important choice is the determination of a heuristic that is vital for the pheromone calculations. The heuristic information $n\ (j) = w_j$ is set equal to an items width for the proposed algorithm.

The algorithm that is working in the above mentioned manner is called the ***hybrid-ACS-1*** algorithm. This algorithm uses original problem data about the problem all through the process. However, if there are too many boxes in a problem, this solution approach could be very inefficient. In such a situation, reducing the number of boxes that is reducing the number of nodes to be visited by each ant could improve the solution. The steps of the ***hybrid-ACS-1*** are illustrated in Table 6.1.

Table 6.1. The steps of the ***hybrid-ACS-1*** algorithm

```
Initialize parameters

for each ant do
        construct the full path
        apply local updating
        compute the fitness of each path
end

for each link do
        apply global updating
end
```

In order to test the hypothesis whether reducing the number of nodes with this method will produce better results or not, the ***hybrid-ACS-2*** algorithm is proposed as an improved version of the ***hybrid-ACS-1***.

As it is previously discussed in Chapter 4, the heuristic filling procedure proposed in this study fills a container in a *"layer-by-layer"* manner. That is the volume utilization of a container greatly depends on the volume utilization of each layer. Therefore, one way of improving the volume utilization of a container could be to improve the volume utilization of an individual layer.

In the proposed ***hybrid-ACS-2*** algorithm, first the container is filled layer-by-layer by the heuristic filling procedure. Afterwards, a utilization-threshold-level (UTL) is determined and the layers of the obtained solution are evaluated according to this value. That is a pre-evaluation to each layer have been applied. According to this; if a layer having volume utilization above the UTL is found, this layer is saved as it and the boxes in this layer is removed from the problem. Otherwise, the boxes in the layers having volume utilization below the UTL are added to the set of available box list. Thus, the numbers of boxes in the problem is reduced and the layers with high volume utilization are saved. Following this step, the layers above the UTL level are allocated to the container and the dimensions of the container are updated. Then, the ACO algorithm is applied to the reduced problem and the overall volume utilization of the container is calculated. The steps of the proposed hybrid algorithm are illustrated in Figure 6.2.

Different from the ***hybrid-ACS-2*** algorithm, in the ***hybrid-ACS-1*** algorithm a pre-evaluation of the layers is not available.

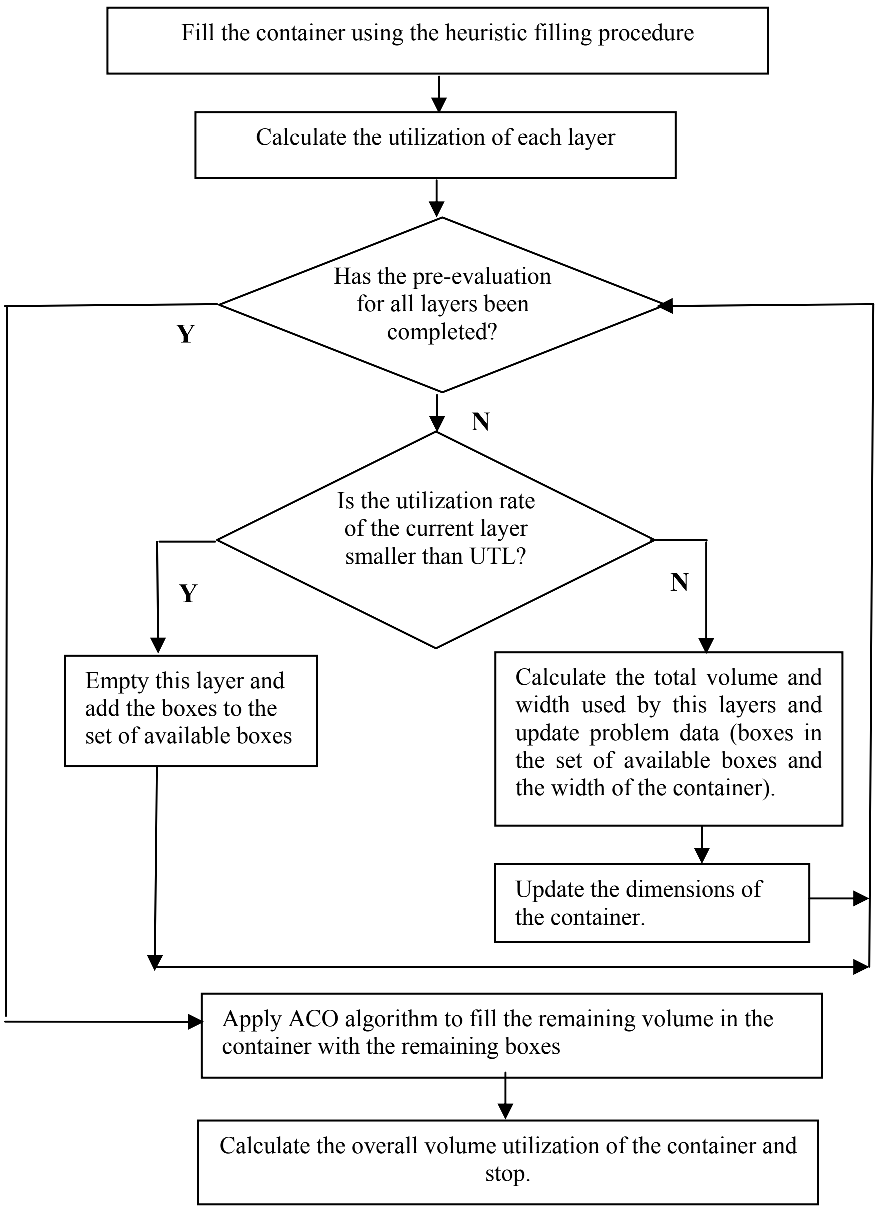

Fill the container using the heuristic filling procedure
Calculate the utilization of each layer
Has the pre-evaluation for all layers been completed?
Y
N
Is the utilization rate of the current layer smaller than UTL?
Y
N
Empty this layer and add the boxes to the set of available boxes
Calculate the total volume and width used by this layers and update problem data (boxes in the set of available boxes and the width of the container).
Update the dimensions of the container.
Apply ACO algorithm to fill the remaining volume in the container with the remaining boxes
Calculate the overall volume utilization of the container and stop.

Figure 6.2. The steps of the ***hybrid-ACS-2*** algorithm (Dereli and Daş, 2010c)

6.3.1. Determination of the parameters of the hybrid-ACS-1 and hybrid-ACS-2 algorithms

A general ACO algorithm has a number of control parameters that affect the performance of the algorithm. These parameters are shown in Table 6.2.

Table 6.2 General ACO parameters (Engelbrecht, 2005).

Parameter	**Meaning**	**Comment**
n_k	Number of ants	
n_t	Maximum number of iterations	
τ_0	Initial pheromone amount	not for Max Min Ant System
p	Pheromone persistence	p_1 , p_2 for ACS
α	Pheromone intensification	$\alpha=1$ for ACS
β	Heuristic intensification	Not for SACO, ANTS, between 1 and 10

In order to determine the parameters of ***hybrid-ACS-1*** and ***hybrid-ACS-2*** algorithms, factorial design is used.

A factorial analysis with three control parameters; namely, beta β, number of ants m and number of iterations $iter$ is presented in order to demonstrate the effects of ***hybrid-ACS-1*** parameters on the solution quality. Design of experiment with three control parameters of the ***hybrid-ACS-1*** algorithm is given in Table 6.3 and 6.4. In order to test the effects of these control parameters (factors), 10 problems from the BR7 test cases with 3*3*2 (= 18) different setting is run 3 times with different seed and totally 54 experiments are conducted. Minitab statistical software is used for the analysis.

Table 6.3. Levels of factors for the factorial design

Factors	**Levels**		
beta β	1	2	5
number of ants m ,	2	4	6
number of iterations $iter$	1000	5000	

The results of the design of experiment have revealed that only factors: $iter$ and m significantly affects the performance of the proposed algorithm. The only significant two-way interaction is detected between parameters β and m. Since the F-ratio of the parameter m is higher than the others, it can be concluded that it is the most significant parameter. This reveals that the number of ants (parameter m) in the algorithm affects the solution quality of the algorithm. Apart from these interactions, there are no significant two-way and three-way interactions between the parameters. Main Effects plots and Interaction plots of this analysis are also shown in Figure 6.3 and 6.4. As a result of this analysis, the beta β value of 5, number of ants m value of 6 and number of iterations $iter$ value of 5000 are determined for the proposed ***hybrid-ACS-1*** algorithm.

Table 6.4. Factorial design on ***hybrid-ACS-1*** response

```
Analysis of Variance for C8, using Adjusted SS for Tests

Source        DF     Seq SS     Adj SS     Adj MS       F      P

m              2  0,0044598  0,0044598  0,0022299  135,61  0,000
iter           1  0,0005143  0,0005143  0,0005143   31,28  0,000
beta           2  0,0000115  0,0000115  0,0000057    0,35  0,708
m*iter         2  0,0000189  0,0000189  0,0000094    0,57  0,568
m*beta         4  0,0007430  0,0007430  0,0001858   11,30  0,000
iter*beta      2  0,0000226  0,0000226  0,0000113    0,69  0,509
m*iter*beta    4  0,0000476  0,0000476  0,0000119    0,72  0,582
Error         36  0,0005920  0,0005920  0,0000164
Total         53  0,0064097

S = 0,00405505   R-Sq = 90,76%   R-Sq(adj) = 86,40%
```

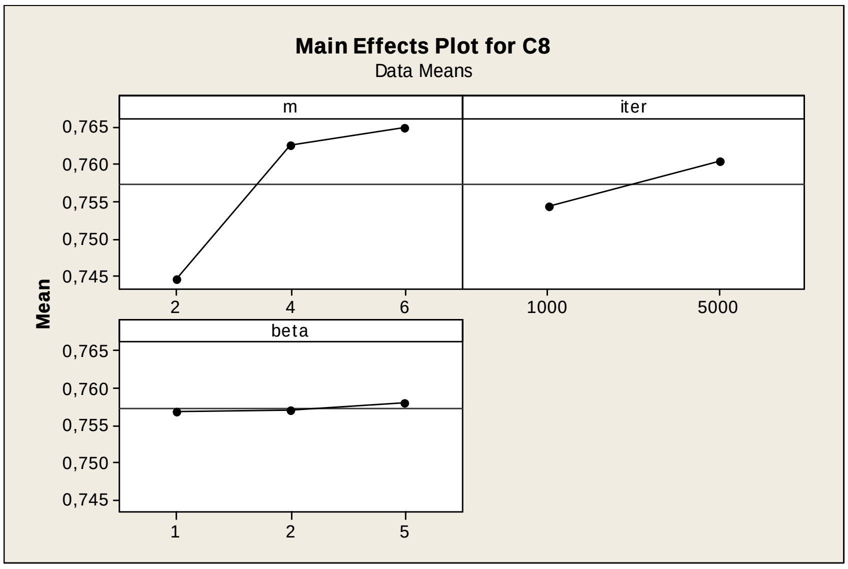


Figure 6.3 Main effects plots

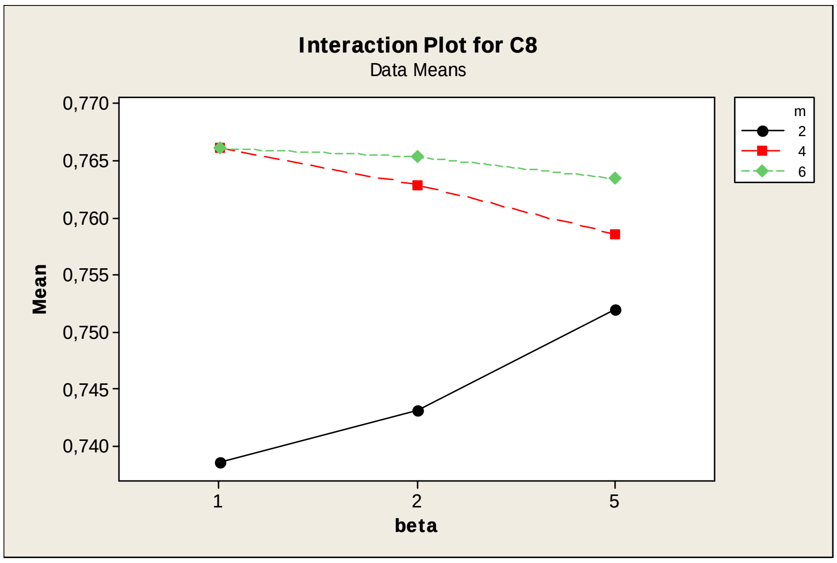


Figure 6.4 Interaction plots

A factorial analysis with four control parameters; namely, beta β, number of ants m, number of iterations $iter$ and utilization-threshold-level $level$ is presented in order to demonstrate the effects of ***hybrid-ACS-2*** parameters on the solution quality. Design of experiment with four control parameters of the ***hybrid-ACS-2*** algorithm is given in Table 6.5 and 6.6. 10 problems with 3*3*2*2 (= 36) different setting is run 3 times with different seed and totally 108 experiments are conducted for the ***hybrid-ACS-2.***

Table 6.5. Levels of factors for the factorial design

Factors		**Levels**	
beta β	1	2	5
number of ants m,	2	4	6
number of iterations $iter$	1000	5000	
utilization-threshold-level $level$	0,8	0,85	

The results of the design of experiment have also revealed that factors: β, m, $iter$ and $level$, all significantly affects the performance of the proposed algorithm. Among these, the parameter m is again the most significant parameter with its high F-ratio. It is observed that for the ***hybrid-ACS-2*** algorithm**,** the parameter $level$ is also a significant parameter. However, the parameters $iter$ and beta are not so significant parameters for this algorithm. The findings show that both the number of ants (parameter m) in the algorithm and the selected utilization-threshold-level (parameter $level$) affect the solution quality of the algorithm.

Table 6.6. Factorial design on ***hybrid-ACS-2*** response

```
Analysis of Variance for C9, using Adjusted SS for Tests

Source              DF     Seq SS     Adj SS     Adj MS       F      P
beta                 2  0,0001590  0,0001590  0,0000795    5,00  0,009
m                    2  0,0040857  0,0040857  0,0020428  128,48  0,000
iter                 1  0,0002789  0,0002789  0,0002789   17,54  0,000
level                1  0,0010490  0,0010490  0,0010490   65,97  0,000
beta*m               4  0,0001746  0,0001746  0,0000436    2,74  0,035
beta*iter            2  0,0000063  0,0000063  0,0000031    0,20  0,821
beta*level           2  0,0000302  0,0000302  0,0000151    0,95  0,392
m*iter               2  0,0002503  0,0002503  0,0001252    7,87  0,001
m*level              2  0,0001409  0,0001409  0,0000705    4,43  0,015
iter*level           1  0,0000068  0,0000068  0,0000068    0,43  0,514
beta*m*iter          4  0,0000200  0,0000200  0,0000050    0,31  0,867
beta*m*level         4  0,0000992  0,0000992  0,0000248    1,56  0,194
beta*iter*level      2  0,0000041  0,0000041  0,0000021    0,13  0,878
m*iter*level         2  0,0000237  0,0000237  0,0000119    0,75  0,478
beta*m*iter*level    4  0,0000490  0,0000490  0,0000123    0,77  0,548
Error               72  0,0011448  0,0011448  0,0000159
Total              107  0,0075226

S = 0,00398752   R-Sq = 84,78%   R-Sq(adj) = 77,38%
```

There are also significant two-way interactions between β and m, m and $iter$ and m and $level$. However, they are not so significant. The analysis also shows that there is no significant three-way and four-way interactions of the parameters. Main Effects plots and Interaction plots of this analysis are also shown in Figure 6.5 and 6.6. As a result of this analysis, the beta β value of 5, number of ants m value of 6, number of iterations $iter$ value of 5000 and utilization-threshold-level $level$ of value 0,8 are determined for the proposed ***hybrid-ACS-2*** algorithm.

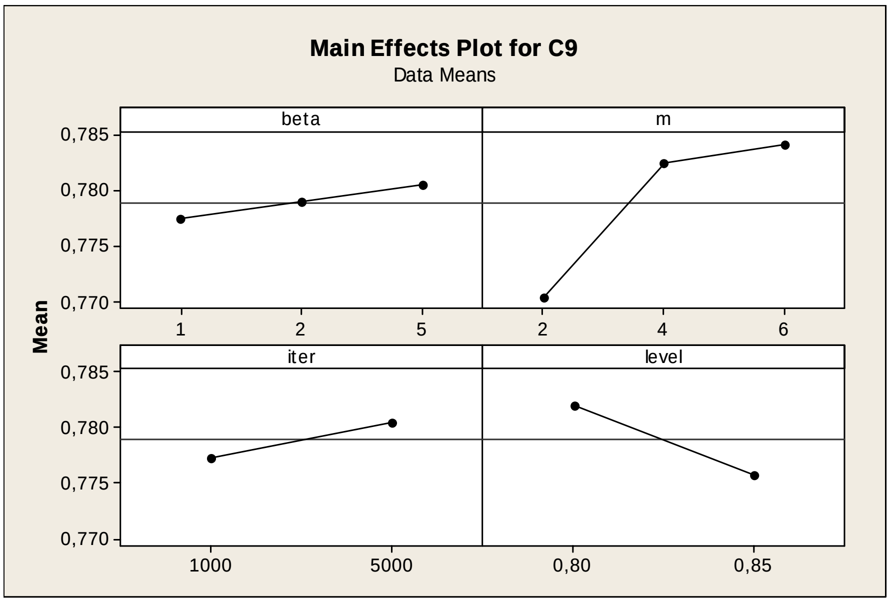


Figure 6.5 Main effects plots

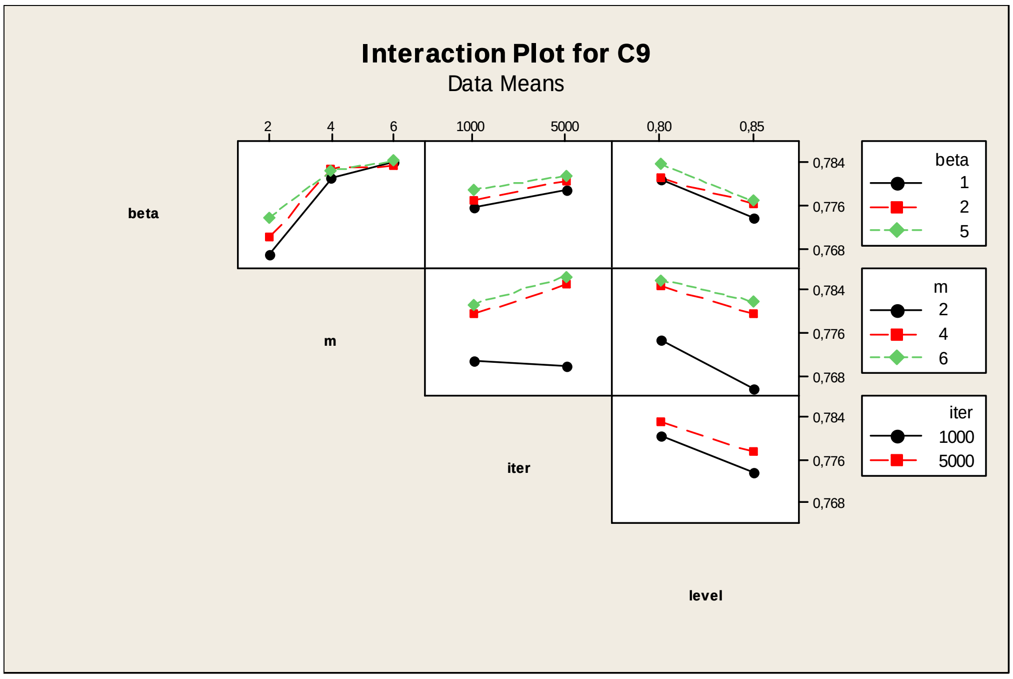


Figure 6.6 Interaction plots

As a result of this analysis, the β value of 5, m value of 6, *iter* value of 5000 are determined for the ***hybrid-ACS-1*** algorithm and the β value of 5, m value of 6, *iter* value of 5000 and *level* (UTL) of value 0,8 are determined for the proposed ***hybrid-ACS-2*** algorithm. Finally, initial pheromone amount τ_0 is set to 0.001, α is set to 1, p_1 is set to 0.9 and p_2 is set to 0.9.

6.4. Computational work

The proposed ACO algorithms are also tested on LN and BR test cases. These test cases are also solved by some of the previous work using different heuristic and meta-heuristic algorithms. The results obtained by the proposed algorithm for these test cases along with the previously reported results are presented in Table 6.7 and 6.8.

Table 6.7. Comparative results with test cases of LN – heuristic approaches

Problem	Loh and Nee (1992)*	Ngoi et al. (1994)	Bischoff et al. (1995)	Bischoff and Ratcliff (1995)	Eley (2002)	Bischoff (2003)	Lim et al. (2005)
LN01	78.1	62.5	62.5	62.5	62.5	NC*	62.5
LN02	76.8	80.7	89.7	90.0	90.8	NC	80.4
LN03	69.5	53.4	53.4	53.4	53.4	NC	53.4
LN04	59.2	55.0	55.0	55.0	55.0	NC	55.0
LN05	85.8	77.2	77.2	77.2	77.2	NC	76.7
LN06	88.6	88.7	89.5	83.1	87.9	NC	84.8
LN07	78.2	81.8	83.9	78.7	84.7	NC	77.0
LN08	67.6	59.4	59.4	59.4	59.4	NC	59.4
LN09	84.2	61.9	61.9	61.9	61.9	NC	61.9
LN10	70.1	67.3	67.3	67.3	67.3	NC	67.3
LN11	63.8	62.2	62.2	62.2	62.2	NC	62.2
LN12	79.3	78.5	76.5	78.5	78.5	NC	69.5
LN13	77.0	84.1	82.3	78.1	85.6	NC	73.3
LN14	69.1	62.8	62.8	62.8	62.8	NC	62.8
LN15	65.6	59.5	59.5	59.5	59.5	NC	59.5
Mean	**74.2**	**69.0**	**69.5**	**68.6**	**69.9**	**-**	**67.0**

Table 6.8. Comparative results with test cases of LN – metaheuristic approaches

Problem	Gehring et al. (1997)	Bortfeldt et al. (1998)	Bortfeldt et al. (2001)	Gehring and Bortfeldt (2002)	Bortfeldt, Gehring, Mack (2003)	Moura and Oliveira (2005)	**Hybrid-ACS-1** (this study)	**Hybrid-ACS-2** (this study)
LN01	62.5	62.5	62.5	NC*	NA**	NA	62,5	62,5
LN02	90.7	96.7	89.8	NC	NA	NA	84,3	80,8
LN03	53.4	53.4	53.4	NC	NA	NA	53,4	53,4
LN04	55.0	55.0	55.0	NC	NA	NA	55,0	55,0
LN05	77.2	77.2	77.2	NC	NA	NA	77,2	77,2
LN06	91.1	96.3	92.4	NC	NA	NA	82,5	85,2
LN07	82.7	84.7	84.7	NC	NA	NA	82,9	84,0
LN08	59.4	59.4	59.4	NC	NA	NA	59,4	59,4
LN09	61.9	61.9	61.9	NC	NA	NA	61,9	61,9
LN10	67.3	67.3	67.3	NC	NA	NA	67,3	67,3
LN11	62.2	62.2	62.2	NC	NA	NA	62,2	62,2
LN12	78.5	78.5	78.5	NC	NA	NA	74,8	77,3
LN13	85.6	85.6	85.6	NC	NA	NA	81,6	81,6
LN14	62.8	62.8	62.8	NC	NA	NA	62,8	62,8
LN15	59.5	59.5	59.5	NC	NA	NA	59,5	59,5
Mean	**70.0**	**70.9**	**70.1**	**-**	**70.9**	**70.3**	**68,5**	**68, 7**

*These values are not computed in this study, ** These values are not available separately.

As can be seen from Table 6.8, ***hybrid-ACS-2*** performs better than the ***hybrid-ACS-1*** algorithm**.** The ***hybrid-ACS-2*** finds optimal solutions for all problems except for problems 2, 6, 7, 12 and 13. When the performance of the ***hybrid-ACS-2*** algorithm is compared to the heuristic approaches available in the literature, the performance gap between the best performing algorithm and the ***hybrid-ACS-2*** algorithm is only 1.72 %. When compared to the meta-heuristic approaches, the performance gap between the best performing algorithm and the ***hybrid-ACS-2*** is 3.1 %. Since the ***hybrid-ACS-2*** algorithm performs better than the ***hybrid-ACS-1*** algorithm**,** BR test cases are only solved using the ***hybrid-ACS-2*** algorithm. The obtained results along with the previously reported results are presented in Table 6.9 and 6.10**.**

Table 6.9. Comparative results with the test cases of BR – heuristic approaches

Problem	Bischoff et al. (1995)	Bischoff and Ratcliff (1995)	Eley (2002)	Bischoff (2003)	Lim et al. (2005)
BR1	81.76	83.79	NA*	89.39	87.40
BR2	81.70	84.44	NA*	90.26	88.70
BR3	82.98	83.94	NA*	91.08	89.30
BR4	82.60	83.71	NA*	90.90	89.70
BR5	82.76	83.80	NA*	91.05	89.70
BR6	81.50	82.44	NA*	90.70	89.70
BR7	80.51	82.01	NA*	90.44	89.40
Mean	**81.97**	**83.50**	**88.75**	**90.55**	**89.13**

Table 6.10. Comparative results with the test cases of BR – metaheuristic approaches

Problem	Gehring et al. (1997)	Bortfeldt et al. (1998)	Bortfeldt et al. (2001)	Gehring, Bortfeldt (2002)	Bortfeldt, et al. (2003)	Mack et al. (2004)	Moura, Oliveira (2005)	**Hybrid-ACS-2** (this study)
BR1	85.80	92.63	87.81	88.10	93.52	93.70	89.07	77.75
BR2	87.26	92.70	89.40	89.56	93.77	94.30	90.43	79.41
BR3	88.10	92.31	90.48	90.77	93.58	94.54	90.86	80.41
BR4	88.04	91.62	90.63	91.03	93.05	94.27	90.42	80.40
BR5	87.86	90.86	90.73	91.23	92.34	93.83	89.57	79.94
BR6	87.85	90.04	90.72	91.28	91.72	93.34	89.71	79.87
BR7	87.68	88.63	90.65	91.04	90.55	92.5	88.05	79.23
Mean	**87.50**	**91.26**	**90.10**	**90.43**	**92.70**	**93.78**	**89.73**	**79.57**

An average of 79.57% volume utilization is obtained for the BR test cases with the **hybrid-ACS-2** algorithm. Compared with other approaches for the BR test cases, there is a performance gap of 14.16% and a performance gap of 12.12% with the best performing meta-heuristic approach and the best performing heuristic approach, respectively.

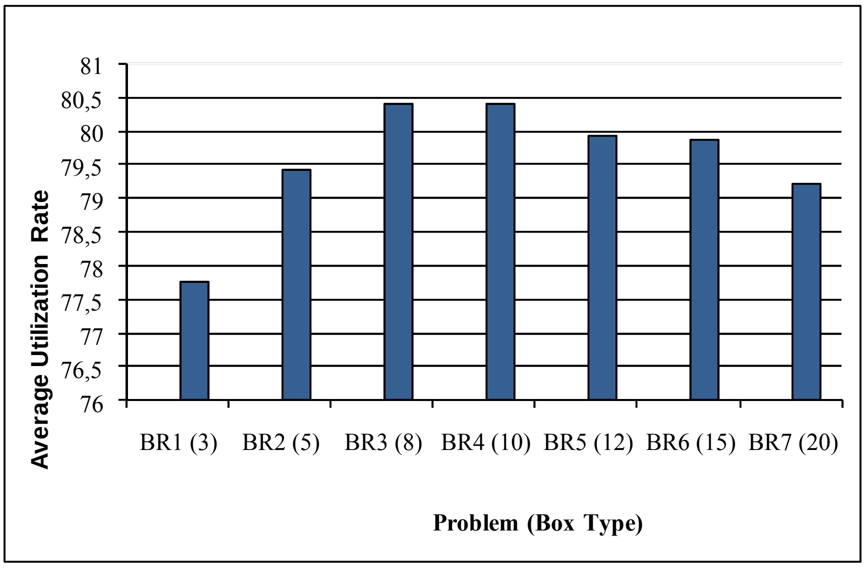


Figure 6.7. Performance of the **hybrid-ACS-2** algorithm for different problems

The performance of the proposed algorithm is changed with the number of box types as can be seen in Figure 6.7. As the number of box type increase; first a sharp increase, but then a gradual decrease is observed (similar to the proposed BA presented in Chapter 5). The best utilization rate is obtained for the BR3 problem set with 8 different box types.

In Figure 6.8, the convergence graph of the **hybrid-ACS-2** algorithm for a problem from BR7 is presented. It is observed that the algorithm converges after 300 iterations (approximately) which is reasonable.

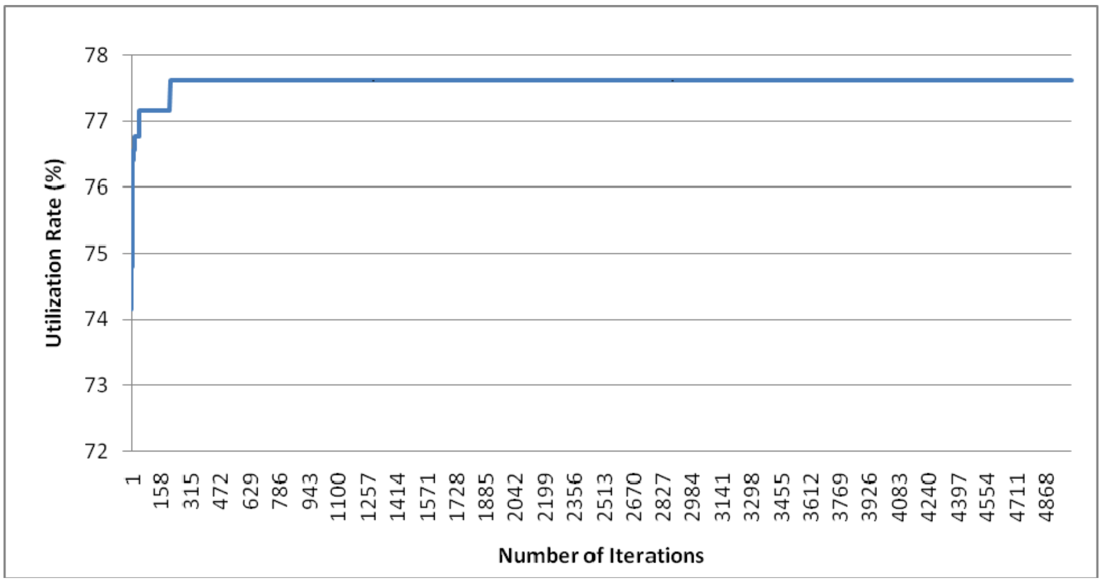


Figure 6.8 Convergence graph of a problem from test case BR7

6.5. Conclusion

In this chapter, two different algorithms based on ACO named ***hybrid-ACS-1*** and ***hybrid-ACS-2*** is presented. The ***hybrid-ACS-2*** algorithm is proposed in an attempt to improve the performance of the ***hybrid-ACS-1*** algorithm by reducing the number of nodes in the solution (each node represents a box to be filled into the container). The performances of both the ***hybrid-ACS-1*** and its later version ***hybrid-ACS-2*** are evaluated via the tests on well known test cases. The comparison between the proposed approaches and other approaches available in the literature indicated that ACO based solution approaches proposed in this study are not very high performing. The reasons behind this situation are discussed in detail in Chapter 9.

CHAPTER 7

CONTAINER LOADING SUPPORT SYSTEM

7.1. Introduction

This chapter describes the container loading support system (CLSS) developed to determine and visualize the container packing pattern of a CL process. The proposed CLSS composes of three main components; a Bees Algorithm and an Ant Colony System as the computational algorithms, the graphical user interface (GUI) and a simulation program. The aim of the designed system is to make the packing pattern more visible to the user in order to simplify the loading process. An illustrative example – a CL problem from the literature - is also provided to introduce the operation of the system and to prove its efficiency.

In the previous chapter, the nature of the CL problem and its importance for industrial applications is discussed briefly. Even if a CL problem is solved to its optimum, packing a shipment into a container is a complex process. It often takes several days to allocate the pooled goods into the number of containers and then to pack the allocated goods into the containers. Occasionally, workers must unload some containers and then reload them in a different pattern to pack more goods in the containers (Chien and Deng, 2002). Chien and Deng (2002) were the first researchers that realized this difficulty. They proposed a decision support system (DSS) based on a heuristic packing procedure. Another DSS for a similar problem namely Air-Cargo Loading Problem was proposed by Chan et al. (2006). The two-phased system is proposed to load air cargo pallets efficiently using Linear Programming and a heuristic.

Different from the previously proposed DSS that are using heuristic algorithms, the designed CLSS is using SI based algorithms namely Bees Algorithm and Ant Colony

System as the core of the system. The details about these algorithms are previously presented in Chapter 5 and 6. Other than the computational algorithms, the CLSS has a graphical user interface (GUI) and a simulation program that visualize the actual packing process in a 3D manner.

7.2. The Container Loading Support System

The CLSS composed of the computational algorithms module containing two algorithms (one is a ant colony based algorithm and the other one is a recently new SI-based algorithm called Bees Algorithm), the GUI and a simulation program that visualize the actual packing process. The data flow of the system is schematized in Figure 7.1 (Dereli and Daş, 2010a).

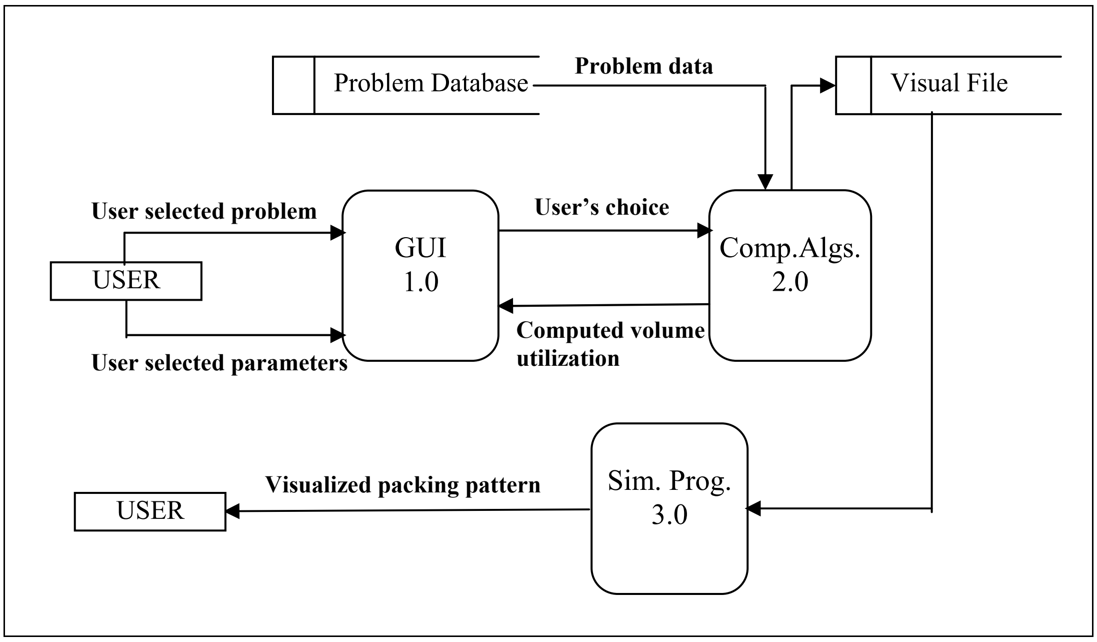


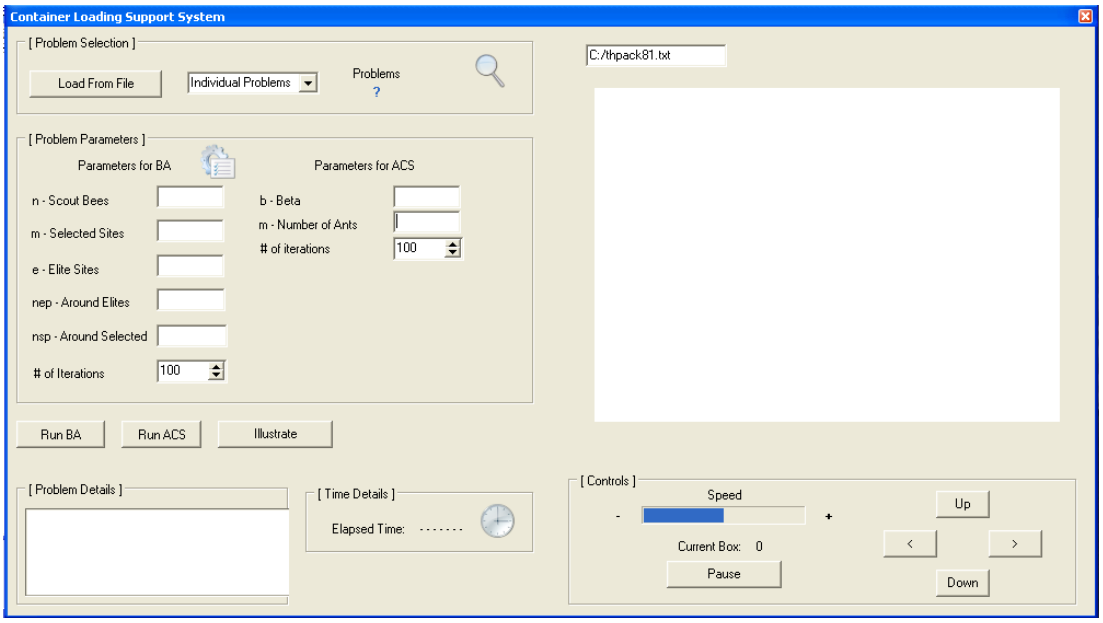
Figure 7.1. Data Flow Diagram of the CLSS system (Dereli and Daş, 2010a).

In order to operate the CLSS, the user should select the parameters of the computational algorithm and problem data from the GUI. If the parameters of the BA or the ACS are selected by the user, it is possible to run the relevant algorithms and see the obtained results found by the selected algorithm. By comparing the obtained results, the user could view the best result provided via CLSS. The parameters of the BA algorithm and the ACS algorithm are introduced in Chapter 5 and 6 in detail.

Later, this data is sent to the computational algorithm where actual processing is done. By using the corresponding problem data (obtained from the problem database) and the parameters defined previously by the user, the module containing both algorithms starts working. When the pre-determined number of iterations is reached, the data about the volume utilization of the solution is sent to the GUI and the data about the position of each box is sent to a file called visual file. The visual file contains the x, y and z coordinates of each box of the final solution computed by the computational algorithm. Finally, the simulation program visualizes the final packing pattern using the visual file.

7.3. The Graphical User Interface and the Simulation Program

The GUI of the CLSS is designed in Borland Builder. This interface is used to select the problem type and the parameters of the computational algorithms and to visualize the packing pattern. The snapshot of the interface is supplied in Figure 7.2.



Individual Problems

# of Iterations 100

# of iterations 100

C:/thpack81.txt

Elapsed Time: -------

Current Box: 0

Figure 7.2. The snapshot of the GUI

A simulation program is used to visualize the packing pattern. The simulated program is coded in OpenGL and integrated to the GUI for user friendly use.

7.4. An Illustrative Application

The following example is supplied to demonstrate how the CLSS system works in a step-by step manner. The problem is selected from the LN (1992) test cases. The aim is to allocate a set of boxes with varying dimensions into a container without any overlap to maximize the volume utilization of the container for each problem.

Step 1: The user selects a test case that is composed of a set of problems from the GUI and determines the individual problem from this set. The snapshot of the CLSS presented in Figure 7.3 illustrates the Problem Selection window used for this operation. Suppose that the user selects the LN test cases and problem number 12. A container having a width of 3200, depth of 2400 and height of 1000 units and a total of 120 boxes of 6 different box types is considered in this example. The dimensions of the boxes corresponding to the selected problem are presented in Table 7.1.

Step 2: Following the selection of the problem from the Problem Selection window, the user enters the parameters of the preferred computational algorithm directly to the cells in the Problem Parameters section. The Problems Parameters section includes the parameters regarding the computational algorithms. These parameters are; number of scout bees *n*, number of selected sites *m*, number of elite sites *e* chosen from *m* sites, number of bees recruited to search *e* elite sites *nep,* number of bees recruited to search *m-e* other sites *nsp* and maximum number of iterations as the termination criteria for the BA algorithm and beta β, number of ants *m* and maximum number of iterations as the termination criteria for the ACS algorithm.

Table 7.1. Data about the user selected problem

Type	Width	Depth	Height	Number of boxes
1	900	275	200	10
2	400	350	275	33
3	1200	300	250	10
4	500	375	275	27
5	800	400	200	15
6	600	300	225	25

Suppose that, for the solution of the selected LN problem the following set of BA parameters is used; number of bees (population) n = 5, number of selected sites m = 3, number of elite sites e = 2, number of bees send to elite points nep = 4, number of bees send to other selected points nsp = 3 and maximum number of iterations = 100. Although such a choice of parameters is used for this example, the user is given the opportunity to try different values for the parameters of the computational algorithm.

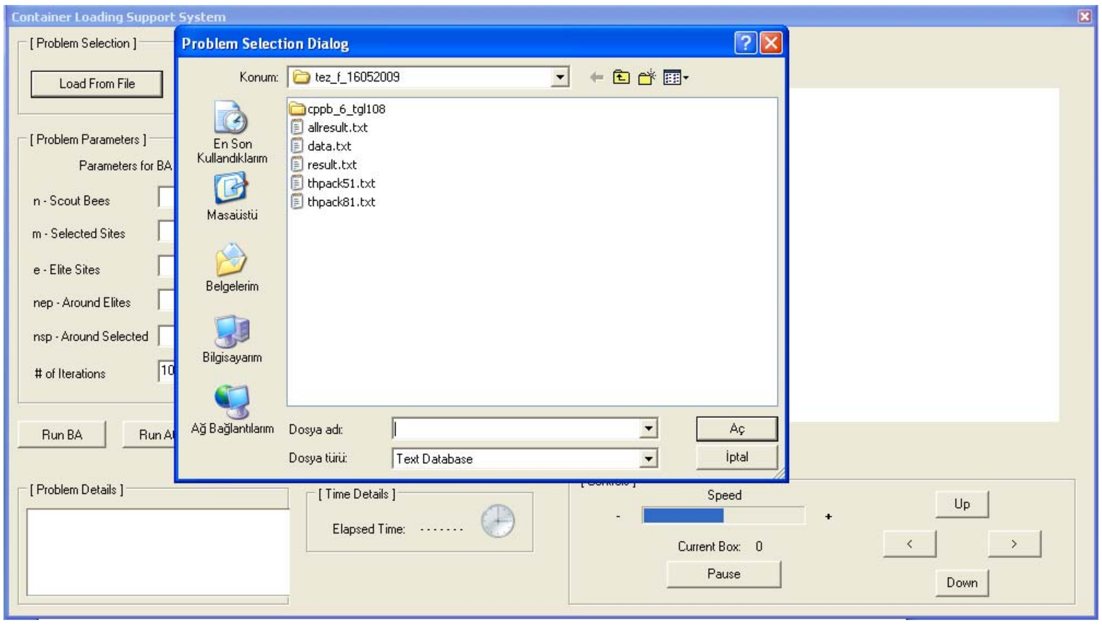


Figure 7.3. The dialog window for the Problem Selection

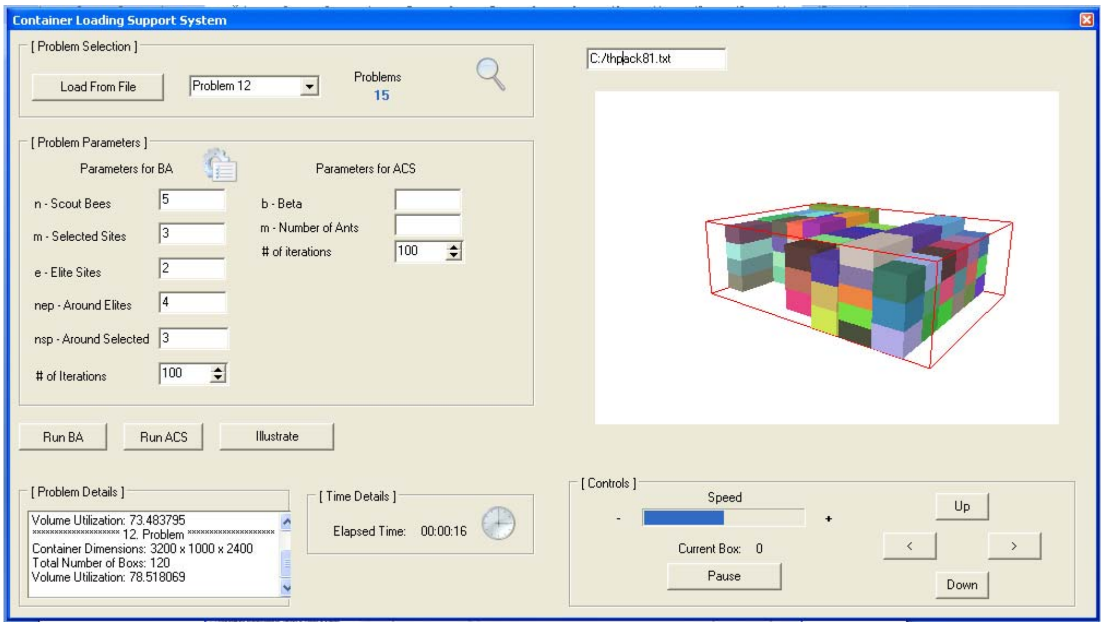
Step 3: Having selected the problem and the corresponding parameters of the selected algorithm, the user can run the selected computational algorithm by clicking the “RUN ACS” or “RUN BA” (one at a time).

Step 4: The BA algorithms work until the number of iterations are reached. Then, the data about the selected problem and the obtained solution is viewed in the *Problem Details* section of the GUI. Besides the value of the obtained solution, it is also possible to learn the elapsed time passed to solve the problem in *Time Details* section.

Step 5: When the user clicks the “ILLUSTRATE” button on the spreadsheet, the simulation program attached to the GUI executes and the program visualize the packing pattern. The program reads the x, y and z coordinates of each box from the visual data file and shows how each box is loaded into the 3D container one by one.

The resultant 3D graph of the packing pattern can be viewed from different angles using the arrow buttons located below the three dimensional graph. In the three dimensional graph each loaded box is shown with different colors. It is also possible both to pause the simulation by using the "PAUSE" button and to adjust the speed of the simulation by using the "-/+" buttons in both sides of the Speed bar.

Figure 7.4 illustrates the final packing pattern for the selected problem together with the volume utilization of the problem in the *Problem Details* section. As can be seen from Figure 7.4 and 7.5, the volume utilization of 78.51% (optimal value for this problem) is reached in 16 seconds.



Container Loading Support System

[Problem Selection]
Problem 12
Problems 15
C:/thpack81.txt

[Problem Parameters]
Parameters for BA
n - Scout Bees: 5
m - Selected Sites: 3
e - Elite Sites: 2
nep - Around Elites: 4
nsp - Around Selected: 3
# of Iterations: 100

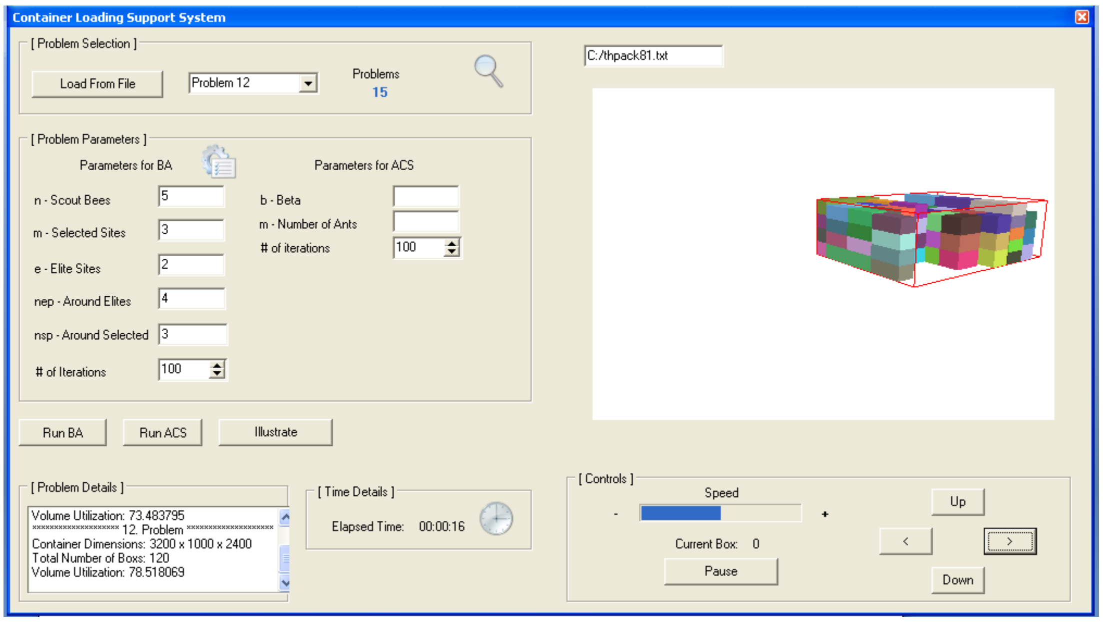
Parameters for ACS
b - Beta:
m - Number of Ants:
# of iterations: 100

[Problem Details]
```
Volume Utilization: 73.483795
******************** 12. Problem ********************
Container Dimensions: 3200 x 1000 x 2400
Total Number of Boxs: 120
Volume Utilization: 78.518069
```

[Time Details]
Elapsed Time: 00:00:16

[Controls]
Current Box: 0

Figure 7.4. The final packing pattern obtained for the problem with 120 boxes – BA algorithm


Container Loading Support System

[Problem Selection]
Load From File
Problem 12
Problems
15

C:/thpack81.txt

[Problem Parameters]
Parameters for BA
n - Scout Bees 5
m - Selected Sites 3
e - Elite Sites 2
nep - Around Elites 4
nsp - Around Selected 3
# of Iterations 100

Parameters for ACS
b - Beta
m - Number of Ants
# of iterations 100

Run BA
Run ACS
Illustrate

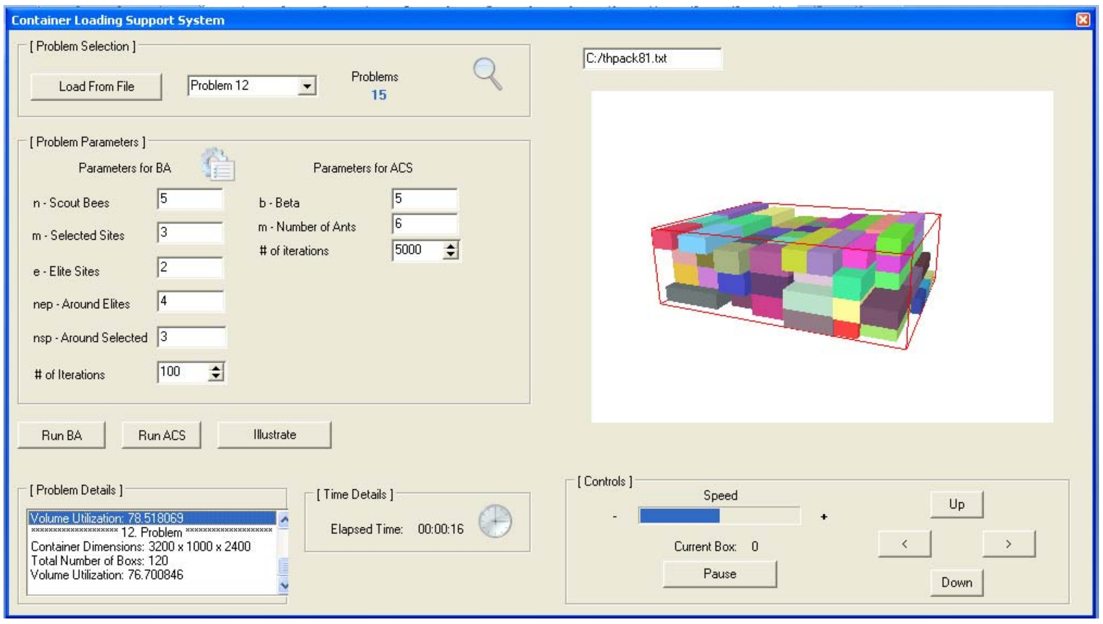
[Problem Details]
```
Volume Utilization: 73.483795
******************* 12. Problem *******************
Container Dimensions: 3200 x 1000 x 2400
Total Number of Boxs: 120
Volume Utilization: 78.518069
```

[Time Details]
Elapsed Time: 00:00:16

[Controls]
Speed
-
+
Current Box: 0
Pause
Up
<
>
Down

Figure 7.5. The final packing pattern obtained for the problem with 120 boxes- BA algorithm (scene from a different angle)

It is possible to solve the same problem with the ACS algorithm. When the current problem is solved with the ACS algorithm (the parameters that are determined in Chapter 6 is used), the final packing is supplied in Figure 7.6.


Container Loading Support System

[Problem Selection]
Load From File
Problem 12
Problems
15

C:/thpack81.txt

[Problem Parameters]
Parameters for BA
n - Scout Bees 5
m - Selected Sites 3
e - Elite Sites 2
nep - Around Elites 4
nsp - Around Selected 3
# of Iterations 100

Parameters for ACS
b - Beta 5
m - Number of Ants 6
# of iterations 5000

Run BA
Run ACS
Illustrate

[Problem Details]
```
Volume Utilization: 78.518069
******************* 12. Problem *******************
Container Dimensions: 3200 x 1000 x 2400
Total Number of Boxs: 120
Volume Utilization: 76.700846
```

[Time Details]
Elapsed Time: 00:00:16

[Controls]
Speed
-
+
Current Box: 0
Pause
Up
<
>
Down

Figure 7.6. The final packing pattern obtained for the problem with 120 boxes – ACS algorithm

7.5. Conclusion

In this chapter, a CLSS to determine and visualize the container packing pattern of a CL process is presented. The system composes of three main components; the computational algorithms based on SI-based algorithm (BA and ACS) hybridized with a heuristic filling procedure, the GUI and the Simulation Program. The module of the Computational Algorithms solves the selected problem by using the input from the GUI. Here the user could run both algorithms and could view the best obtained solution found by these algorithms via CLSS. The GUI, which is coded in Borland Builder, enables the determination of problem related choices by the user (input to the Computational Algorithms), and the Simulation Program which is coded via OpenGL, illustrates the final packing pattern in a 3D manner. The CLSS is also suitable for decision makers

The presented Simulation Program, which is a component of the proposed DSS, can be used independent of the CLSS to visualize a packing pattern computed with a different algorithm especially to check if there occurs an overlapping between boxes or not. Thus, this component of the system can also be used for academic work on CL problems where visualizing the packing pattern is necessary.

CHAPTER 8

MULTI-OBJECTIVE CONTAINER LOADING PROBLEM

8.1. Introduction

As briefly discussed in Chapter 3, many approaches have been developed to solve CL problems along with many practical constraints and different objective functions. There are several reasons of this popularity as reported in Ertek and Kılıç (2006). First of all, the CL problem is a NP-hard problem (Pisinger, 2002) and it has been recognized that it has a wide range of industrial applications.

Despite this considerable attention, studies on CL problems with multiple objectives are very limited (Liu et al., 2006). This issue was also pointed out in Dyckhoff (1990) in his typology where he stated that for many CPP, more than one objective has to be considered. One of the main contributions of this thesis is to study CL problems with multiple objectives which are frequently encountered in a typical transportation process. The case, from which the described problem is motivated, is presented in the following Section. Next, the described problem and problem assumptions are supplied in Section 8.3. In order to solve the mentioned problem, two multi-objective optimization methods; goal programming and weighted-sum approach are utilized. Due to the complexity of the model (which is presented in Chapter 3), for both approaches a SA algorithm hybridized with a heuristic filling procedure (previously described in Chapter 4) is used to solve the offered models. Some introductory information about the SA is given in Section 8.4. Finally, the computational work and the relevant results obtained as a result of the both model are shared in Sections 8.5.

8.2. The Case Study

A medium-sized distribution company in Gaziantep (a metropolitan province located in south-east of Turkey) delivers goods ordered by supermarkets which are located in the south-east part of Turkey. The firm mainly distributes *Procter and Gamble*'s products and their own products (paper towels, toilet tissues, paper napkins, etc.) which are manufactured at their own plant in Gaziantep, Turkey. The orders are stored in a database and a decision-maker decides which orders/ products to load to their own vehicles or to rented carriers. The rented trucks are paid according to the total weight of the shipment regardless of the total volume (for example, 10 $ per ton). Thus, the decision-maker prefers to load and ship a shipment with a higher total weight into its own vehicles rather than a shipment with a low total weight. On the other hand, shipping the shipment in an on-time manner is an important issue for the company. If the decision-maker can utilize the capacity of the owned vehicle in the best possible way, he/she can guarantee the on-time delivery of all of the products in its own vehicle.

Another issue that the decision-maker should take into account is the relation between the volume of the items and the weight of the items. In particular, the weight of the *house-hold, personal care, sanitary paper products* and *shaving products* per unit volume drastically differs from one product to the other that is x cm^3 of product A, which may yield a smaller weight than y cm^3 of product B (x >>y). Having considered this situation, each day the decision-maker should load the items (boxes) - that would provide the highest total weight - to the vehicles in the best possible way to decrease transportation cost and in turn increase the profitability of the company.

Motivated from the above described situation, two objective functions for the CL problem are defined in this thesis. The first objective is to *obtain a packing pattern with maximum total weight* and the second objective is to *maximize the volume utilization of the vehicle.* The mentioned goals and the proposed models for the solution of the problem is introduced in the next Section.

8.3. Problem formulation

CL problems with multi-objectives can be defined as follows; Given a set of *n* items with width (w_i), depth (d_i), height (h_i) and weight (weight_i) and a single container with known dimensions (W, D, H) where $w_i \leq W$, $h_i \leq H$ and $d_i \leq D$, the problem is to pack items into the container without any overlap while maximizing the total weight of the packed items and the utilization rate of the container (Dereli and Daş, 2010b). The problem is solved under the following assumptions:

(1) Items are rectangular boxes defined with known dimensions (w_i, d_i, h_i)
(2) Each box can be arranged originally in the container in a maximum of 6 "rotation variants" if not prohibited.
(3) Each box can lie on the container floor or can be stacked on top of another.
(4) Stability of the box arrangements is not considered thus the use of spacing material is considered to avoid possible problems.

8.4. Simulated annealing (SA) algorithm

SA is a method for obtaining good solutions to difficult optimization problems and is introduced by Kirkpatrick et al. (1983) as an analogy to the statistical mechanics of annealing in solids. It is a non-derivative method which has received much attention over the last few years (Eglese, 1990). Similar to other non-derivative methods such as Genetic algorithms, Random search, Tabu search and complex/simplex, SA is more likely to find a global optimum and not be stuck on local optima as gradient methods might do. It is also slightly less computational expensive as compared to Genetic Algorithms (Andersson, 2000).

SA differs from iterative algorithms in that it has a mechanism which helps it to escape from local optimum and rather reach to global optimum. This is because SA not only accepts neighborhood solutions better than the current solution, but also accepts neighborhood solutions worse than current solution with a probability. This probability which is known as *acceptance probability* is related to the *temperature*, which decreases during the process. As the temperature decreases, the acceptance probability also decreases. This means that as the temperature decreases, the probability of accepting worse neighborhood solutions decreases. During the annealing process, the temperature decreases gradually. At each temperature, a predetermined number of iterations to search the solution space are conducted. The search terminates when the stopping criteria are met (Dereli and Daş, 2007).

Both the proposed goal programming model and the weighted-sum model is solved with a SA algorithm which is the adapted to solve the multi objective problems. The proposed algorithms are motivated from the work of Baykasoğlu (2005). The fundamentals of the proposed algorithm are summarized as follows;

- A solution is represented by a bit string representation. For example, bit string representation of a solution composed of seven different types of boxes, for which only the base rotation is permitted, is presented in Figure 8.1. It should be noted that the length of this string is determined by the number of different types of boxes in the problem. This structure is preferred to the structure in which each bit in the string represents a box in the problem. In a problem where there are 100 boxes of seven different box types, the second structure will yield a bit string of length 100 which will be a very inconvenient and time-consuming structure for the large-sized problems.

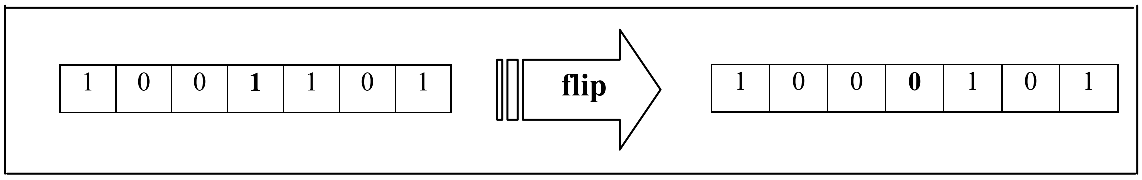


Figure 8.1. Neighborhood solution generation using the flip operator (only for base-rotated boxes)

- In order to reach neighborhood solutions, the flip operator used for the BA (explained previously in Chapter 5) is employed.

- For the goal programming approach, objective function of a solution is presented as the weighted sum of the deviations from the defined goals, whereas for the weighted-sum approach, objective function of a solution is presented as the weighted-sum of the normalized objectives.

- For the solution of the goal programming model; if a neighborhood solution having the objective value of (0.11) is obtained as a result of the flip operator, this solution should be accepted when the previous solution has an objective value of (0.12). Because the sum of the deviations are minimized in the reached neighborhood solution. However, a neighborhood solution having an objective value (0.13) could be accepted with probability or rejected since it has a worse objective value than the current solution. At this point acceptance or rejection is related to the temperature which greatly affects the acceptance probability.

For the solution of the weighted-sum model, suppose that a neighborhood solution having the objective value of (0.12) is obtained as a result of the flip operator. Then, this solution will be accepted since the previous solution has an objective value of (0.11). The value of the objective function is maximized in this newly reached neighborhood solution. However, a neighborhood solution having an objective value (0.10) could be accepted with probability or rejected since it has a worse objective value than the current solution. At this point acceptance or rejection is related to the temperature which greatly affects the acceptance probability.

A generic pseudo-code of the SA algorithm which is hybridized with the heuristic filling procedure - used for the both approaches- is presented in Table 8.1.

Table 8.1. A pseudo-code of the proposed SA

```
Step 1. Generate an initial solution and calculate the value of the objective function ($fitness_0$) using
        the heuristic filling algorithm;
        $Solution = fitness_0$;
Step 2. Parameter initialization;
        2.1. Set the annealing parameters; $T_{in}$, $T_f$, $il_{max}$ and $\alpha$.
        2.2. Read the number of box types $N$.
Step 3. Annealing Schedule;
        3.1. Inner loop initialization; $il = 0$;
        3.2. At every temperature achieve equilibrium. Execute inner loop until the condition in 3.2.4
        is met;
            3.2.1. $il = il + 1$;
            3.2.2. Generate a neighborhood solution and calculate the value of the objective function
            ($fitness_{il}$) using the heuristic filling algorithm;
            3.2.3. Accept or reject the solution as described previously;
            3.2.4. IF ($il \geq il_{max}$)
            THEN terminate inner loop and GOTO step 3.3
            ELSE continue inner loop and GOTO step 3.2.1
      3.3. $T_{iter+1} = \alpha * T_{iter}$;
      3.4. IF ($T_{iter+1} < T_f$)
      THEN terminate inner loop and GOTO step 4
      ELSE continue inner loop and GOTO step 3.1
Step 4. Terminate the best solution Solution and stop.
```

8.5. Computational work

8.5.1. Solution of container loading (CL) problems with single objective function

The proposed algorithm for the multi-objective container loading (MOCL) problem is first tested on LN and BR test cases which are composed of problems defined with a single objective function. Following choices are made for the solution of the LN test cases with the SA algorithm.

- Initial value of the temperature T_{in} is set to 200.
- To change the temperature a proportional temperature function is employed.

$$T(iter+1) = \alpha T(iter) \quad where\ 0.8 \leq \alpha \leq 0.99 \qquad (8.1)$$

α is a constant and lies between 0.8 and 0.99. In this work, the value of α is 0.987.

- The number of iterations $il_{\max}$ that should be performed at each temperature is equal to the number of box types N in each order.
- When the value of the final temperature T_f falls below 0.05 the algorithm is terminated.

LN test cases are also solved by some of the previous work using different heuristic and meta-heuristic algorithms. The results obtained by the proposed algorithm for these test cases along with the previously reported results are presented in Table 8.2 and 8.3.

Table 8.2. Comparative results with the test cases of LN – heuristic approaches

Problem	Loh and Nee (1992)*	Ngoi et al. (1994)	Bischoff et al. (1995)	Bischoff and Ratcliff (1995)	Eley (2002)	Bischoff (2003)	Lim et al. (2005)
LN01	78.1	62.5	62.5	62.5	62.5	NC*	62.5
LN02	76.8	80.7	89.7	90.0	90.8	NC*	80.4
LN03	69.5	53.4	53.4	53.4	53.4	NC*	53.4
LN04	59.2	55.0	55.0	55.0	55.0	NC*	55.0
LN05	85.8	77.2	77.2	77.2	77.2	NC*	76.7
LN06	88.6	88.7	89.5	83.1	87.9	NC*	84.8
LN07	78.2	81.8	83.9	78.7	84.7	NC*	77.0
LN08	67.6	59.4	59.4	59.4	59.4	NC*	59.4
LN09	84.2	61.9	61.9	61.9	61.9	NC*	61.9
LN10	70.1	67.3	67.3	67.3	67.3	NC*	67.3
LN11	63.8	62.2	62.2	62.2	62.2	NC*	62.2
LN12	79.3	78.5	76.5	78.5	78.5	NC*	69.5
LN13	77.0	84.1	82.3	78.1	85.6	NC*	73.3
LN14	69.1	62.8	62.8	62.8	62.8	NC*	62.8
LN15	65.6	59.5	59.5	59.5	59.5	NC*	59.5
Mean	**74.2**	**69.0**	**69.5**	**68.6**	**69.9**	**-**	**67.0**

* These values are **not computed** and/or presented. ** These values are **not available** separately.

Table 8.3. Comparative results with the test cases of LN – metaheuristic approaches

Problem	Gehring et al. (1997)	Bortfeldt et al. (1998)	Bortfeldt et al. (2001)	Gehring and Bortfeldt (2002)	Bortfeldt, Gehring, Mack (2002)	Moura and Oliveira (2005)	This work
LN01	62.5	65.5	62.5	NC*	NA**	NA**	62.5
LN02	90.7	96.7	89.8	NC*	NA**	NA**	90.1
LN03	53.4	53.4	53.4	NC*	NA**	NA**	53.4
LN04	55.0	55.0	55.0	NC*	NA**	NA**	55.0
LN05	77.2	77.2	77.2	NC*	NA**	NA**	77.2
LN06	91.1	96.3	92.4	NC*	NA**	NA**	85.8
LN07	82.7	84.7	84.7	NC*	NA**	NA**	84.2
LN08	59.4	59.4	59.4	NC*	NA**	NA**	59.4
LN09	61.9	61.9	61.9	NC*	NA**	NA**	61.9
LN10	67.3	67.3	67.3	NC*	NA**	NA**	67.3
LN11	62.2	62.2	62.2	NC*	NA**	NA**	62.2
LN12	78.5	78.5	78.5	NC*	NA**	NA**	78.1
LN13	85.6	85.6	85.6	NC*	NA**	NA**	83.9
LN14	62.8	62.8	62.8	NC*	NA**	NA**	62.8
LN15	59.5	59.5	59.5	NC*	NA**	NA**	59.5
Mean	**70.0**	**70.9**	**70.1**	**-**	**70.9**	**70.3**	**69.6**

* These values are **not computed** and/or presented.   ** These values are **not available** separately.

As can be seen from Table 8.3, the proposed algorithm in this study finds optimal solutions for all problems except for problems 2, 6, 7 and 13 *(the average computation time for all problems is 138.86 seconds)*. When the performance of the proposed algorithm is compared to the heuristic approaches available in the literature, the algorithm performs quite well after the heuristic proposed in Bischoff et al. (1995). When compared to the meta-heuristic approaches, the performance gap between the best performing algorithm and the proposed algorithm is only 1.9 %.

Following the solution of LN test cases, the test cases from BR are also solved. For these problems - which are harder as compared to LN test cases - SA parameters are re-determined experimentally where the values of the variables T_{in} is 5000, α is 0.987, $il_{\max}$ is N (equal to the number of box types in each problem) and T_f is 0.0001. The results obtained by the proposed algorithm for the test cases along with the previously reported results are presented in Table 8.4 and 8.5.

Table 8.4. Comparative results with the test cases of BR – heuristic approaches

Problem	Bischoff et al. (1995)	Bischoff and Ratcliff (1995)	Eley (2002)	Bischoff (2003)	Lim et al. (2005)
BR1	81.76	83.79	NA*	89.39	87.40
BR2	81.70	84.44	NA*	90.26	88.70
BR3	82.98	83.94	NA*	91.08	89.30
BR4	82.60	83.71	NA*	90.90	89.70
BR5	82.76	83.80	NA*	91.05	89.70
BR6	81.50	82.44	NA*	90.70	89.70
BR7	80.51	82.01	NA*	90.44	89.40
Mean	**81.97**	**83.50**	**88.75**	**90.55**	**89.13**

*These values are not available separately.

Table 8.5. Comparative results with the test cases of BR– metaheuristic approaches

Problem	Gehring et al. (1997)	Bortfeldt et al. (1998)	Bortfeldt et al. (2001)	Gehring and Bortfeldt, (2002)	Bortfeldt, Gehring, Mack, (2003)	Moura, Oliveira (2005)	This work
BR1	85.80	92.63	87.81	88.10	93.52	89.07	86.38
BR2	87.26	92.70	89.40	89.56	93.77	90.43	87.70
BR3	88.10	92.31	90.48	90.77	93.58	90.86	87.06
BR4	88.04	91.62	90.63	91.03	93.05	90.42	86.61
BR5	87.86	90.86	90.73	91.23	92.34	89.57	86.10
BR6	87.85	90.04	90.72	91.28	91.72	89.71	85.47
BR7	87.68	88.63	90.65	91.04	90.55	88.05	84.49
Mean	**87.50**	**91.26**	**90.10**	**90.43**	**92.70**	**89.73**	**86.26**

BR test cases are solved in a reasonable amount of time (195 seconds – average computation time for single problem) by using the proposed SA algorithm with a filling performance of 86.26%. When compared to the best performing algorithm, there is a performance gap of 4.6%. The results obtained for the both test cases reveal that the proposed algorithm is a suitable tool for solving container loading problems with its relatively simple structure.

The convergence graph obtained for a problem from test case BR1 is illustrated in Figure 8.2. It is obvious from the graph that, the algorithm converges after a reasonable number of iterations.

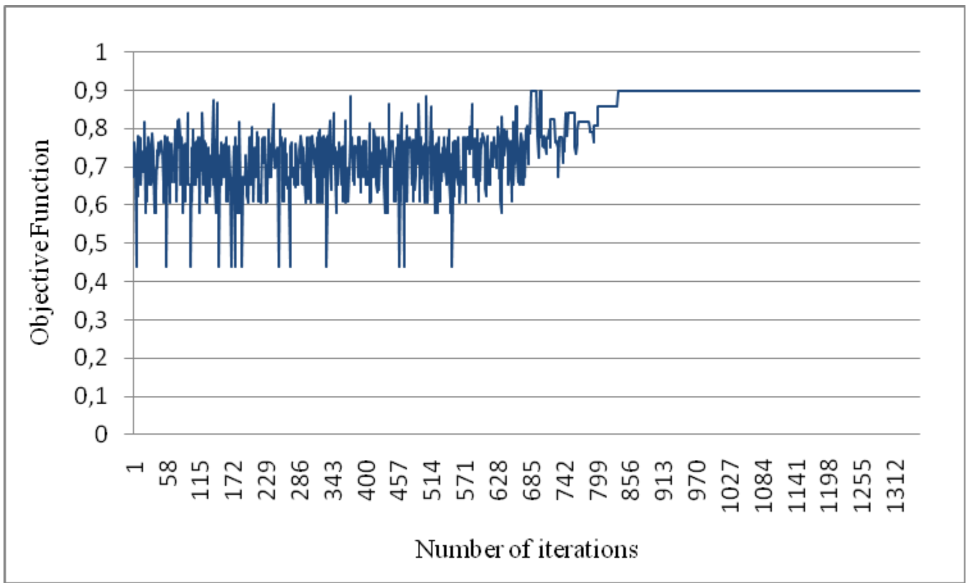


Figure 8.2. Convergence graph for the hybrid-SA algorithm

8.5.2. Solution of container loading (CL) problems with multi-objective functions (through a real example)

Many methods are available for solving Multi Objective Optimization (MMO) problems, and many of them involve converting the MOO problem into one or a series of Single Objective Optimization (SOO) problems. Each of these problems involves the optimization of a 'scalarizing' function, which is a function of original objectives, by a suitable method for SOO (Rangaiah, 2009). Thus, there are many MOO methods available. In Figure 8.3 these methods are classified as generating methods and preference-based methods.

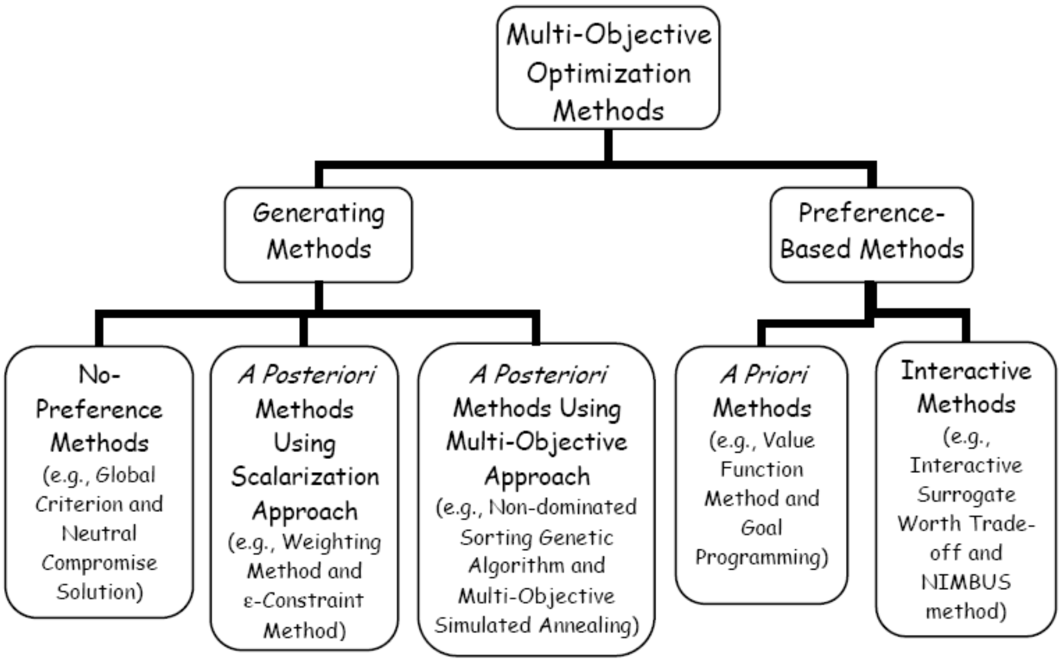

Multi-Objective Optimization Methods
Generating Methods
Preference-Based Methods
No-Preference Methods (e.g., Global Criterion and Neutral Compromise Solution)
A Posteriori Methods Using Scalarization Approach (e.g., Weighting Method and ε-Constraint Method)
A Posteriori Methods Using Multi-Objective Approach (e.g., Non-dominated Sorting Genetic Algorithm and Multi-Objective Simulated Annealing)
A Priori Methods (e.g., Value Function Method and Goal Programming)
Interactive Methods (e.g., Interactive Surrogate Worth Trade-off and NIMBUS method)

Figure 8.3. Classification of MOO methods (Rangaiah, 2009)

Generating methods generate one or more Pareto-optimal solutions without any inputs from the decision maker; on the other hand, preference-based methods utilize the preferences specified by the decision makers at some stage(s) in solving the MOO problem (Rangaiah, 2009).

No preference method, posteriori methods using scalarization and posteriori methods using multi-objective approach are generating methods. In the *No preference method*, decision maker do not articulate her/his decisions during the process. Examples of *No preference method* are the method of global criterion and multiple objective proximal bundle method.

For the methods in which the decision-maker articulates her/his preferences after the process, the decision-makers are given a set of Pareto optimal solutions from which the decision-maker is free to select the most suitable one that reflects her/his preference. *Posteriori methods using the scalarization approach* includes the e-constraint and weighting methods; whereas *Posteriori methods using Multi-objective approach* includes population-based methods such as non-dominated sorting Genetic Algorithm, multi-objective Differential Evolution and multi-objective Simulated Annealing.

Preference based methods include the *Priori methods* and *Interactive methods.* If the decision-maker articulates her/his preferences before the process, methods on *Priori articulation of preferences* are used. Value function methods, lexicographic ordering and goal programming are examples of these methods.

Methods, which include interaction with the decision makers during the solution of the problem, are called *Interaction methods*. Examples of these methods are interactive surrogate worth trade-off method and the NIMBUS method (Rangaiah, 2009).

In the previous section, the performance of the SA algorithm that is hybridized with heuristic filling procedure was discussed. This section will focus on the solution of the previously mentioned MOCL problem with two different methods namely Goal programming *(Priori methods)* and Weighted-sum *(Posteriori methods using the scalarization approach).*

For both of the selected approaches, problem data that has been collected from the company mentioned in Section 8.2 is used. The mentioned company distributes Procter and Gamble's products and their own products (paper towels, toilet tissues, paper napkins, etc.), which are manufactured at their own plant. They provided their order lists which include the products (12 different products in the example) to be shipped in a particular day as well as quantities, dimensions (width x depth x height) and weights of the boxes. An order list including all of the required data for the solution of the MOCL problem is supplied in Table 8.6**.** The company uses their own vehicles having a loading capacity of (530x220x210) cm^3 (in width x depth x height) and weight capacity of 7200 kg in order to carry their orders.

Table 8.6. An order list including all of the required data for MOCL problem

Product ID	Product Description	Dimensions (cm) Width/Depth/Height	Number of boxes	Weight of boxes (kg)	Total volume (cm^3)
1	detergent	40 / 36 / 28	325	20	13104000
2	bleaching liquid	54 / 28 / 30	25	22	1134000
3	personal care	54 / 28 / 30	75	2.35	3402000
4	detergent	39 / 29 / 32	75	20	2714400
5	shaving product	15 / 10 / 20	10	1.47	30000
6	baby care	42 / 37 / 25	150	2.8	5827500
7	toothpaste	36 / 18 / 18	3	3.72	34992
8	shampoo	18 / 17 / 22	25	4.97	168300
9	shampoo	22 / 17 / 22	50	5	411400
10	shampoo	12 / 11 / 16	5	1.33	10560
11	bleaching liquid	30 / 27 / 40	20	17.6	648000
12	shaving product	19 / 7 / 21	3	0.12	8379
TOTAL			**766**	**9905.37**	**27493531**

Both models are solved with the proposed SA algorithm where the parameters T_{in} is 5000, α is 0.987, $il_{\max}$ is N (equal to the number of box types in each problem) and T_f is 0.0001.

8.5.2.1. Goal Programming model

Goal programming is a Priori articulation method and has been employed for the solution of numerous types of MOO problems in the literature. Marler and Arora (2004) have also underlined that the most common way of conducting multi-objective optimization is by priori articulation of the decision-makers preferences. This means that before the actual optimization is conducted the different objectives are somehow aggregated to one single figure of merit.

Goal programming (GP) was first introduced by Charnes and Cooper (1961) as a tool to resolve infeasible linear programming (LP) problems. It is one of the most commonly used mathematical programming tools to model multiple-objective optimization problems (Baykasoğlu, 2005). There are two main methods for solving GP models; weighted GP and preemptive GP. In the weighted GP method, the goals are assigned weights and a single-objective function is formulated as the minimization of weighted deviations from the defined goals. In preemptive GP, the goals are grouped according to their importance and more important goals are achieved before less important goals. In this study, the MOCL container loading problem is transformed to a SOO problem using weighted GP method.

For the presented goal programming model the following notations are used;

i	$1,2,\ldots n$; *index for the boxes*
x_i	$\left\{ \begin{array}{ll} 1, & \textit{if the } i^{th} \textit{box is packed} \\ 0, & \textit{otherwise} \end{array} \right\}$
v_i	*Volume of box* i, *where* $v_i = w_i \times d_i \times h_i$
$weight_i$	*Weight of box* i
V_c	*Volume of the container* c, *where* $V_c = W \times D \times H$
u_c	*Volume utilization rate of the container c*
t_weight_t	*Total weight of the packed boxes*

Goal 1: Obtaining a packing pattern having total weight as close to $t\_weight_{\max}$ (weight capacity of the owned vehicle) as possible where d_2^- is the under achievement and d_2^+ is the over achievement of the weight goal.

$$t\_weight_t + d_2^- - d_2^+ = t\_weight_{\max}, \quad \text{where } t\_weight_t = \sum_{i=1}^{n} weight_i x_i \tag{8.2}$$

Goal 2: Maximizing the volume utilization rate u_c of the container *c*, where d_1^- is the under achievement and d_1^+ is the over achievement of the volume utilization goal.

$$u_c + d_1^- - d_1^+ = 1, \qquad \text{where} \qquad u_c = \frac{\sum_{i=1}^{n} v_i x_i}{V_c} \tag{8.3}$$

In order to reach these goals, a weighted GP model is formulated. The objective is to minimize the weighted sum of deviations from the defined goals.

$$\begin{aligned} \min\ & \lambda_1 d_1^- + \lambda_2 d_2^- \\ s.t.\ & u_c + d_1^- - d_1^+ = 1 \\ & t\_weight_t + d_2^- - d_2^+ = t\_weight_{\max} \\ & \lambda_1 + \lambda_2 = 1 \\ & x_i \in (0,1)\ \ where\ i = 1,2,...n \\ & d_1^-, d_1^+, d_2^-, d_2^+ \geq 0 \end{aligned} \tag{8.4}$$

As the defined goals are of different magnitudes, the goals are normalized using the worst and the best possible values of the objectives. In order to solve the proposed weighted GP model, a SA algorithm is designed which is described in the next section. The goals are given weights between 0 to 1 by 0.1 increment/decrements. The trade-offs between these two goals can be seen in Table 8.7.

Table 8.7. Results Obtained for the MOCL Problem

Weights (Goal1, Goal2)	(0.1, 0.9)	(0.2, 0.8)	(0.3, 0.7)	(0.4, 0.6)	(0.5, 0.5)	(0.6, 0.4)	(0.7, 0.3)	(0.8, 0.2)	(0.9, 0.1)
Goal 1 – weight max.	6713,37	6713,37	7157,06	7157,06	7150,41	7150,41	7150,41	7151.37	7151.37
Goal 2 – volume util.	87,51	87,51	86,13	86,13	86,09	86,09	86,09	85,96	85,96
Dev. from Goal 1 (kg)	486,63	486,63	42,94	42,94	49,59	49,59	49,59	48,63	48,63
Dev. from Goal 2 (%)	12,49	12,49	13,87	13,87	13,91	13,91	13,91	14,04	14,04
Weighted sum of deviations	0,192488	0,192488	0,144664	0,144664	0,145988	0,145988	0,145988	0,147154	0,147154

The results offer a set of solutions for the decision maker. In the described situation, the decision-maker seeks a solution which provides a good balance between the defined goals. In this case, the decision maker should choose the packing pattern 86.13% of volume utilization and 7157.06 kg, which yields the smallest weighted sum of deviations from the defined goals. The convergence graph for this solution can be seen in Figure 8.4 and 8.5. If the decision-maker favors this solution, the final view of the packing pattern for this order is shown in Figure 8.6.

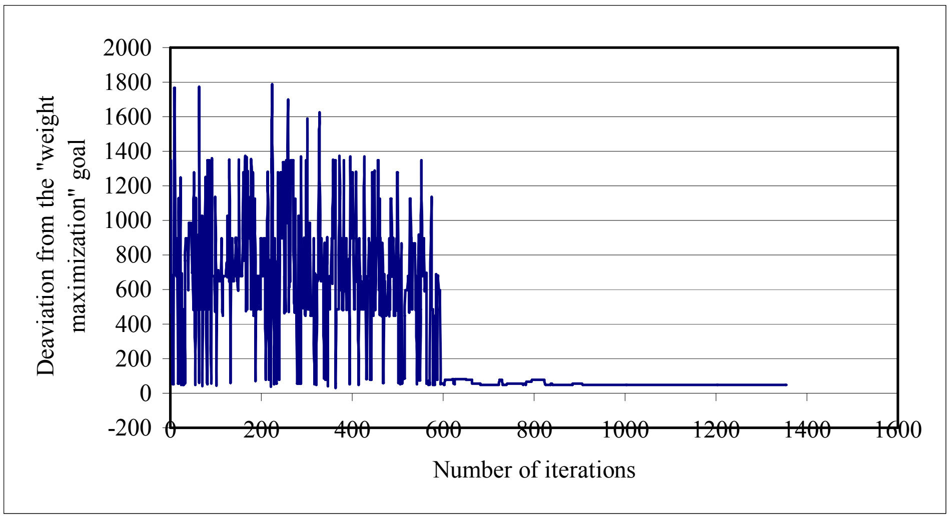


Figure 8.4. Convergence graph for the first goal

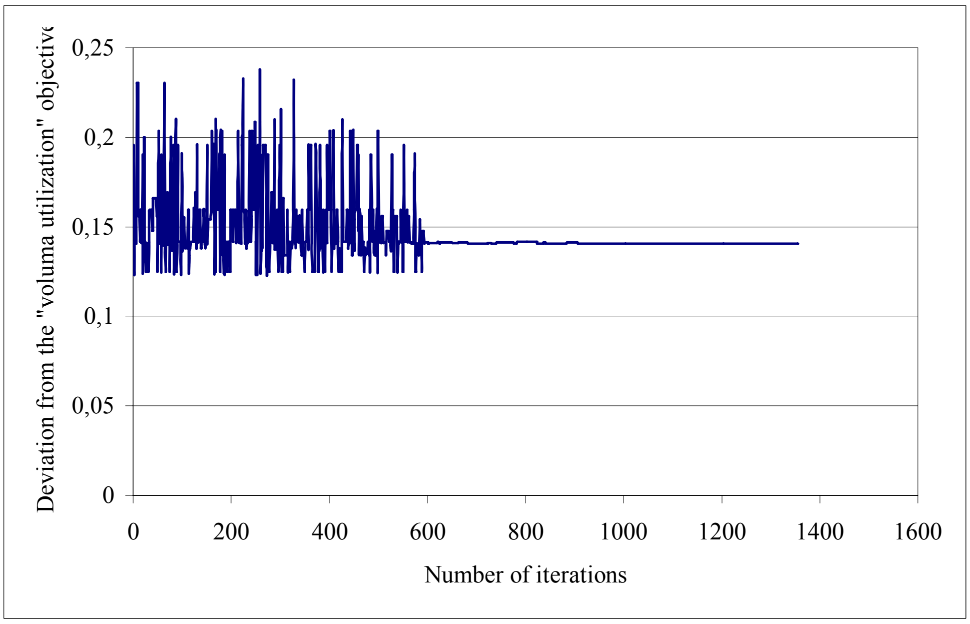


Figure 8.5. Convergence graph for the second goal

If transporting as many goods as possible to obtain an on-time delivery is an important goal for the decision maker, then the packing pattern with the 87.51% of volume utilization can be a good solution. In spite of this, if a very profitable solution is desired, the decision maker can choose the packing pattern with 86.13% of volume utilization and 7157.06 kg of total weight, since this packing pattern has the highest total weight among the other solutions. At this stage, it is decision maker's job to select the best alternative by taking into account the company's transportation policy, profitability and on-time delivery of the shipments.

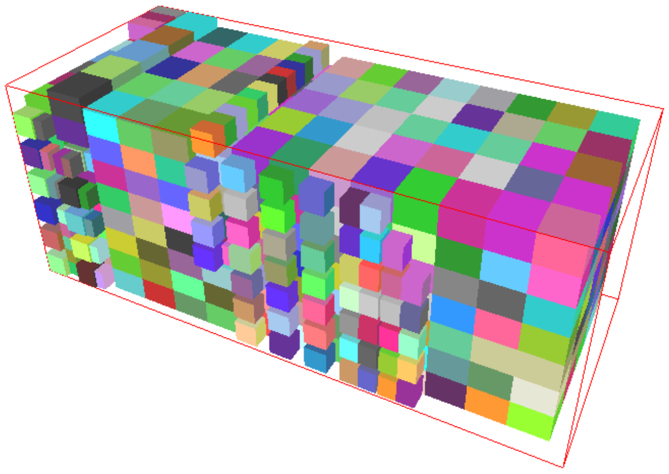


Figure 8.6(a). Filled container – side view

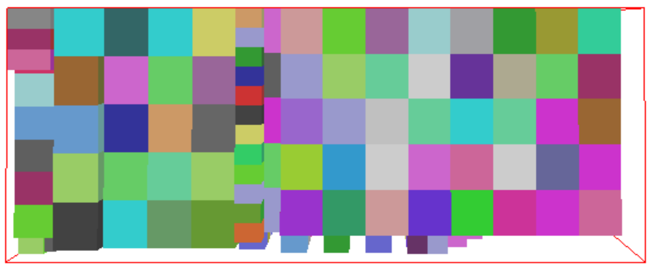


Figure 8.6(b). Filled container – top view

Finally, the problem is also solved by considering each goal individually. This way the differences between the solution obtained by considering one and more objectives can be figured out. The single objective problems are also solved with the same set of SA parameters which are used for the multi-objective problem. In case of the consideration of the goal "maximization of weight" alone, a total weight of 7173.52 kg and a volume utilization of 83.06% is achieved whereas when only the goal "maximization of volume utilization" is considered, a total weight of 6713.37 kg and a volume utilization of 87.51% is achieved. As it is obvious, the proposed solution with 86.13% of volume utilization and 7157.06 kg is a satisfactory and more desirable solution for the company as compared to solutions obtained from the solution of the problem in a single objective manner.

8.5.2.2. Weighted-sum model

One of the most popular approaches of *Posteriori methods* is weighted-sum approach (Bui and Alam, 2008). In the weighted-sum approach, all the objectives are combined into a single objective with the use of a weight vector. In this study, the MOCL problem is solved by obtaining a single objective problem using the weighted-sum method.

Objective 1: Maximize the total weight of the allocated boxes to the container c.

$$\max f_1 = t\_weight_t\,, \quad where\ t\_weight_t = \sum_{i=1}^{n} x_i weight_i \tag{8.5}$$

Objective 2: Maximizing the volume utilization rate u_c of the container *c*.

$$\max\ f_2 = u_c\,, \qquad where \qquad u_c = \frac{\sum_{i=1}^{n} v_i x_i}{V_c} \tag{8.6}$$

In order to reach the set of Pareto optimal solutions, the formulated weighted-sum model is as follows, where $t\_weight_{\max}$ represents the loading capacity of the vehicle in terms of weight;

$$\begin{aligned} &\max f = w_1 f_1 + w_2 f_2 \\ &s.t. \quad t\_weight < t\_weight_{\max} \\ &\qquad w_1 + w_2 = 1 \qquad where\, w_1, w_2 > 0 \\ &\qquad x_i \in (0,1) \qquad where\, i = 1, 2, \ldots n \end{aligned} \tag{8.7}$$

As the defined objectives are of different magnitudes, the objectives are normalized before. The normalization is done by solving maximization and minimization single criterion problems for each of the criteria, discarding the rest of the criteria (Borisova, 2006).

$$\max \left\{ w_1 \frac{t\_weight_t - t\_weight_{t,\min}}{t\_weight_{t,\max} - t\_weight_{t,\min}} + w_2 \frac{u_c - u_{c,\min}}{u_{c,\max} - u_{c,\min}} \right\} \tag{8.8}$$

To solve the proposed weighted-sum model, previously described SA algorithm is used. The objectives are given weights between 0 to 1 by 0.1 increments/ decrements. The trade-offs between these two objectives can be seen in Table 8.8.

Table 8.8. Results obtained for the MOCL problem

Weights (w_1, w_2)	(0.1, 0.9)	(0.2, 0.8)	(0.3, 0.7)	(0.4, 0.6)	(0.5, 0.5)	(0.6, 0.4)	(0.7, 0.3)	(0.8, 0.2)	(0.9, 0.1)
f_1	87.518	87.518	86.09	86.13	86.09	85.96	86.09	85.96	86.09
f_2	6713.37	6713.37	7150.41	7157.6	7150.41	7151.37	7150.41	7151.37	7150.41

The results offer a set of solutions for the decision maker. The Pareto curve for this problem can be seen in Figure 8.7**.** In the described situation, the decision-maker seeks a solution which provides a good balance between the defined objectives. In case, the decision maker wants to pay less money to the rented trucks, he/she can choose the packing pattern 86,13% of volume utilization having a total weight 7157,6 kg, since this packing pattern has the highest total weight among the other solutions.

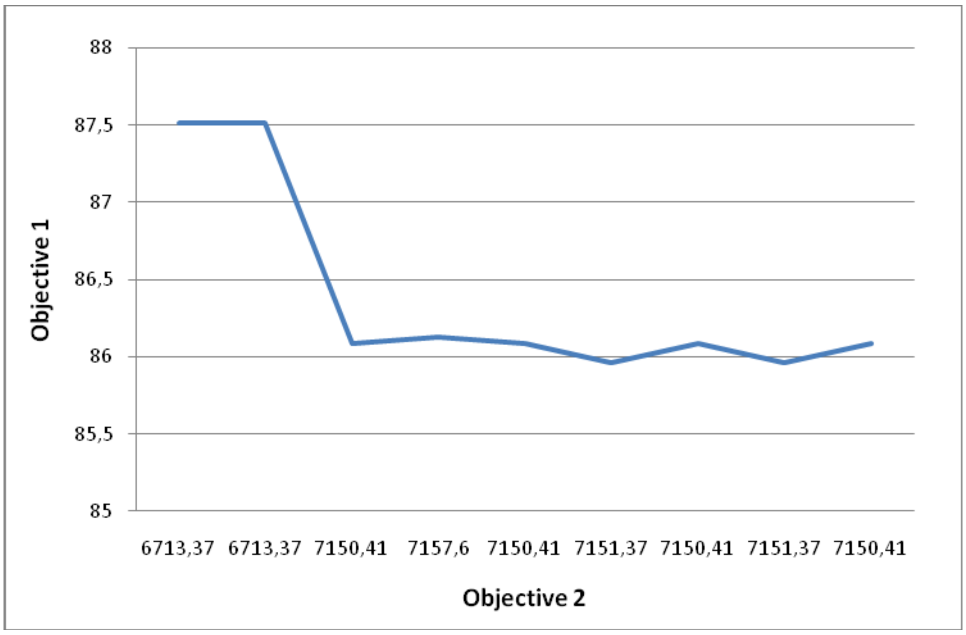


Figure 8.7. The Pareto curve for the solved problem

Both to obtain an on-time delivery and to transport as many goods as possible, the packing pattern with an 87,518% of volume utilization having a total weight of 6713,37 kg. can be a good solution.

In case of the consideration of the objective “maximization of weight” alone, a total weight of 7173.52 kg and a volume utilization of 83.06 % is achieved; whereas when only the objective “maximization of volume utilization” is considered, a total weight of 6713.37 kg and a volume utilization of 87.518% is achieved. The handling of the defined multi-objective problem in a single objective manner revealed that by handling the defined problem in a multi-objective manner it is able to consider objectives simultaneously which results in a compromise between the objectives.

8.6. Conclusion

In this chapter, two different approaches to the solution of MOCL problems that are mostly encountered in transportation and wholesaling industries are explored. The main goal is to load the items (boxes) that would provide the highest total weight to the container in the best possible way. These two objectives “maximization of weight” and “maximization of volume utilization” are conflicting objectives since the volume of a box is usually not proportional to its weight. Using both the Goal Programming and the Weighted-sum approach, the objectives are combined into a single objective. An SA algorithm accompanied by a heuristic filling procedure is then proposed to solve the model. The proposed algorithm has been tested on a set of benchmark problems available in the literature and also on real-world data provided by a distribution company.

CHAPTER 9

CONCLUSIONS

9.1. Present Study

CL problem is a NP-Hard problem which is important for commercial applications in transportation industry. Consequently, approaches in the literature focuses on offering high performance solutions to improve the efficiency of algorithms for these commercial applications. In this thesis, an attempt to propose better algorithms has been done. The research is mainly concentrated on two contributions; one of them is the application of two population based optimization techniques, BA and ACO to the CL problem to search alternative ways for the solution of this NP-Hard problem and the other one is the definition of a new problem called MOCL problem which is frequently encountered in industry.

Chapter 1 presented an introduction to the CL problem and its logistics dimensions. A detailed discussion about Cutting and Packing problems which also embraces the CL problems was presented in Chapter 2. In Chapter 3, a detailed literature survey about the CL problems was discussed taking into account the type of the solution technique proposed so far. In Chapter 4, the proposed heuristic filling procedure that is used for the ACO and BA algorithm and also for the defined MOCL CL problem was introduced. In Chapter 5, the proposed BA algorithm ***hybrid-BA*** had been supplied. Next, algorithms based on ACO algorithm, ***hybrid–ACS-1*** and ***hybrid-ACS-2*** were presented. The computational results for both ACO based algorithms were also supplied within this chapter. The developed decision support system called CLSS was introduced in Chapter 7. Following these chapters, the newly defined MOCL problem and its solution were introduced in Chapter 8. Finally, the study is concluded here, in Chapter 9 with conclusions and recommendations for future work.

9.2. Observations on the Developed SI-based Algorithms and Further Studies

As it is mentioned in the previous chapters, two swarm-based techniques ACO and BA are implemented for the solution of CL problems due to the existing gap in the literature.

Up to date, BA has not been widely used for solving discrete combinatorial optimization problems since they were originally developed for solving continuous optimization problems and their full potential has not been tapped yet. It has been generally used for solving continuous optimization problems like traveling salesman and scheduling as well as used in neural network and data mining applications. Its application to discrete combinatorial optimization problems is less common.

With the purpose of developing a suitable BA algorithm working with the discrete variables, two operators; **1-flip** and **k-flip** are defined and both operators are utilized to solve the CL problem. Using both operators, possible rotation variants of the boxes are reached and the boxes are given priority according to their side dimensions. Then, starting from the highest priority box, these boxes are filled to the container by the proposed heuristic filling procedure which is actually a *"wall building"* approach. The developed algorithm - called ***hybrid-BA-*** is tested on test cases and finally the performance of it is compared with the previous studies from the literature that used the same test cases.

Next, the suitability of the ACO-based algorithm is discussed through two different solution approaches; ***hybrid-ACS-1*** and ***hybrid-ACS-2*** algorithms. In the first approach (***hybrid-ACS-1***), with the use of ACO algorithm, a sequence for all of the boxes in the problem - showing which box should be packed to the container first - is determined. Then having this sequence, boxes are loaded into the container by the proposed heuristic filling procedure that is also used for BA. Tests on LN test cases revealed that the algorithm has a low performance.

The second approach (***hybrid-ACS-2***) is proposed with the intention to improve the performance of the first approach. This improved algorithm is inspired from the idea that the volume utilization of a container greatly depends on the volume utilization of each layer in "wall building" algorithms. For this purpose, first the container is filled *layer-by-layer* by the heuristic filling procedure. Afterwards, a utilization-threshold-level (UTL) is determined experimentally and layers are evaluated according to this value. If a layer having a volume utilization that is above the UTL is found, this layer is saved as it is. Otherwise, the boxes in the layers having a volume utilization that is below the UTL are added to the set of available box list. Thus, the numbers of boxes in each problem is reduced and layers with high volume utilization are saved. Following this step, the set of available boxes and the dimensions of the containers are updated. Then, the ACO algorithm is applied to the reduced problem and the overall volume utilization of the container is calculated. Tests on test cases have shown that the later algorithm ***hybrid-ACS-2*** performs better compared to the ***hybrid-ACS-1.***

In Table 9.1 and 9.2, the results obtained by different approaches offered in this study are summarized. Although the SA algorithm (named hybrid-SA in the mentioned tables) presented in Chapter 8 is proposed for the MOCL problem, test cases specific for the CL problem is solved in order to demonstrate the performance of the algorithm.

Table 9.1. Results obtained with the offered algorithm for the test cases of LN

Problem	***hybrid-BA***	***hybrid-ACS-1***	***hybrid-ACS-2***	***hybrid-SA***
LN01	62.5	62,5	62,5	62.5
LN02	86.3	84,3	80,8	90.1
LN03	53.4	53,4	53,4	53.4
LN04	55.0	55,0	55,0	55.0
LN05	77.2	77,2	77,2	77.2
LN06	89.2	82,5	85,2	85.8
LN07	83.2	82,9	84,0	84.2
LN08	59.4	59,4	59,4	59.4
LN09	61.9	61,9	61,9	61.9
LN10	67.3	67,3	67,3	67.3
LN11	62.2	62,2	62,2	62.2
LN12	78.5	74,8	77,3	78.1
LN13	83.6	81,6	81,6	83.9
LN14	62.8	62,8	62,8	62.8
LN15	59.5	59,5	59,5	59.5
Mean	**69.46**	**68,5**	**68, 7**	**69.6**

Table 9.2. Results obtained with the offered algorithm for the test cases of BR

Problem (box type)	***hybrid-BA***	***hybrid-ACS-2***	***hybrid-SA***
BR1(3)	83.41	77.75	86.38
BR2(5)	84.60	79.41	87.70
BR3(8)	85.42	80.41	87.06
BR4(10)	85.19	80.40	86.61
BR5(12)	85.11	79.94	86.10
BR6(15)	84.69	79.87	85.47
BR7(20)	83.99	79.23	84.49
Mean	**84.63**	**79.57**	**86.26**

The performance evaluation of the proposed SI based algorithms (***hybrid-ACS-1, hybrid-ACS-1*** and ***hybrid-BA)*** revealed that ***hybrid-BA*** is the best performing algorithm and ***hybrid-ACS-1*** is the worst performing algorithm. When the SA approach proposed for the MOCL problem is also considered, it is obvious that ***hybrid-SA*** is the best performing algorithm of all.

There could be several reasons behind the poor performance of ***hybrid-ACS-1*** algorithm. One of them could be the mismatch between the used neighborhood search structure and the nature of the algorithm. A different neighborhood search structure better suited to the nature of the ACO algorithm which mostly performs superior for graph-like problems can improve the performance of the proposed ACO-based algorithms. It is known that pure ACO usually has good globe search ability but poor local search ability like most evolutionary algorithms do. Local search process is often performed to explore the neighborhood of a generated solution for better ones (Luo et al., 2008). Therefore, addition of a local search to the ACO-based algorithms may improve their performance. Another alternative approach to improve these algorithms could be the use of specifically designed operators. Both the mentioned approaches are in the scope of future work.

Mack et al. (2004) expressed that the solution quality of a meta-heuristic for CL problem depends mainly on the "kernel heuristic" of an algorithm. Here "kernel algorithm" (also called decoder algorithm) refers to the algorithm that is used for the selection and placement of the given items. On the other hand, the used meta-heuristic strategy is certainly another important factor, but its influence on the overall solution quality is limited (Mack et al., 2004).

It should also be noted that the proposed heuristic filling algorithm used throughout this thesis is a kind of *"wall-building"* procedure as mentioned previously. It loads the container *layer-by-layer* recursively in a 3D manner. As a result of this packing process, the container is filled with the isolated vertical layers where spanning of the boxes between layers is avoided. This way, the objective function value of a solution is computed. Another issue for further studies could be the improvement of this procedure with the addition of some specialized features. This could greatly improve the performance of the algorithm in which the mentioned procedure is used.

According to Mack et al. (2004) type of load is very important when solving CL problems. For weakly heterogeneous box types, identical item dimension is a good opportunity. In this case, building an arrangement of identical items is beneficial. For strongly heterogenous box types, vertical layers produce promising results. Thus, an opportunity for improvement could be developing a procedure based on different filling approaches for better volume utilization.

9.3. Observations on the Proposed MOCL Problem and Further Studies

Having been inspired from a real world case, a new problem called MOCL problem was defined in Chapter 8. This problem is encountered in a medium-sized distribution company in Gaziantep (a metropolitan province located at the south-east of Turkey) that delivers the goods ordered by supermarkets. The main goal of this multi-objective problem is to load the items (boxes) that would provide the highest total weight to the container in the best possible way. These two objectives "maximization of weight" and "maximization of volume utilization" are conflicting since the volume of a box is usually not proportional to its weight. The proposed solution approach to solve the MOCL container loading problem is an SA algorithm based on *goal programming* or *weighted-sum approach* which is hybridized with a heuristic filling procedure.

Performance of the proposed SA algorithm is initially compared for the well-known test cases from the literature with single-objective function. This analysis has revealed that the proposed algorithm is quite effective in solving the CL problems. The algorithm is then tested on a real case (which is really large as compared to the problems in the literature) including 766 items/boxes. The suitability of the algorithm for the multi-objective case is also checked and the results show that the proposed algorithm can produce a set of Pareto optimal solutions offering some trade-offs between the two objectives. Supported with this knowledge, it is decision maker's job to evaluate the set of solutions and choose a desired solution according to their particular application. At this point, what the decision maker should do is to select the best solution by taking into account the company's transportation policy, profitability and on-time delivery of the shipments.

It is also well worth pointing out that the development and presentation of a best performing algorithm was not the main objective of our work. There are already several high performing algorithms available in the literature. We proposed a simple but quite effective SA-based algorithm in order to solve MOCL problems described in this work. The consideration of the multi-objectives is the main feature of our work. The proposed filling heuristic is designed by taking the neighborhood structure described for the CL problem into account. Reasonable results which are comparable with to those produced by other algorithms in the literature have been produced. The idea underlying this comparison was to discuss the suitability of our approach for the solution of the MOCL problem described in this study.

Stability is an important aspect to consider in the container loading problems (Moura and Oliveira, 2005). It prevents cargo from being damaged during transportation. However, stability of packing is in general not considered by *wall-building heuristics* (Kocjan and Holmström, 2006). The case study presented in Chapter 8 focused on a medium-sized company which distributes the goods ordered by supermarkets such as paper towels, toilet tissues, paper napkins. Thus the company rarely faces with the problem of stability. The use of *'spacing materials'* is considered, if any problem related to stability will occur. Since our heuristic filling procedure uses no *amalgamation of unused spaces* in the filling process while exploiting a wall-building approach and it is not strictly required by the company, the *'stability constraint'* is not considered in this work. This issue is also considered in the list of assumptions in the *'problem formulation'* presented in Chapter 8.

Having a multi-objective problem at hand, a question that arises is which method is the best to solve such a problem given the variety of methods for conducting MOO. Unfortunately, there is no distinct answer. Thus, two different approaches from two different set of methods namely Goal programming *(Priori methods)* and Weighted-sum *(Posteriori methods using the scalarization approach)* have been employed for the solution. Further studies could concentrate on the solution of this problem with different MOO methods.

A similar question about the use of SA could also arise. SA is one of the most powerful and robust which are more likely to find a global optimum and not be stuck on local optima as gradient methods might do. It is also slightly less computational expensive as compared to GAs (Andersson, 2000). In this study, SA is preferred to the other non-derivative optimization methods because of its outstanding and inherent properties as described above.

Finally, the proposed algorithm can further be extended with the employment of different set of objectives and constraints in case of different situations. For example, new objectives related to the environmental factors can be added to the proposed model. One of these objectives could be the "minimization of fuel consumption". It is known that there is a tradeoff between the weight of a shipment and the fuel consumption of a vehicle. Taking this into account, a new model including this new objective could lead to a reduced shipment cost. Due to the lack of exact figures about this tradeoff, such an objective has not been included in the present model. Embedding the methodology to a decision making framework could be also evaluated in the context of further studies.

REFERENCES

Abbass, H.A. (2001). MBO: Marriage in Honey Bees Optimization A Haplometrosis Polygynous Swarming Approach. *Proceedings of the Congress on Evolutionary Computation (CEC2001)*, Seoul, Korea, 207-214.

Abraham, A., Das, S., Ro, S. (2008). *Swarm Intelligence Algorithms for Data Clustering* in *Soft Computing for Knowledge Discovery and Data Mining.* Springer, 279 -313.

Afshar, A., Haddad O. B., Marino, M.A., Adams, B.J. (2008). Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. *Journal of the Franklin Institute*, doi:10.1016/j.jfranklin.2006.06.001.

Agerschou, H., Lundgren, T., Sørensen, T., Ernst, J., Korsgaard, L.R., Schmidt and Chi, W.K. (1983). *Planning and Design of Ports and Marine Terminals*. John Wiley and Sons, Chichester.

Andersson J., (2000). *A survey of multiobjective optimization in engineering design*. Department of Mechanical Engineering, Linköping University; 581 83 Linköping. Sweden. *Technical Report*: LiTH-IKP-R-1097.

Baykasoğlu A., (2005). Preemptive goal programming using simulated annealing. *Engineering Optimization*, **37,** 49-63.

Baykasoğlu, A., Özbakır, L., Tapkan, P. (2007). Swarm Intelligence: Focus on Ant and Particle Swarm Optimization. In Felix T. S. Chan & Manoj Kumar Tiwari (Eds), *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*, 113-144.

Benatchba, K., Admane, L., Koudil, M. (2005) ‘Using bees to solve a data mining problem expressed as a max-sat one’, Proceedings of IWINAC'2005, International Work Conference on the Interplay between Natural and Artificial Computation, Canary Islands, Spain, pp.212-220.

Beni, G. (1988) The concept of cellular robotic systems, *Proceedings of the IEEE International Symposium on Intelligent Control*, pp.57-62, IEEE Computer Society Press.

Bischoff E.E., Marriott M.D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research,* **44,** 267-276.

Bischoff E.E., (2003). Dealing with load bearing strength considerations in container loading problems. *Technical Report*, European Business Management School. University of Wales, Swansea,

Bischoff, E.E., Ratcliff, M.S.W. (1995). Issues in the development of approaches to container loading. *Omega. Int. J. Mgmt Sci.,* **23/4**, 337-390.

Bischoff, E.E., Ratcliff, M.S.W. (1995). Loading multiple pallets. *Journal of Operational Research Society*, **46/11**, 1322-1336.

Bischoff, E.E., Janetz, F., Ratcliff, M.S.W. (1995). Loading pallets with non-identical items. *European Journal of Operational Research*, **84**, 681-692.

Bonabeau, E., Dorigo, M., Theraulaz, G. (1999). *Swarm Intelligence: from natural to artificial systems*. Qxford University Press.

Borissova, D., (2006). Multicriteria Choice of the NVG Optoelectronic Channel Elements. *Problems of Engineering Cybernetics and Robotics*, **56**, 61-68.

Bortfeldt, A., Gehring, H. (1998). Ein Tabu Search - Verfahren für Containerbeladeprobleme mit schwach heterogenem Kistenvorrat. *OR Spektrum*, **20**, 237-250.

Bortfeldt, A, Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, **131**, 143-161.

Bortfeldt, A., Gehring, H., Mack, D. (2003). A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing,* **29**, 641-662.

Bowersox D.J., Closs D. J., Cooper M.B., *Supply Chain Logistics Management,* McGrawHill, 2002.

Bozorg, H. O., Afshar, A, (2004). MBO Algorithm, A New Heuristic Approach in Hydrosystems Design and Operation. *1st International Conference on Managing Rivers in the 21st Century*, 499-504.

Bui, L.T., Alam, S. (2008). *Multi-objective optimization in computational intelligence - Theory and practice*, IGI Global.

A. Charnes A., W.W. Cooper (1961). *Management Models and Industrial Applications of Linear Programming*, Wiley. New York.

Chen, C.S., Lee, S.M., Shen, Q.S., (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, **80**, 68- 76.

Chan, Felix T. S., Bhagwat, R., Kumar, N., Tiwari, M. K., Lam, P., (2006). Development of a decision support system for air-cargo pallets loading problem: A case study. *Expert Systems with Applications*, Volume **31**, Issue 3, 472-485.

Chien, C.F., Deng, J.F., (2002). A container packing support system for determining and visualizing container packing patterns. *Decision Support Systems*, **1**, 12.

Chien, C., Wu, W., (1998). A recursive computational procedure for container loading. *Computers and Industrial Engineering*, **35**.

Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.L. (2006). A bee colony optimization algorithm to job shop scheduling. Proceedings of the 2006 Winter Simulation Conference, pp.1954 – 1961.

Cordon, O., Herrera, F., Stützle, T. (2002). A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and Trends, *Mathware & Soft Computing, 9.*

Daş, G.S., Dereli, T. (2007). Container Loading using hybrid Bees Algorithm. *8th Workshop of the EURO Working Group "EU/ME, the European Chapter on Metaheuristics",* October 4-5, Stuttgart, Germany, 52-59.

de Castro, L.N. (2002). Immune, swarm, and evolutionary algorithms, Part-I: basic models. *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'O2),* 3, 1464-1468.

Dereli, T., Daş, G. S. (2007). A hybrid simulated annealing algorithm for two-dimensional strip packing problems. *Adaptive and Natural Computing Algorithms, Part 1*, **4431,** 508-516.

Dereli, T., Daş, G.S. (2010a). Development A Decision Support System for Solving Container Loading Problems, *Transport,* (accepted).

Dereli, T., Daş, G.S. (2010b). A Hybrid Simulated Annealing Algorithm for Solving Multi-Objective Container Loading Problems, *Applied Artificial Intelligence,* (accepted).

Dereli, T., Daş, G.S. (2010c). Konteyner Yükleme Problemleri için Karınca Kolonisi Optimizasyonu Yaklaşımı, *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, (accepted).

Dereli, T., Seçkiner, S.U., Daş, G.S., Göçken, H., Aydın, M.E. (2009). An exploration of the literature on the use of 'swarm intelligence-based techniques' for public service problems. *European Journal of Industrial Engineering*, **3/4**, 379-423.

Dorigo, M., Maniezzo, V., Colorni, A. (1991). Positive feedback as a search strategy (Tech. Rep. 91-016). Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica.

Dorigo, M., Di Caro, G., Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, **5/2**, 137-172.

Dorigo, M., Gambardella, L.M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary* Computation, **1**, 1, 53-66.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research,* **44/2**, 145-159.

Dyer, F.C. (2002). The biology of the dance language. *Annual Review of Entomology,* **47**, 917-949.

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

Eglese R.W. (1990). Simulated Annealing: A tool for Operational Research. *European Journal of Operational Research*, **46,** 271-281.

Engelbrecht, A.P. (2005). *Fundamentals of Computational Swarm Intelligence*, Wiley.

Ertek, G., Kılıç, K. (2006). Decision support for packing in warehouses. *Lecture Notes in Computer Science,* **4263,** 115-124.

Faina, L. (2000). A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research*, **126,** 340-354.

Fathian, M., Amiri, B., Maroosi, A. (2007). Application of Honey-Bee Mating Optimization Algorithm on Clustering. *Applied Mathematics and Computation*, **190/2**, 1502-1513.

Gehring, H., Bortfeldt, A. (1997). A genetic algorithm for solving the container loading problem. International Transactions in Operational Research, 4, 401-418.

Gehring, H., Bortfeldt, A. (2002). A parallel genetic algorithm for solving the container loading problem. *International Transactions on Operational Research*, **9/4**, 497–511.

Gehring, H., Menschner, K., Meyer, M. A. (1990). Computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, **44,** 277-288.

George, J.A., Robinson, D.F. (1980). A heuristic for packing boxes into a container. *Computers and Operations Research*, **7,** 147-156.

Gravel, M., Price, W.L., Gagne, C., (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research,* **143,** 218–229.

Haessler, R.W., Talbot, F.B. (1990). Load planning for shipments of low density products. *European Journal of Operational Research*, **44,** 289-299.

Harrison, A., van Hoek R., (2002). *Logistics Management and Strategy*, Prentice Hall.

He, K., Huang, W., (2009). Solving the single container loading problem by a fast heuristic method. *Optimization Methods and Software*, 1-15.

Hemminki, U. (1993). A heuristic for container loading. *Report* 141, University of Turku, Institute for Applied Mathematics, Hopper, E., Turton, B.C.H., (1997). A genetic algorithm for a 2D industrial packing problem. *Computers and Industrial Engineering*, **37**, 375-378.

Huang, W., He, K. (2009). A caving degree approach for the single container loading problem. *European Journal of Operational Research,* **196,** 93–101.

Hopper, E., Turton, B.C.H., (2001). An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, **128**, 34-57.

Kang, M.K., Jang, C.S., Yoon, K.S. (2010). Heuristics with a new block strategy for the single and multiple containers loading problems. Journal of the Operational Research Society, **61**, 95-107.

Karaboğa, D., Baştürk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, **39/3**, 459–471.

Kennedy, J., Eberhart, R.C. (1995). Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks (Perth, Australia) , IEEE Service Center, Piscataway, NJ, IV, pp. 1942-1948.

Kennedy, J., Eberhart, R.C., Shi, Y. (2001). *Swarm Intelligence*, Morgan Kaufmann.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, **220,** 671-680.

Kocjan, W., Holmström, K. (2006). The AUTOPACK Project, Algorithms for container loading. *Research Report*, MdH/IMa, 2006-03, Department of Mathematics and Physics, Mälardalen University, ISSN 1404-4978, SE-721 23 Västerås, Sweden.

Kong, M., Tian., P., Kao, Y., (2008). A new ant colony optimization algorithm for the multidimensional knapsack problem. *Computers and Operations Research*, **35/8,** 2672-2683.

Koudil, M., Benatchba, K., Tarabet, A., Sahraoui, E.B. (2007). Using Artificial Bees to Solve Partitioning and Scheduling Problems in Codesign. *Applied Mathematics and Computation*, **186/2**, 1710-1722.

Levine, J., Ducatelle, F. (2004). Ant colony optimisation for bin packing and cutting stock problems. *Journal of Operational Research Society*, **55**, 705-716.

Li, H-L., Tsai, J-F, Hu, N-Z, (2003). A distributed global optimization method for packing problems. *Journal of Operational Research Society*, 419-425.

Liang S.C., Lee, C.Y., Huang S.W., (2007). Hybrid Meta-Heuristic for the Container Loading Problem. *Communications of the IIMA*, 7/4, 73-84,

Lim, Rodrigues, B., Yang, Y. (2005). 3-D Container packing heuristic. *Applied Intelligence*, **22**, 125-134.

Liu, D.S., Tan, K.C., Goh, C.K., Ho, W.K. (2006). On solving multiobjective bin packing problems using particle swarm optimization. *IEEE Congress on Evolutionary Computation,* 2095-2102.

Lodi A., (2002), "Multi Dimensional packing by tabu search". Technical Report.

Loh, H. T., Nee, A. Y. C. (1992). A packing algorithm for hexahedral boxes. *Proceedings of the Industrial Automation Conference Singapore*, 2, 115-126.

Luo, D., Wu, S., Li, M., Yang, Z. (2008). Ant Colony Optimization with Local Search Applied to the Flexible Job Shop Scheduling Problems. *IEEE*, 1015 - 1020.

Lucic, P. (2002). *Modeling Transportation Problems Using Concepts of Swarm Intelligence and Soft Computing*, PhD Thesis, Civil Engineering, Faculty of the Virginia Polytechnic Institute and State University.

Mack, D., Bortfeldt, A., Gehring, H., (2004). A parallel hybrid local search algorithm for the container loading problem. *International Transactions in Operational Research*, **11,** 511-533.

Marler, R.T., Arora, J.S. (2004). Review of multi-objective optimization concepts and algorithms for engineering. *Technical Report*, No: ODL-01.04. Optimal design laboratory, College of Engineering, The University of Iowa, Iowa City.

Martello S., Pisinger D., Vigo D. (2000). The three-dimensional bin packing problem, *Operations Research*, **48**, 256-267.

Montgomery, D.C. (1991). *Design and analysis of experiments.* John Wiley & Sons, New York.

Morabito, R.N., Arenales, M.N. (1994). An and-or graph approach to the container loading problem. *International Transactions in Operational Research*, **1,** 59-73.

Moura, A., Oliveira J. (2005). A grasp approach to the container-loading problem. *IEEE Intelligent Systems*, 50-57.

Nakrani, S., Tovey, C. (2007). From honeybees to Internet servers: biomimicry for distributed management of Internet hosting centers. *Bioinspiration and Biomimetics*, **2**, 182-197.

Nepomuceno, N., Pinheiro, P., Coelho, A.L.V., (2007). Tackling the container loading problem: A hybrid approach based on Integer Linear Programming and Genetic Algorithms. *EvoCOP 2007, LNCS 4446*, pp. 154 – 165.

Ngoi, B.K.A., Tay, M.L., Chua, E.S., (1994). Applying spatial representation techniques to the container packing problem. *International Journal of Production Research*, **32/1**, 111-123.

Parreno, F., Alvarez-Valdes, R., Oliveira, J.F., Tamarit, J.M. (2010). Neighborhood structures for the container loading problem : a VNS implementation. *Journal of Heuristics*, **16**, 1-22.

Perretto, M., Lopes, H.S. (2005). Reconstruction of phylogenetic trees using the ant colony optimization paradigm, *Genetics and Molecular Research,* **4/3**, 581-589.

Pham, D.T., Koc, E., Ghanbarzadeh, A., Otri, S., Rahim, S., Zaidi, M. (2006a) The bees algorithm - a novel tool for complex optimization problems, Proceedings of the 2nd Int. Virtual Conference on Intelligent Production Machines and Systems (IPROMS'2006), Oxford, Elsevier.

Pham, D.T., Otri, S., Ghanbarzadeh, A., Koc, E. (2006b). Application of the bees algorithm to the training of learning vector quantization networks for control chart pattern recognition, Proceedings of Int. Conference on Information and Communication Technologies, 24-28 April 2006, Umayyad Palace, Damascus, Syria, pp.1624-1629.

Pham, D.T., Koc, E., Ghanbarzadeh, A., Otri, S. (2006c). Optimization of the weights of multi-layered perceptrons using the bees algorithm, Proceedings of the 5th Int.Symposium on Intelligent Manufacturing Systems, Sakarya, Turkey, 38-46.

Pisinger, D. (2002). Heuristics for the container loading problem. *European Journal of Operational Research*, **141**, 382-392.

Rangaiah, G.P. (2009). Multi-Objective Optimization Techniques and Applications in Chemical Engineering, World Scientific, Singapore.

Rosen, K, H. (2003). Discrete mathematics and its applications. 5th ed.. McGraw Hill.

Seeley, T. D. (1955). *The wisdom of the hive.* Cambridge: Harvard University Press.

Teo , J. Abbass, H.A. (2003). A True Annealing Approach to the Marriage in Honey-Bees Optimization Algorithm. *International Journal of Computational Intelligence and Applications*, **3/2**, 199-211.

Teodorovic, D. (2003). Transport modeling by multi-agent systems: a swarm intelligence approach. *Transportation Planning and Technology*, **26/4**, 289-312.

Teodorovic, D., Lucic, P. (2005). Schedule synchronization in public transit using the fuzzy ant system. *Transportation Planning and Technology*, **28/1**, 47-76.

Terno, J., Scheithauer, G., Sommerweiss, U., Riehme, J. (2000). An efficient approach for multi-pallet loading problem. *European Journal of Operational Research*, **123**, 372-381.

van de Voort,M., O'Brien, K.A., Rahman, A., Valeri, L., (2003). Seacurity: Improving the Security of the Global Sea-Container Shipping System, Rand.

Vis, Iris F.A., de Koster, René, (2003). Transshipment of containers at a container terminal: An overview. *European Journal of Operational Research*, **147/1**, 1-16.

von Frisch, K. (1967). The Dance Language and Orientation of Bees. Cambridge MA: Harvard University Press.

Wang, Z., Li, K.W., Levy, J.K. (2008). A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. *European Journal of Operational Research,* **191**, 86–99.

Wäscher, G., Haussner, H., Schumann, H. (2007). An improved typology of cutting and packing problems. European Journal of Operational Research. 183, 1109-1130.

Wood, D.F., Barone, A. P., Murphy P.R., Wardlow D.L., (2002). *International Logistics*, AMACOM.

Yang, X.S. (2005). Engineering optimizations via nature-inspired virtual bee algorithms. IWINAC 2005, LNCS 3562, Yang, J. M. and J.R. Alvarez (Eds.), Springer-Verlag, Berlin Heidelberg, pp.317–323.

Yang, C., Simon, D. (2005). A new particle swarm optimization technique. Proceedings of the 18th Int. Conf. on Systems Engineering (ISCEng'05).

Yeung, L.H.W, Tang, W.K.S. (2005). A hybrid genetic approach for container loading in logistics industry. *IEEE Transactions on Industrial Engineering*, **52,** 617-627.

Zhao, P., Zhao, P., Zhang X. (2006). A new ant colony optimization for the knapsack problem. *7th International Conference on Computer-Aided Industrial Design and Conceptual Design CAIDCD '06,* 2006.

Web1 (www.hlfreght.com/cargo-frameset.html)

Web2 (http://photo.bees.net/biology/ch6/dance2.html)

Web3 (http://www.scholarpedia.org/article/Ant_colony_optimization)

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Gülesin Sena Daş
Nationality: Turkish (TC)
Date and Place of Birth: 2 June 1979, Ankara
Marital Status: Married with two children
Phone: +90 506 608 45 76
Fax:
email: senaemre@yahoo.com, sena.das@tubitak.gov.tr

EDUCATION

Degree	**Institution**	**Year of Graduation**
MS	Gaziantep University, Dept. of Industrial Engineering	2003
BS	Gazi University, Dept. of Industrial Engineering	2001
High School	METU Development Foundation High School, Ankara	1997

WORK EXPERIENCE

Year	**Place**	
2008- Present	TÜBİTAK	Scientific Programmes Assistant Expert
2001- 2008	Gaziantep University, Dept. of Industrial Engineering	Research Assistant

FOREIGN LANGUAGES

English (very good)

PUBLICATIONS

International Journals

Dereli, T., **Daş, G.S.,** Development A Decision Support System For Solving Container Loading Problems, Transport (accepted).

Dereli, T., **Daş, G.S.,** A Hybrid Simulated Annealing Algorithm for Solving Multi-Objective Container Loading Problems, Applied Artificial Intelligence (accepted).

Dereli, T., **Daş, G.S.,** Konteyner Yükleme Problemleri için Karınca Kolonisi Optimizasyonu Yaklaşımı, Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, (kabul edildi).

Dereli, T., **Daş, G.S.,** A hybrid 'bee(s) algorithm' for solving container loading problems, Applied Soft Computing (submitted).

Dereli, T., Seçkiner, S.U., **Daş, G.S.**, Göçken, H., Aydın, M.E., An exploration of the literature on the use of 'swarm intelligence-based techniques' for public service problems, European Journal of Industrial Engineering, **3/4**, 379-423, 2009.

Dereli, T., **Daş, G.S.**, A hybrid Simulated annealing algorithm for two-dimensional strip packing problems, Adaptive and Natural Computing Algorithms, Part 1, 4431, 508-516, 2007.

Baykasoğlu, A., Dereli, T., **Daş, S. E.**, Project Team Selection using Fuzzy Optimization Approach, Cybernetics and Systems, 38, 155 - 185, 2007.

Dereli, T., Baykasoğlu, A., **Daş, S. E.,** Fuzzy Quality Team Formation for Value Added Auditing, Journal of Engineering and Technology Management, 2006.

National Journals

Dereli, T., Baykasoglu, A., **Emre, G. S.**, Sevim, T., Çatışma Yönetimi, Kalder Forum, 4(14), 31-39, 2004.

International and National Conference Papers

Dereli, T., Baykasoglu, A., **Emre, S.**, Tanış, S., Varlık, G., Bir Çeviklik Aracı Olarak: Yeni Ürün Geliştirme Sürecinin Planlanması, XXIII Yöneylem Araştırması ve Endüstri Mühendisliği Kongresi, 3-5 Temmuz 2002, İstanbul, Turkey.

Dereli, T., Baykasoglu, A., **Emre, S.**, Tanış , S., Sevim, T., Çeviklik, Tepkisellik ve Esnekliğin Yeni Ürün Geliştirme Sürecine Yansımaları, III. Ulusal Üretim Araştırmaları Sempozyumu Bildiriler Kitabı, Kültür Üniversitesi, İstanbul, 19-20 Nisan 2003, pp. 607-612.

Baykasoglu, A., Dereli, T., **Emre, S.**, A Fuzzy Approach To Project Team Formation Problems, TAINN'2003, International Twelfth Turkish Symposium on Artificial Intelligence and Neural Networks, 02-04, July, 2003, Çanakkale, Turkey, Vol. E7, pp.704-714.

Baykasoglu, A., Dereli, T., **Emre, S.**, Göçken, T., Çok ürünlü bulanık ekonomik parti büyüklüğü probleminin tabu arama ve tavlama benzetimi algoritmaları ile çözülmesi, YA/EM'2004: Yöneylem Araştırması / Endüstri Mühendisliği Kongresi XXIV. Ulusal Kongresi, 15-18 Haziran 2004, Gaziantep, Adana , pp. 157-159.

Baykasoglu, A., Dereli, T., Göçken, T., **Emre, S.,** Çok objektifli üretim planlaması problemlerinin bulanık matematiksel programlama ile çözülmesi, YA/EM'2004:

Yöneylem Araştırması / Endüstri Mühendisliği Kongresi XXIV. Ulusal Kongresi, 15-18 Haziran 2004, Gaziantep, Adana, pp. 500-502.

Dereli, T., Baykasoglu, A., **Emre, S.**, Göçken, T., Kalite denetim takımlarının oluşturulması, YA/EM'2004: Yöneylem Araştırması / Endüstri Mühendisliği Kongresi XXIV. Ulusal Kongresi, 15-18 Haziran 2004, Gaziantep, Adana, pp. 169-171.

Dereli, T., Baykasoglu, A., **Emre, S.**, Göçken, T., Üç boyutlu paketleme problemlerine analitik yaklaşımlar, YA/EM'2004: Yöneylem Araştırması / Endüstri Mühendisliği Kongresi XXIV. Ulusal Kongresi, 15-18 Haziran 2004, Gaziantep, Adana, pp. 493-495.

Dereli, T., **Daş, G. S.**, A hybrid simulated annealing algorithm for 2D packing problems, Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, May 29-31, 2006, Sakarya University, pp. 55.

Dereli, T., **Daş, G. S.**, Üç Boyutlu Konteyner Yükleme Problemlerinin Çözümü İçin Sezgisel Bir Yaklaşım, YA/EM'2006: Yöneylem Araştırması / Endüstri Mühendisliği Kongresi XXVI. Ulusal Kongresi, 3-5 Temmuz 2006, Kocaeli.

Dereli, T., **Daş, G. S.**, Çok amaçlı konteyner yükleme problemi ve bir uygulama, YA/EM'2007: Yöneylem Araştırması / Endüstri Mühendisliği Kongresi XXVII. Ulusal Kongresi,2-4 Temmuz 2007, İzmir, pp. 663- 668.

Dereli, T., **Daş, G.S**., A hybrid Simulated annealing algorithm for two-dimensional strip packing problems, ICANNGA 2007, Warsaw, Poland.

Daş, G. S., Dereli,T., Container Loading using hybrid Bees Algorithm, 8th Workshop of the EURO Working Group " EU/ME, the European Chapter on Metaheuristics", October 4-5, 2007, Stuttgart, Germany, 52-59.

Daş, G. S., Dereli,T., Ant Algorithms for Container Loading Problems, Workshop on Women in Industrial Engineering Academia – the U.S. and Middle East, July 7 – 11 2008, ANKARA and ISTANBUL, TURKEY, Poster.

HOBBIES

Puzzle, Reading and Travelling.