Yard Allocation Problem in Bulk Port Terminals

by

Işıl Koyuncu

A Thesis Submitted to the

Graduate School of Sciences and Engineering

in Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in

Industrial Engineering

Koç University

July 2015

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Işıl Koyuncu

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

____

Prof. Dr. Ceyda Oğuz (Advisor)

____

Prof. Dr. Barış Tan

____

Assoc. Prof. Dr. Kerem Bülbül

Date: ______________________________

ABSTRACT

The bulk port terminals are important links in world trade; therefore the operations offered by the terminals should be efficient and effective. These operations include allocating berths to the upcoming vessels, storing and handling bulk materials on the stockyard, and assigning and scheduling stacker and reclaimers to collect the material from the yard and convey to the vessels. Although yard assignment or storage allocation problems at the yard side of the ports have been studied for many years, most of the studies mainly focus on storing containers, and bulk port terminals receive less attention than they deserve.

In many port terminals, dry bulk materials are stored and shipped on a regular basis. In this thesis, we investigate the yard assignment of dry bulk materials, such as coal, in bulk ports. The ultimate goal of this study is to develop a methodology to determine the location of stockpiles on the stockyard while minimizing the total travel distance and the total dwell time of materials that are handled in ports. Therefore, we present mathematical models which address this yard assignment problem by generalizing the well-known multi-dimensional packing models.

The first two models proposed in this thesis address the yard allocation problem (YAP) in a continuous time and continuous space. Continuous YAP is studied where the length of the stockyard as well as the planning time horizon is continuous. On the other hand, the third model we propose discusses the YAP in discrete time and discrete space.

In our models, we consider two types of stockyards: with pads and without pads. In the absence of pads, the stockpiles can be stored freely provided that they are parallel to the edges of the stockyard. Therefore, in addition to the assignment of materials to the storage locations, the first model also determines the orientation of the bulks given that the dimensions of each bulk are known. We perform computational experiments which indicate that the models can solve small- and medium-sized real life instances. However, since the underlying multi-dimensional packing problems are NP-hard, developing a computationally efficient mathematical model is challenging. Thus, a hybrid metaheuristic (YATS_VNS) which is based on Tabu Search and Variable Neighborhood Search to solve YAP in large-scaled bulk port terminals is presented. Finally, we analyze the performance of both the mathematical models and the metaheuristic algorithm with four sets of generated data including instances based on real life data.

For small-sized instances, mathematical models as well as heuristics can solve YAP to optimality within a reasonable time. For large data sets, we manage to obtain small gaps at least with one of the mathematical models. Moreover, we compare the metaheuristic results with the best obtained solutions. The results indicate that we improve the solution quality and running time with YATS_VNS in almost every instance.

ÖZET

Kuru dökme yük terminalleri dünya ticaretinde önemli bağlantı noktalarını oluşturduğundan terminaller tarafından sunulan operasyonların verimli ve etkili olması gerekmektedir. Bu operasyonlar rıhtımların gelen gemilere atanması, kuru dökme yüklerin elleçlenmesi ve depo alanında depolanması, ve yükleri depo alanından toplayan boşaltıcıların atanması ve çizelgelemesini içermektedir. Yük depolama problemi konteyner terminallerinde çokça çalışılmış olsa da kuru yük terminallerinde yük depolama problemi hak ettiğinden daha az ilgi görmüştür.

Çoğu dökme yük terminalinde, kuru yükler düzenli olarak depolanmakta ve sevk edilmektedir. Bu tezde, dökme yük terminallerinde kuru yükleri depolama problemi (YDP) üzerinde çalışılmıştır. Bu araştırmanın amacı, yüklerin depo alanında aldığı yolu ve elleçlenmek için bekledikleri süreyi enazlayacak depo alanındaki yerleri belirleyen bir yöntembilimi geliştirmektir. Bu amaca ulaşmak için bu tez çalışmasında, YDP'yi çözmek üzere Karma Tam Sayılı Doğrusal Programlama (KTSDP) modelleri ve üstsezgisel algoritmalar sunulmaktadır.

Sunulan ilk iki model YDP'yi sürekli zaman ve aralıkta ele almaktadır. Sürekli YDP depolama alanı uzunluğu ve planlama dönemi sürekli olduğunda çalışılmaktadır. Öte yandan, üçüncü model YDP'yi kesikli zaman ve aralıkta incelemektedir.

Geliştirilen modellerde iki tip depolama alanını ele alınmıştır: şerit alanlardan oluşan depolar ve bölüntüsüz tek alandan oluşan depolar. Şerit depo alanlarının yokluğunda, stok yığını depolama alanı üzerinde alan sınırlarına paralel olacak şekilde serbestçe yerleştirilebilir. Bunun sonucunda, ilk model yığınların depolama yerinin yanı sıra yığınların alandaki yönünü de belirler. Sayısal deneyler, modellerin küçük ve orta ölçekli gerçek hayat örneklerini çözebildiğini göstermektedir. Bunun yanı sıra YAP'ın bir özel hali olan çok boyutlu kutulama problemleri halihazırda NP-zor problemler kategorisinde olduğu için verimli bir KTSDP modeli sunmak zorlu olmaktadır. Bu nedenle büyük ölçekli limanlarda YDP'nin çözümünde kullanılmak üzere Tabu Arama ve Değişken Komşu Arama algoritmalarını birleştiren karma bir üstsezgisel algoritma (YATS_VNS) önerilmiştir. Son olarak, sunulan KTSDP modellerinin ve üstsezgisel algoritmanın performansı gerçek hayat verilerini de içeren dört farklı veri kümesi ile test edilmiştir.

Küçük ölçekli örneklerde, KTSDP modelleri ve üstsezgisel algoritma YDP için eniyi sonucu makul bir sürede elde edebilmektedir. Büyük ölçekli örneklerde ise ikinci KTSDP modeli ile eniyi çözüme yakınsayan sonuçlar elde edilmiştir. Bunlara ek olarak, üstsezgisel algoritmanın çözümleri KTSDP modelinden elde edilen sonuçlar ile karşılaştırılmıştır. Sonuçlara göre, YATS_VNS algoritması ile çözüm kalitesi ve çözüm süresi büyük ölçekli örneklerden oluşan dördüncü veri setindeki neredeyse tüm örneklerde iyileştirilmiştir.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my advisor Prof. Dr. Ceyda Oğuz for her guidance, encouragement and trust throughout my thesis study.

I am also grateful to the rest of my thesis committee Prof. Dr. Barış Tan and Assoc. Prof. Dr. Kerem Bülbül for their insightful comments and valuable suggestions.

I would like to acknowledge the financial support of TUBITAK and facilities of Koç University during my M.Sc. study.

I would like to extend my special thanks to Özgür Ünsal and Fatih Rahim, who were more than good friends, were always willing to support me with their helpful suggestions. I also would like to thank my flatmates Yasemin Vardar and Serena Muratcıoğlu for their support, love and best wishes. They brought color to my life and we shared joyful moments together.

I would like to thank "a mysterious stranger who turns up in the midst of the night". He was always there with love, cheering me up and stood by me through the good and bad times. With a great patience, he corrected my writings and provided a continuous support even from a great distance.

Last but not the least, I would like to thank my parents for supporting me spiritually throughout my graduate study and my life. If they had not been there for me, not a single line of this thesis would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

According to United Nations Conference on Trade and Development (UNACTAD) 2014 Report, world merchandise trade volumes expanded in 2013 with a rate of 2.2% (UNACTAD, 2014). Consequently, growth in world maritime transportation is announced as 3.8% which corresponds to nearly 9.6 billion tons. Maritime transportation is mostly based on containerized and dry bulk materials. They comprise the largest share, approximately 70%, of all cargos. On the other hand the remaining 30% of shipments includes wet bulk materials such as petroleum and crude oil. Containerized materials are transported using boxed-shaped equipment with mostly standard sizes, whereas due to uniformity of bulk materials, they are stored in a single hold and shipped in large quantities without containers. Additionally, as industries, such as manufacturing and construction, flourish, the demand for energy and raw materials increase. Big developing countries have a tendency to economize energy resources efficiently; therefore they switch to coal powered factories. These recent trends, especially the increases in major dry-cargo trade volumes, such as coal, affect the maritime transportation volumes as well. In parallel with these increments, the trade of bulk commodities grew by 5.5% (UNACTAD, 2014). The bulk port terminals are important links in world trade; therefore the operations offered by the terminals should be efficient and effective. These terminals establish the backbone of the maritime logistics and supply chain. They provide transportation and handling operations and assure an integrated management between seaside operations and landside operations to handle inbound and outbound materials that traverse the bulk port terminals.

By applying a correct planning with a high efficiency, the throughput of bulk port terminals can be increased. Figure 1 shows some of the important operations of bulk port terminals.

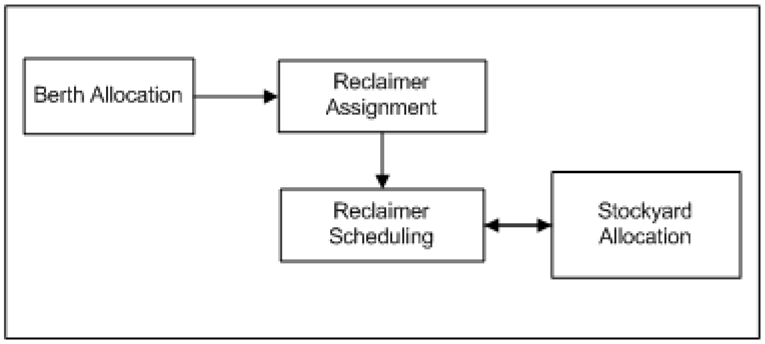

Figure 1. Operations of bulk port terminals

These operations include allocating berths to the upcoming vessels, storing and handling bulk materials on the yard, and assigning and scheduling stacker and reclaimers to collect the material from the yard and send to the vessels.

Although berth allocation problem (BAP) and stacker-reclaimer scheduling problem are highly studied, yard allocation problem in bulk port terminals receive less attention. To the best of our knowledge, a few studies are conducted in an integrated setting with BAP (Robenek *et al.* (2014) and Boland *et al.* (2012)). Therefore, the motivation of this research is to focus on yard management of bulk port materials and provide exact solutions as well as fast and good quality solutions to this problem.

In this thesis, stockyard allocation problem for bulk materials is studied. Stockyard is an enclosed area where bulk materials are stored without any containers. In import terminals, the inbound materials are unloaded form vessels and stocked until trains or trucks arrive to transfer the material. Similarly, in export terminals, outbound materials are brought to the yard by trains or trucks and stored until the relevant vessel comes. Since the storage principles of import and export terminals are similar, in this thesis only export terminals are considered.

1.1 Bulk Port Terminals

Nowadays, 90% of import and export merchandise is transported by sea (UNACTAD, 2014). Maritime transportation is less costly and safer when compared to other means of transportation. As world merchandise trade volumes expanded, intercontinental shipment volumes are increased. Thus, the effective and efficient management of port terminals is becomes more important.

Figure 2 shows an overview of a bulk port terminal. The bulk materials are mostly consisting of raw materials such as coal, iron-ore, petroleum or other underground resources that are used in different industries. The bulk materials are arriving to the port by trains or trucks depending on the volume and the origin of the material. These materials are unloaded to the dump yard and then transported to the stockyard with conveyor belts. Stackers load the material and built stockpiles on the yard. These materials are collected from the yard with reclaimers. Collected materials are loaded to the incoming vessels which are berthing at the quayside of the port. Ship loaders serve these vessels and when all the material is loaded to the vessel, it departs from the port.

Since bulk materials are imported and exported all around the world through these terminals, it is very important to serve the arriving vessels promptly and properly. Unlike container terminals where containers arrive from different places and after cross-docking they are sent to the destinations, in most of the bulk port terminals, the materials sent to the destination points directly. Therefore, the most crucial factor that affects the transportation time is the time that vessels spent in the bulk port terminal. Thus, any effort that increases the efficiency and effectiveness of a bulk port terminal is worthwhile.

Boland *et al.* (2012) emphasize the different characterizations of bulk terminals in terms of storage policies. While some terminals perform as a cargo assembly terminal, others have dedicated stockyards. Cargo assembly terminals adopt a “pull-based” strategy. They order materials based on the demand. For example, once a vessel is nominated then the demanded material is brought to the stockyard. On the other hand, in a dedicated stockpiling terminal, a “push-based” strategy is adopted. There are predetermined places for different bulk material types and materials are prepared to be loaded beforehand. In this thesis, we focus on terminals which operate in a “pull-based” manner.

Bulk materials are stored in huge volumes and once a stockpile is started to build, usually it is not moved to another place. Since the movement operation is costly and time consuming, it is crucial to decide the right places for each stockpile.

The following sections provide related definitions about bulk port terminals. The common types of bulk materials, their storage category and other structures and equipment that are used in the terminals are discussed.

1.1.1 Bulk Materials

Bulk materials are materials which are stored in a single hold and shipped in large quantities without containers. Bulk materials can be divided into two categories, such as dry bulk materials and wet bulk materials. Dry bulk materials, such as iron ore, coal, minerals are stored without any containers. These materials are piled at the stockyards of port terminals or mines. Due to the uniformity of the materials, they are ordered and shipped mostly in large volumes. Since they are shipped in large quantities, bulk materials are commonly transported by intercontinental vessels or freight trains. On the other hand, wet bulk materials, such as petroleum and gasoline are stored in a single hold and in large volumes. Additionally, these liquid products can be transported by intercontinental tankers as well as pipelines.

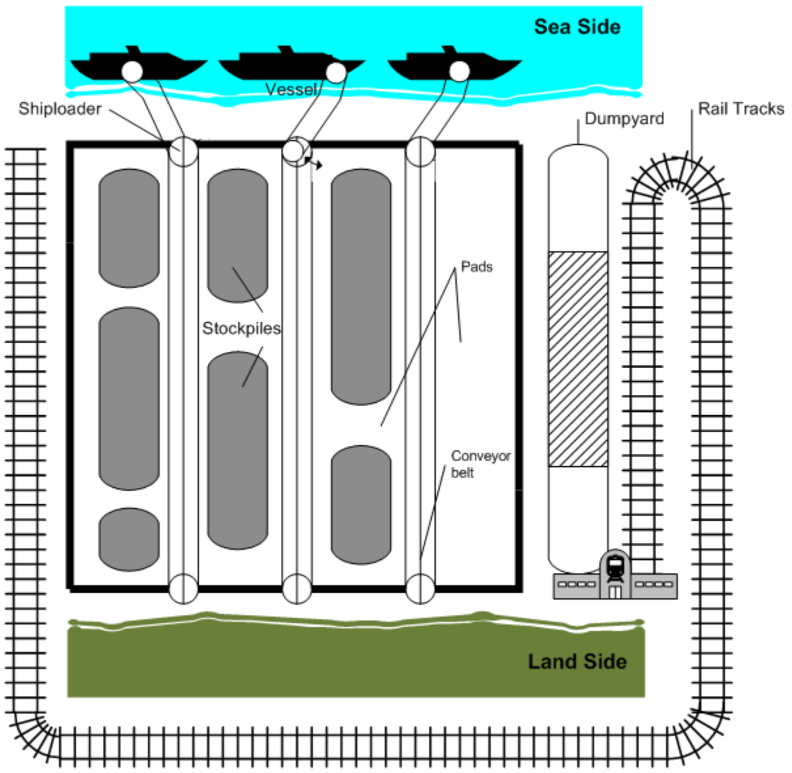

Figure 2. An overview of a bulk port terminal

According to UNACTAD (2014), world dry-cargo transportation grew 5.5% and it was 6.7 billion tons in 2013. The trade of dry bulk commodities continues to increase and constitutes the largest share of dry-cargo volumes (Clarkson Research Services, 2014a).

Moreover, dry bulk materials can be categorized into two as major and minor bulk materials. Major bulk materials include coal, iron ore and grain and these commodities have the largest share by volume among the dry bulk cargoes. Minor bulk materials are forest products, agricultural cargoes and cement. Every year, approximately 2.9 billion tons of major dry bulk materials and 1.4 billion tons minor dry bulks are transported (Clarkson Research Services, 2014a).

1.1.2 Stockpiles

A stockpile is a large supply of bulk material that is kept for future use (Stockpile, 2015). Since bulk materials are stored without any containers, they are piled at the stockyard and stored as stockpiles. Hence it is an important aspect of the bulk material handling process. Stockpiles are used to store bulk materials in different environments, such as in a port, refinery or manufacturing facilities. An example of a stockpile can be seen in Figure 3.

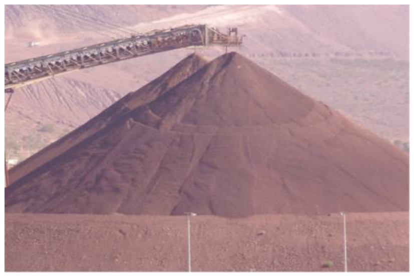

Figure 3. A stockpile of iron ore

Because of the large quantities and the uniformity of the materials, they can be built in huge conical shapes at the stockyard. A typical storage form is piles having rectangular bases. Figure 4 shows a typical bulk port stockyard and stockpiles. Since bulk materials are in large volumes, their locations have a strong effect on the utilization of a stockyard.

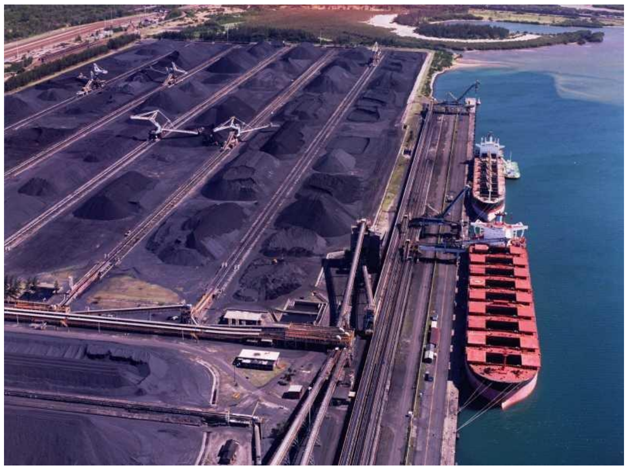

Figure 4. A typical bulk port stockyard

1.1.3 Pads, Conveyor Belts and Stacker-Reclaimers

At the stockyard of most of the bulk port terminals, there are pads where the bulk materials are piled. These pads are separated from each other with a conveyor belt which carries the materials. In Figure 4, there are six pads which lie along the quay side. There are four stackers-reclaimers moving along the conveyor belts, which were built at the stockyard to carry materials.

Stockpiles are formed using stackers to build piles along the length of a conveyor. In most of the cases, stockpiles are built so that they span the entire width of the pad. Therefore, the width of the stockpiles can be assumed as equal to the width of a pad. After a vessel arrives to the berth, reclaimers retrieve the material from the stockyard for loading.

1.1.4 Rail tracks

In general, there are rail tracks around the ports for the trains which bring the bulk material to the port (Figure 5). Trains may come from one or more suppliers to the port.

Usually, these materials are unloaded to the dump stations, and then carried to the stockyard. The scheduling of the trains is also an important problem and studied by Liu and Kozan (2009), but it is outside the scope of this thesis.

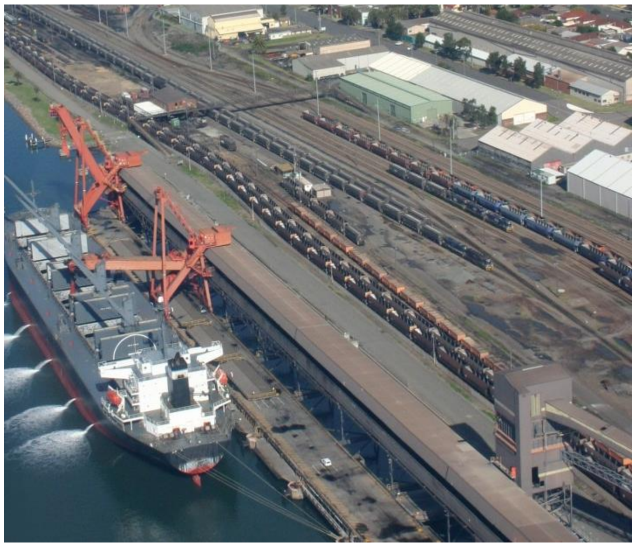

Figure 5. Rail tracks around a port terminal

1.2 Operations in Bulk Port Terminals

Robenek *et al.* (2014) divides the operations in bulk port terminals into two levels: (1) Tactical level and (2) Operational level. Tactical level operations include medium to short term decisions. For example, berth allocation problem and stockyard management decisions can be listed as the most significant tactical level decisions in bulk port terminals. On the other hand, daily operations such as crane scheduling, yard equipment selection can be handled at the operational level. In this study, we focus on a tactical level operation which is the stockyard allocation problem.

The vessels arrive at bulk ports to be served, loaded or unloaded. Ship loaders serve the upcoming vessels, by either loading them with the material at the stockyard or unloading the material from the vessel. Berth allocation problem (BAP) refers to assigning berths to the incoming vessels efficiently so that the demand of the vessels is met with minimum delay. BAP mostly considers the decisions of vessels' docking locations and the berthing sequence, concurrently. In most of the bulk ports, the materials are brought to the stockyard according to the berthing time of vessels. Therefore, the results of BAP are essential for stockyard allocation. The departure times of vessels are provided by solving BAP. Thus, the output of BAP can be used as an input to yard allocation problem (YAP).

In practice, these decisions can be made by the port managers who are experienced and have the insight into port operations. Additionally, scientific approaches based on operations research methods can be used as decision support systems.

Although yard assignment (or storage allocation) problems at the yard side of the ports have been studied for many years, in most of the studies, researchers mainly focus on storing containers and bulk port terminals receive less attention than they deserve. In many port terminals, dry bulk materials are stored and shipped on a regular basis. Therefore, in this thesis, the yard assignment of bulk materials, such as coal, in bulk ports are investigated.

1.3 Outline

The organization of the thesis is structured as follows. In Chapter 1, the operations of bulk port terminals are briefly introduced and the related definitions are provided. In Chapter 2, the yard allocation problem is explained and the associated studies are reviewed. In addition to port operations, multi-dimensional packing problems and related metaheuristic approaches are surveyed. In Chapter 3, the methodologies adopted in this thesis are presented. Chapter 4 provides mixed integer linear programming (MILP) models to solve yard allocation problem. In addition to exact solutions, Chapter 5 proposes metaheuristic algorithms based on Tabu Search (TS) and Variable Neighborhood Search (VNS). Implementations of TS and TS hybridized with VNS are discussed in this chapter. Chapter 6 reports the results of MILP models and metaheuristics, and provides a comparison of the models. Finally, Chapter 7 concludes with future research possibilities and important remarks.

Chapter 2

PROBLEM DEFINITION AND LITERATURE REVIEW

2.1 Problem Definition

The tactical operations at the container terminals (CT) have been studied for many years. Comprehensive surveys about CT are provided in Carlo *et al.* (2014), Bierwirth *et al.* (2010) and Stahlbock and Voß (2008). Although bulk port terminal and container terminal operations are similar, their yard sides and storage areas are significantly different. In general, containers can be placed on top of each other. However, this cannot occur at the bulk port stockyards due to the nature (characteristics) of the piles. Bulk materials are stored without any containers therefore they need to be piled on the ground. Piles are built as rectangular pyramids; thus, the surface area has a rectangular shape. This rectangular surface area of each stockpile will represent its size. In this research, the stockyards which only have open air storage areas are considered. Therefore, the height of each stockpile is disregarded. Considering this property of the bulk materials, we can approach this problem as a multi-dimensional packing problem.

The ultimate goal of this study is to develop a methodology to determine the location of products at the stockyard while minimizing the total travel distance and the total dwell time of materials that are handled in dry bulk material ports. Thus, we developed three mathematical models which address the yard assignment problem (YAP) by generalizing the well-known multi-dimensional packing models. In our models, we consider two types of stockyards; with pads and without pads. In addition to the assignment of materials to the storage locations, first model also determines the orientation of the bulks given that the dimensions of each bulk are known. However, since multi-dimensional packing problems are NP-hard, developing a computationally efficient mathematical model is challenging. Thus, a hybrid metaheuristic which is based on Tabu Search (TS) and Variable Neighborhood Search (VNS) to solve YAP in large-scaled bulk port terminals is presented.

In order to solve the stockyard assignment problem, two simultaneous decisions should be made. The first outcome is the location of the stockpiles at the yard which will be loaded to the arriving vessels to minimize the total travel distance of these materials from their storage point to the vessel. Secondly, when to start building these stockpiles should be determined in contemplation of minimizing the dwelling time of a stockpile at the stockyard. Since the ultimate goal of the bulk material terminals is to serve vessels with minimum delay, the stockpiles must be loaded to the vessels before the concerned vessels depart. Hence, the models proposed in Chapter 4 and Chapter 5 minimize the cost of dwelling time and the cost of travel time of a stockpile at the stockyard.

2.2 Assumptions

In this thesis, the following assumptions are made for YAP:

- The stockyard and the stockpiles are rectangular.
- The widths and lengths of stockpiles are known.
- The dimensions of the stockyard are known.
- In some cases, there are dump stations around the stockyards where trains bring the material and piles located onto these places and then they are moved by conveyor belts to the stockyard (Boland, 2012). In this thesis it is assumed that trains bring the bulk material directly to the stockyard if the stockyard is available at that

moment; otherwise piles dwell in the dump stations. Therefore, the time that a stockpile spends on the dump station is also important.

- Building a stockpile can only start after the nomination of a vessel which will carry that stockpile.
- A stockpile must leave before the ship's departure from the berth.
- Since bulk materials are stored in huge volumes, once a stockpile is started to build, it cannot be moved to another place.
- The storage times of stockpiles are known.
- The arrival times of bulk materials are known.
- The departure times of vessels are known.
- The stockyards with and without pads are considered. The layout of these stockyards can be seen in Figure 6.
- If pads exist, stockyard consists of 4 pads having same widths and lengths.

Additionally, the following additional constraints which may be applied to packing problems are investigated to determine the similar constraints of some multi-dimensional packing problems that correspond to the physical constraints of a bulk port stockyard. One of these constraints is orientation constraint. While placing items into the object, items may have fixed orientation as well as they can be rotated by 90 degrees. Orientation constraint ensures that the orientation of all items is fixed. Additionally, there are guillotine constraints. If guillotine constraint is required then all items must be obtained through a sequence of edge-to-edge cuts. Thus, one can attain four subtypes of two-dimensional packing problems if these two constraints are combined. These four subtypes can be listed as below:

- RF: Items can be rotated 90 degrees and no guillotine cutting is required

- OF: Orientation of items is fixed and no guillotine cutting is required
- RG: Items can be rotated 90 degrees and guillotine cutting is required
- OG: Orientation of items is fixed and guillotine cutting is required

The physical conditions of a stockyard may require orientation constraint where all stockpiles must be parallel to the pads. In the presence of pads we solve the yard allocation problem with OF constraints. Additionally, for some small stockyards, the piles can be built freely at the stockyard. Therefore no orientation constraint is required. For this type of ports, we solve the problem with RF constraints. The layouts of both stockyards can be seen in Figure 6.

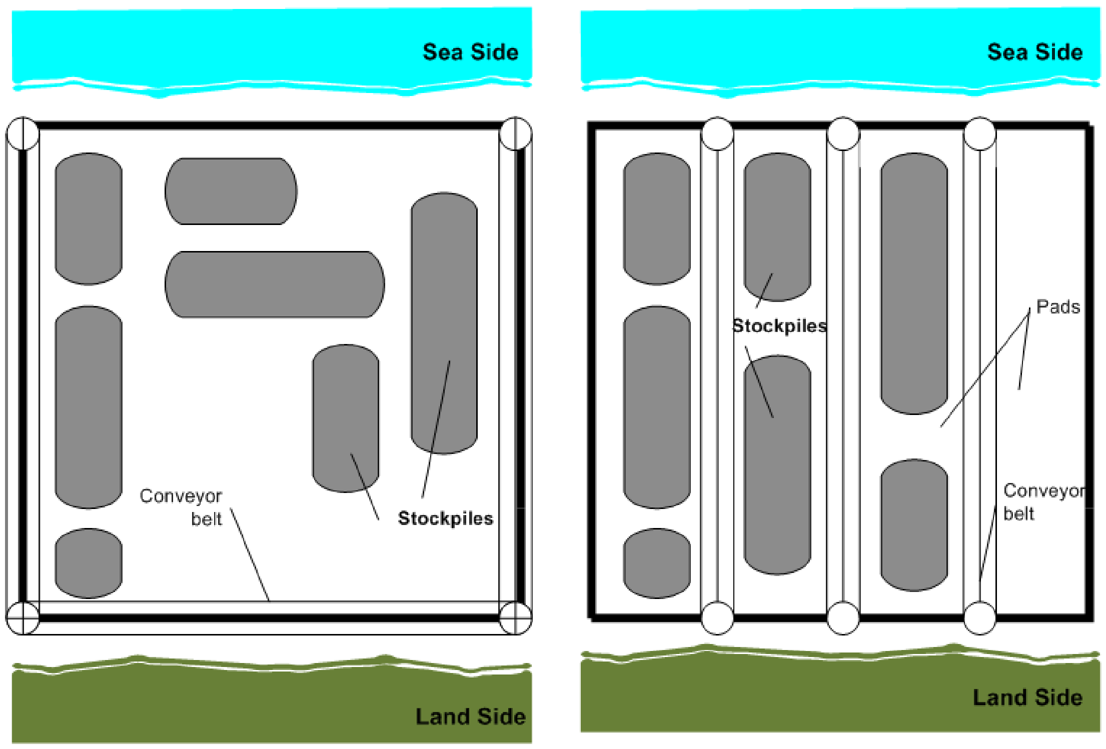

Figure 6. A stockyard without pads (left) and a stockyard with pads (right)

2.3 Literature Review

Although the focus of this research is the yard allocation problem in the bulk port terminals, in order to better grasp the nature of the problem and methodologies used in related studies, an extensive literature review on general bulk port operations, yard allocation problem in bulk port terminals, multi-dimensional packing problems and metaheuristic algorithms is conducted and presented in this chapter.

2.4 General Bulk Port Operations

Schott and Lodewijks (2007) identify the emerging research areas with increasing supply and demand trend in the dry bulk market. In this paper, the dry bulk terminals are analyzed in terms of terminal logistics, layout and design, storage and handling of the bulk material, environmental and maintenance issues. For this purpose, the bulk terminals in the Le Harve – Hamburg area are investigated. The mentioned terminals are examined and compared in terms of handling, discharge capacities, and flexibilities of different equipment. Additional costs, environmental issues, and contamination problems are discussed for the open air and the covered storage facilities. Since there are various dry bulk terminal layouts, the arrangements of the overall layout, equipment and routing, size of the stockyard as well as maintenance and stockyard management can differ and these are important to study in order to balance the utility of different parts of the system.

Junior *et al.* (2012) introduces a greedy heuristic for berth allocation in tidal bulk ports to minimize the overall service time. They assume a discrete space for quay length, tidal time windows are known in advance and tidal waves occur in a regular frequency. In addition, time scale is also assumed to be discrete. The stock level issues in stockyard are not considered in this study. The objective is to minimize the total service time over the planning horizon. It is assumed that the expected time of arrival (ETA) is known and used for prioritizing vessels accordingly. First, vessels are sorted according to their ETAs. First-in-first-out (FIFO) rule can be used for sorting. Since the heuristic is greedy, vessels are prioritized according to their expected time of arrivals and then using the FIFO rule vessels are assigned with the minimum operational cost. One can obtain quick solutions using this approach; however, these solutions are limited to local optimal. These results showed that further improvements should be done such as adjustment of the port lineup to avoid stock level crashes.

Hu and Yao (2010) study Stacker-Reclaimers (S-R) scheduling problem in bulk port terminals to provide more efficient yard operations. The objective function of the proposed mixed integer programming (MIP) model is to minimize the makespan of the serviced vessels. Since the problem is considered as NP-hard, a parthenogenetic algorithm is introduced to solve the S-R scheduling problem near optimal.

Articles reviewed in Section 2.4 provide a better understanding of bulk port operations in general. Some real life constraints are originated from these papers.

2.4.1 Yard Allocation Problem (YAP)

Although YAP in container terminals appears in the literature considerably, YAP in bulk port terminals receive less attention. To the best of our knowledge, a few studies are conducted in an integrated setting. Since the berth allocation and yard management are interrelated problems, sometimes researchers develop integrated models for these two problems. However, the proposed solution methods are either metaheuristics which cannot guarantee optimality or complex mixed integer programming models which cannot solve even small-sized instances to optimality within an acceptable amount of time. Despite the application of decomposition techniques, BAP and YAP with instances for medium-sized port terminals cannot be solved to optimality. Additionally, in consideration of stockyard management, the determination of stockyard size for dry bulk terminals is also important.

Robenek *et al.* (2014) provide an overview of bulk port terminal operations. They provide insight on making tactical and operational level decisions. They mainly focus on tactical level decisions which are the berth allocation problem (BAP) and the yard assignment problem (YAP). They present an exact method as well as a metaheuristic algorithm to solve the integrated BAP and YAP. They examine a bulk port terminal where the length of quay is continuous, vessel arrivals are dynamic and the yard space is discretized. They divide stockyard into several locations and quay side is partitioned into several sections. Their aim is to minimize the total service time of vessels. They consider small-, medium- and large-sized instances with 10, 25 and 40 vessels, respectively. Additionally, on the yard side, they consider liquid bulk and dry bulk materials together. An exact solution algorithm is proposed based on branch and price. Since the proposed mixed integer programming model for integrated problem is complex, it is not likely to solve BAP and YAP to optimality even for small-sized ports. Therefore, to reduce the problem size, a set partitioning problem is formulated. However, they can only solve some small-sized problems to optimality using the set partitioning decomposition technique. Moreover, solving medium- and large-sized instances takes hours, with an average 4.11% and 3.77% optimality gaps, respectively. In addition, they develop a metaheuristic approach based on critical-shaking neighborhood search (CSNS) in order to obtain near optimal solutions in a short time. However, although the running time is improved for all instances, the average of optimality gaps of the results obtained with the CSNS heuristic are 4.78%, 8.02% and 31.16% for small-, medium- and large-sized instances, respectively. The results indicate that, although CSNS can obtain solutions within a minute, one disadvantage of this algorithm is that the resulting objective values for large-sized instances deviates 31.16% from the best known objective value (i.e. the upper bound of objective function obtained by running the exact mathematical model, for an average of approximately 3 hours) on average.

Boland *et al.* (2012) describe the coal terminals in Port of Newcastle, Australia. They explain port operations in detail as well as the transportation of coal from mine load points to the terminals. They differentiate different terminal types with respect to operating characteristics such as push based manner or pull based manner. Additionally, they provide real life data and constraints which must be complied in the course of the operations of bulk port terminals They also focus on a stockyard planning problems which is an integrated problem consisting of berth allocation decisions, stockpile location decisions and stockpile assembly start time decisions. They consider stream of ship arrivals and reclaiming start times of materials. Their objective is to maximize throughput which is obtained by minimizing the mean delay time of ships. They introduced a hybrid algorithm which consists of a greedy construction heuristic, enumeration and integer programming. They develop a stockyard planning technology which can be applied to the coal terminals for better management. However, since they do not provide any exact model, they only compare the efficiency of proposed heuristic algorithms. They test the efficacy of their algorithms with real-life data from the Port of Newcastle.

Vianen *et al.* (2013) propose a methodology, sustained with a simulation tool, to decide the required stockyard size in bulk port terminals. They focus on stockyards for dry bulk materials such as coal or iron ore. They investigate inventory models to store dry bulk materials and provide a simulation based approach to determine the stockyard size for bulk terminals. Finally, they perform a case study, in which they apply their methodology to the mentioned problem and present their findings.

2.4.2 Multi-Dimensional Packing Problems

Since YAP in bulk port terminals resembles multi-dimensional packing problems (MDPP), exact models and heuristic algorithms should be examined to incorporate the similar constraints in both problems.

Multidimensional packing operations can be observed in different environments. Hopper and Turton (2001) define packing problems as optimization problems engaging in to find a good arrangement of multiple smaller items in larger container objects (usually called as bin). Thus, in many industries packing problems need to be solved with different constraints including aforesaid orientation and guillotine constraints and different objectives. In most of the industries, manufacturing, service or telecommunication, some portion of the limited resources need to be allocated to the tasks or jobs. For example, manufactured goods need to be packed into containers to be shipped and the objective is minimizing the number of these containers. This type of packing problems called three-dimensional bin packing (3D-BP) problems and it is one of the well-known MDPP. In YAP a pad can be considered as a bin whereas, stockpiles are items that are placed on the pads. Additionally, since we study only open air storage areas, the height of the stockpiles are not considered. Therefore, the third dimension of a bin can represent the planning horizon.

Chen *et al.* (1995) study three-dimensional palletization and developed a mathematical model to place items into bins while minimizing the unused space. They consider multiple containers and multiple item sizes. They also investigate the cases where changing the orientation of an item in a bin and overlapping of items in a container are allowed. In one of our models we allow rotating stockpiles 90 degrees at the stockyard. Therefore, we borrow the concept which combines non-overlapping constraint and the orientation constraints for three dimensional packing problems from Chen *et al.* (1995).

Bay *et al.* (2010) focus on space and time allocation problem for shipyard assembly halls which is closely related to the three-dimensional bin packing problem due to the non-overlapping constraints. They study space and time allocation for blocks (prefabricated keel elements), which cannot be placed on top of each other; therefore, they relate the third dimension in the bin packing problem with the time allocation to produce these large building blocks. In addition, they develop a Guided Local Search (GLS) heuristic for the 3D-BP problem. Their study aims to order the blocks into the rectangular areas, without overlap, and without violating the time constraints.

Martello and Vigo (1998) evaluate the two-dimensional finite bin packing problem. They analyze a well-known lower bound and perform a worst-case analysis. Additionally, they propose new lower bounds to use within a branch-and-bound algorithm. Their results indicate that the worst case performance analysis is valid even if the items are allowed to rotate by any angle.

Lodi *et al.* (2002a) present a survey on two-dimensional packing problems. They discuss mathematical models, survey lower bounds, approximation algorithms, heuristic and metaheuristic methods and exact enumerative approaches. They include papers which present two-dimensional bin and strip packing problems and knapsack problems. Exact models and approximation algorithms are presented in detail.

Lodi *et al.* (2002b) focus on recent advances on two-dimensional bin packing (2D-BP) problems. Upper and lower bounds, exact models, heuristic and metaheuristic algorithms are presented. They discuss the variants of 2D-BP problems. In real life contexts additional constraints such as orientation and/or guillotine cuts may be required. A typology for the possible cases produced by the aforesaid constraints is presented.

Arahori *et al.* (2012) propose an exact algorithm to solve the two-dimensional strip packing (2D-SP) problem with and without 90 degrees rotation of items. They derive a methodology to obtain new lower bounds and use the branch and bound method. The computational results indicate that their algorithm succeeds to find the optimal solutions for some of the benchmark instances which have not been solved by any of the other algorithms.

Castro and Oliveira (2011) develop scheduling inspired models for two-dimensional packing problems. They developed exact mixed integer linear programming models for two-dimensional packing problems as well as hybrid discrete/continuous time-space heuristic algorithms. They also consider the rotation of the rectangles while packing them.

Along with the other two-dimensional packing problems Lodi *et al.* (2002) investigate the 2D-BP problems. They examine mathematical models, and survey lower bounds, approximation methods, heuristic and metaheuristic algorithms and exact enumerative approaches. Additionally, they consider some special cases where the items have to be packed into rows by forming levels. The level structure in the paper can be incorporated to our model as the arrival-departure times of stockpiles. In addition to all approximation algorithms, they also consider the proposed metaheuristic techniques with lower bounds for 2D-BP problems.

2.4.3 Metaheuristic Approaches

In the literature, metaheuristic algorithms are studied, surveyed and presented in many research articles. Since the scope of this thesis is yard allocation problem inspired by multi-dimensional packing models, metaheuristics including but not limited to Tabu Search (TS), Genetic Algorithms (GA) and heuristic algorithms Bottom-Left (BL), First Fit (FF) for 2D-BP, 2D-SP problems and container loading problems (CLP) are reviewed in this section.

Tabu Search (TS) is introduced by Glover (1986) and it guides a local search method to examine the solution space beyond the local optimality. According to the classification suggested by Glover and Laguna (1998) TS exploits adaptive memory, some systematic neighborhood search to select the next move or to improve the current solution and moves from one solution to another in each iteration.

Mladenović and Hansen (1997) propose the Variable Neighborhood Search (VNS) which systematically allows changes in the neighborhood. Since then, VNS is used to solve various combinatorial optimization problems. Beltrán *et al.* (2004) proposed a hybrid VNS and Greedy Randomized Adaptive Search Procedure (GRASP) algorithm for the strip packing problem and Behnamian *et al.* (2009) solve parallel-machine scheduling problems with sequence-dependent setup times using an Ant Colony Optimization (ACO) with VNS and Simulated Annealing (SA) with VNS algorithms.

Jain and Gea (1998) study two-dimensional packing problems using genetic algorithms. They introduce a new concept of two-dimensional genetic chromosome. Moreover, the results indicate that their approach is suitable and quite effective for packing problems with any kind of items including irregular items with holes.

Berkey and Wang (1987) analyze the performance of existing and a new packing heuristic for a special case of 2D-BP problem where the number of bins is finite. They propose a new heuristic by adapting well-known heuristics such as bottom-left, next-fit, first-fit, and best-fit to finite bin case. They implement a best-fit-decreasing-height heuristic followed by a best-fit-decreasing heuristic and called it finite best-strip. Their results imply that the proposed heuristic reflects the proven worst case bounds for the existing two-dimensional packing heuristics.

Bortfeld and Gehring (1998) propose Tabu Search to solve container loading problems (CLP). CLP resembles to multi-dimensional packing problems where there is only one container (bin) and a set of rectangular goods, referred to below as items. The aim of this problem is to find a feasible packing of all items in the container to maximize the stowed item volume. Additionally, any orientation or guillotine cut constraints can be required. In Bortfeld and Gehring (1998) the solutions are created by using an integrated greedy heuristic. Thus, the overall control mechanism of the heuristic is a superior TS. In addition, Bortfeld and Gehring (2001) design a genetic algorithm to solve CLP. Later, Bortfeld

(2005) adapt Bortfeld and Gehring’s (2001) GA for the 2D-SP problem with rectangular pieces. The aim of the proposed algorithm is to provide a GA for the 2D-SP problem that works without encoding any solution.

Hopper and Turton (2001) hybridize two heuristic approaches which are based on bottom-left (BL) heuristic with three meta-heuristic algorithms (genetic algorithms (GA), simulated annealing (SA) and naïve evolution (NE)) and local search heuristic (hill-climbing) to solve two-dimensional packing problems. They compare the results of the algorithms with respect to the solution quality and the computational time. In their study, SA achieves best solution quality whereas the computational time of GA and NE are less than that of SA.

Lodi *et al.* (1999) investigate all four cases of bin packing problems that are mentioned before. Those cases arise from the restrictions made on the orientation of the items and whether they can be obtained through guillotine cuts or not. They introduce a new heuristic for each class of problem and a TS which can be adapted to these specific problems only by applying the heuristic introduced for that problem. They conclude that among the different metaheuristic algorithms, TS recently proved to be particularly effective for 2D-BP problems. The essential characteristic of their TS algorithm is that the neighborhood structure is dynamically changing its size.

In addition to these papers, Coffman *et al.* (1983) generalize the online bin packing models into dynamic one-dimensional bin packing (1D-BP) problem. They mainly focus on FF algorithm to obtain a performance bound. They claim that no online packing algorithm can perform better than FF in terms of bounds.

Epstein and Levy (2010) enhanced the idea of dynamic 1D-BP problem into dynamic multi-dimensional bin packing problem. The aim is to pack arriving multi-dimensional items, such as rectangles or boxes into bins, such as squares, or cubes. They consider the online problem where the future input, that is, arrival or departure of an item, is not known.

However, in this thesis, it is assumed that arrival and departure times of items (stockpiles) are known at the beginning of the planning horizon. Moreover, they consider where items are assigned to bin permanently but relocation of an item within a bin is possible. In the classical multi-dimensional online packing problems, when an item is assigned to a place, its location is not changed. We also adopt the classical approach, where if a stockpile is built at the stockyard, then its location does not change during the planning horizon.

Similar to Epstein and Levy (2010), Han *et al.* (2010) study dynamic bin packing of unit fraction items. They also focus on FF algorithm to find theoretical results.

For further information about metaheuristics and their applications the reader is referred to Laporte and Osman (1995), Gendreu *et al.* (1995), Al-Mahmeed (1996), Charon and Hudry (1996), Osman and Kelly (1996).

Chapter 3

METHODOLOGY

To the best of our knowledge, since bulk port terminals receive less attention than container terminals, only a few exact and heuristic approaches are proposed to solve yard allocation problem in bulk port terminals in the literature. In order to provide better yard management for port terminals, we adopt two approaches in which we provide exact solutions and near optimal but quick solutions. The methodologies used in this thesis to solve the yard allocation problem are provided in this chapter.

Three mixed integer linear programming models focusing on different characteristics of the problem are developed to determine the optimal positions of every stockpile on the yard and the optimal start time of building a stockpile. Additionally, metaheuristic algorithms are proposed to provide a better yard management for large-scaled bulk port terminals. In Section 3.1 a brief explanation about the MILP models is provided.

Moreover, a TS which utilizes a bottom-left-fill like heuristic is proposed to obtain near optimal results within a reasonable time. Furthermore, to obtain better results, a hybridization of the TS with VNS is presented. Therefore, in Section 3.2 the basic steps of TS and VNS algorithms are discussed.

3.1 Mixed Integer Linear Programming Models

Mixed integer linear programming (MILP) models are used to solve various combinatorial optimization problems. They are mathematical models that consist of both continuous and integral decision variables, a linear objective function and linear constraints. The running time of the MILPs solely depend on the complexity of the problem. Thus, for complex problems, running an MILP can be very time consuming even for mid-size instances. On the other hand, they provide the optimal solution (i.e. global maximum or minimum, depending on the nature of the objective function) of the problem.

In this study, three MILP models inspired by multi-dimensional packing problem models are introduced to solve the continuous and the discrete yard allocation problems individually.

The first and the second models proposed in this thesis address the YAP in a continuous time and continuous space. Continuous YAP is studied where the length of the stockyard as well as the planning time horizon are continuous. On the other hand, the third model we propose discusses the YAP in discrete time and discrete space. In all models the same two objectives are adopted: minimizing the total distance travelled at the stockyard and minimizing the total dwelling time over all stockpiles.

Although most of the real-life bulk port terminals have conveyor belts that were built at the stockyard and divide it into small areas called pads, first a more general purpose mathematical model is proposed. Then, this property of bulk port stockyards is considered in the second and the third models.

The stockpile assignment problem resembles to multi-dimensional packing problem where there are multiple objects (bins) having the length and the width same as the pads at the stockyard and the items are stockpiles which will be placed on the pads. Since the height of each stockpile will not be important (the stockyard is an open air storage area), we can use the third dimension as the time.

There is no need to force guillotine constraints which are discussed in Section 2.2 in a bulk port stockyard due to physical capabilities of yards. On the other hand, in most of the stockyards, the width of pads determines the width of stockpiles therefore the orientation of the piles must be fixed. However, some small-sized stockyards may allow rotation of stockpiles if they do not require conveyor belts at the stockyard. If the considered stockyard is assumed to be without pads, then the orientation constraint can be relaxed. Therefore, in this thesis, RF and OF types of multi-dimensional packing models are considered.

3.2 Metaheuristic Approach

Talbi (2009) defines heuristics as methodologies used to find 'good' solutions for large-size problem instances. Heuristics perform at acceptable costs and level in a wide range of problems. However, they do not guarantee any approximation bounds on the obtained solutions. Thus, they are designed and tailored to solve a specific problem and/or an instance.

Metaheuristics, on the other hand, are general-purpose algorithms. They can be adopted to solve almost any optimization problem. Metaheuristics utilize high level methodologies when compared to heuristics. They can be used as guidelines that control the underlying heuristics.

Metaheuristic term is coined with the Glover's (1986) paper in which the Tabu Search is introduced. Thereafter, metaheuristics became widely applied methods to find good quality solutions for optimization problems.

Although many metaheuristics exist in the literature, notable examples can be listed as simulated annealing (SA), tabu search (TS), genetic algorithms (GA), variable neighborhood search (VNS), ant colony optimization (ACO), naïve evolution (NE) and particle swarm optimization (PSO). In this thesis, a hybridization of TS and VNS are utilized.

Hopper and Turton (2000, 2001) differentiate metaheuristics by their encoding and decoding requirements. They divide metaheuristics into three groups as follows:

1) In the first group, algorithms use encoding and decoding the solutions in order to obtain final results. The search is conducted in the space of the encoded solution. Move operators, crossover or mutation operators are always problem-independent. Thus, a decoding routine is required to transform encoded solutions into complete solution space.

2) The second group incorporates methods that usually uses encoded solutions but these solutions already contain geometrical information about the final solution. Mostly problem specific operators are used in these algorithms. Although there is information about the solution space information within the encoded solution space, additional decoding routine is required for the final positioning.

3) Finally, search is conducted directly in the fully defined solution space for the third group. Since the methods search the original problem space, the solutions are manipulated by specific operators. As a result, additional decoding routine is not required in this group of algorithms.

In this thesis, a tabu search (YATS) and a tabu search hybridized with variable neighborhood search (YATS_VNS) algorithms which uses a Bottom-Left-Fill (BLF) like heuristic are proposed. Detailed explanations of these models and the components of proposed YATS and YATS_VNS are provided in Chapter 5.

The proposed algorithms are included in the first category characterized by Hopper and Turton (2000, 2001). Therefore, problem independent neighborhoods are selected. Moreover, a complex encoding and decoding routine is needed in order to obtain the final solution. The algorithms utilize a BLF like heuristic as a decoder to transform encoded solutions into the stockyard assignments to the piles. BLF is a well-known heuristic for 2D packing problems.

3.2.1 Tabu Search

In the literature, metaheuristics are grouped under various categories according to different characteristics of them. Glover and Laguna (1998) classify metaheuristics in terms of three basic design categories: (1) the use of memory, (2) the used neighborhood exploration strategy and (3) the number of solutions generated to carry next iteration. According to these categories, TS exploits adaptive memory, a systematic neighborhood search to select the next move or to improve the current solution, and moves from one solution to another in each iteration.

The basic TS algorithm is introduced by Glover (1986). Basic steps of TS Algorithm are provided in Figure 7. TS guides a local search method to examine the solution space beyond the local optimality. The most important component of TS is the tabu list. Recently visited solutions are recorded in a tabu list and this list comprises the memory component of TS. The solutions in the tabu list are restricted to be visited for a predetermined number of iterations. Another component of TS is tabu tenure. It denotes how many iterations a solution will be kept in the tabu list. Therefore, the tabu tenure and the attributes of a solution which will be added to the tabu list should be determined prior to the search.

The basic steps of tabu search can be summarized as follows. The search starts with an initial solution and an empty tabu list. Therefore, the solutions only in the admissible neighborhood, that is, solutions which are not in the tabu list can be visited in the following iterations. After evaluating the admissible neighborhood, the best solution is chosen. Then, the selected solution is added to the tabu list. Moreover, the tabu list and the objective value are updated accordingly. This process continues until a stopping condition is evoked. By this approach, possible cycles can be avoided. Thus, one can escape from a local optimum.

SIMPLE TABU SEARCH

1. Select an initial $x \in X$and let $x^* \coloneqq x$. Set the iteration counter $k = 0$ and begin with T empty.
2. If $S(x) - T$is empty, got to Step 4.
   Otherwise, set $k \coloneqq k + 1$ and select $s_k \in S(x) - T$ such that
   $s_k(x) =$ OPTIMUM $(s(x){:}\, s \in S(x) - T)$.
3. Let $x \coloneqq s_k(x)$. If $c(x) < c(x^*)$, where x^* denotes the best solution currently found, let $x^* \coloneqq x$.
4. If a chosen number of iterations has elapsed either in total or since x^* was improved, or if $S(x) - T = \emptyset$ upon reaching this step directly from Step 2, stop.
   Otherwise, update T (as subsequently identified) and return to Step 2.

Figure 7. Basic steps of TS Algorithm (Glover, 1986)

3.2.2 Variable Neighborhood Search

Mladenović and Hansen (1997) propose Variable Neighborhood Search (VNS), a metaheuristic which systematically allows changes in the neighborhood. Due to relative easiness of application of the procedure, VNS can be used as a standalone approach as well as a hybridization element of other metaheuristics to solve various combinatorial optimization problems. For example, Beltrán *et al.* (2004) proposed a hybrid VNS and GRASP algorithm for the strip packing problem and Behnamian *et al.* (2009) solve parallel-machine scheduling problems with sequence-dependent setup times using an ACO-VNS and SA-VNS algorithms.

VNS applies a local search within multiple neighborhoods systematically. In order to apply VNS, maximum number of neighborhoods, the structures of these neighborhoods and the local search procedure should be determined.

The basic steps of variable neighborhood search can be summarized as follows. First, an initial solution is identified. Then, the first neighborhood of this solution is searched to obtain a new solution in random. Then, a local search is performed on the selected random solution and if a better solution is achieved, the incumbent and the best objective values are updated. Moreover, the search continues with the first neighborhood. Otherwise, the neighborhood is changed and the above steps are performed in that neighborhood. Whenever a better solution is obtained, then the search returns to the first neighborhood. Until a stopping condition is met, the previous steps are repeated. The steps of the basic VNS proposed by Mladenović and Hansen (1997) can be seen in Figure 8.

```
Initialization Select the set of neighborhood structures $N_k$ for $k = 1, \dots, k_{max}$ that will be used in the search; find an initial solution $x$; choose a stopping condition;
Repeat the following sequence until the stopping condition is met:
(1)Set $k \leftarrow 1$;
(2)Repeat the following steps until $k = k_{max}$:
    (a) Shaking Generate a point $x'$ at random from the $k^{th}$ neighborhood of $x\big(x' \in N_k(x)\big)$;
    (b) Local search Apply some local search method with $x'$ as initial solution; denote with $x''$ the so obtained local optimum;
    (c) Move or not If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $N_1 (k \leftarrow 1)$; otherwise, set $k \leftarrow k + 1$;
```

Figure 8. Steps of the basic VNS (Mladenović and Hansen, 1997)

Chapter 4

MIXED INTEGER LINEAR PROGRAMMING MODELS TO SOLVE YAP

In this chapter, three mixed integer linear programming (MILP) models to solve yard allocation problem in bulk port terminals are proposed. We introduce two continuous and one discrete MILP model by generalizing the well-known multi-dimensional packing models with additional constraints derived from real life bulk port terminals.

4.1 Continuous Yard Allocation Model in the Absence of Pads (CSA)

In small-sized terminals, the bulk material handling may be performed by mobile equipment such as loaders and various shuttles instead of stationary machinery such as conveyor belts which are built at the stockyard. In the absence of conveyor belts, hence pads, it is assumed that the area of the stockyard is a whole rectangle. Similarly, all piles are assumed to be rectangles and the orientation of each pile at the stockyard need to be decided. In this section, first, the sets, parameters, the decision variables are defined. Secondly, the continuous stockyard allocation model is presented. Finally, the objective function of the model and the constraint sets are explained.

Sets and Parameters

S	Set of stockpiles.
L	Length of the stockyard.

W	Width of the stockyard.
T	Maximum time period.
M	Arbitrarily large number.
ct	Cost of transferring a stockpile per meter.
cw	Cost of dwelling a stockpile per hour.
$arrTime_s$	Arrival time of stockpile $s \in S$.
$depTime_s$	Departure time of vessel that stockpile $s \in S$ will be uploaded into.
$length_s$	Length of stockpile $s \in S$.
$width_s$	Width of stockpile $s \in S$.
$storageTime_s$	Storage time of stockpile $s \in S$ at the stockyard.
$ConnectX$	X-coordinate of the connection points of conveyor belts to the ship loaders.
$ConnectY$	Y-coordinate of the connection points of conveyor belts to the ship loaders.

Decision Variables

x_s: Position of stockpile $s \in S$ on the X-axis.

y_s: Position of stockpile $s \in S$ on Y-axis.

t_s: Start time of building stockpile $s \in S$.

$$o_s = \begin{cases} 1, \text{ if lenght of stockpile } s \in S \text{ is parallel to X-axis,} \\ 0, \text{ otherwise.} \end{cases}$$

$$a_{sk} = \begin{cases} 1, \text{ if stockpile } s \in S \text{ is at the left side of stockpile } k \in S, \\ 0, \text{ otherwise.} \end{cases}$$

$$b_{sk} = \begin{cases} 1, \text{ if stockpile } s \in S \text{ is at the right side of stockpile } k \in S, \\ 0, \text{ otherwise.} \end{cases}$$

$$c_{sk} = \begin{cases} 1, \text{ if stockpile } s \in S \text{ is behind stockpile } k \in S, \\ 0, \text{ otherwise.} \end{cases}$$

$$d_{sk} = \begin{cases} 1, \text{ if stockpile } s \in S \text{ is in front of stockpile } k \in S, \\ 0, \text{ otherwise.} \end{cases}$$

$e_{sk} = \begin{cases} 1, \text{ if stockpile } s \in S \text{ is below stockpile } k \in S, \\ 0, \text{ otherwise.} \end{cases}$

$f_{sk} = \begin{cases} 1, \text{ if stockpile } s \in S \text{ is above stockpile } k \in S, \\ 0, \text{ otherwise.} \end{cases}$

$traveledDist_s$: Rectilinear distance traveled by stockpile $s \in S$.

Model

$$minimize \sum_s ct * traveledDist_s + \sum_s cw * t_s \qquad (A0)$$

$Subject\ to$:

$$x_s + length_s * o_s + width_s * (1 - o_s) \leq x_k + (1 - a_{sk}) * L \qquad \forall s, k \in S : s \neq k \quad (A1)$$

$$x_k + length_k * o_k + width_k * (1 - o_k) \leq x_s + (1 - b_{sk}) * L \qquad \forall s, k \in S : s \neq k \quad (A2)$$

$$y_s + length_s * o_s + width_s * (1 - o_s) \leq y_k + (1 - c_{sk}) * W \qquad \forall s, k \in S : s \neq k \quad (A3)$$

$$y_k + length_k * o_k + width_k * (1 - o_k) \leq y_s + (1 - d_{sk}) * W \qquad \forall s, k \in S : s \neq k \quad (A4)$$

$$arrTime_s \leq t_k + (1 - e_{sk}) * T \qquad \forall s, k \in S : s \neq k \quad (A5)$$

$$arrTime_k \leq t_s + (1 - f_{sk}) * T \qquad \forall s, k \in S : s \neq k \quad (A6)$$

$$a_{sk} + b_{sk} + c_{sk} + d_{sk} + e_{sk} + f_{sk} = 1 \qquad \forall s, k \in S : s \neq k \quad (A7)$$

$$a_{ss} + b_{ss} + c_{ss} + d_{ss} + e_{ss} + f_{ss} = 0 \qquad \forall s, k \in S : s \neq k \quad (A8)$$

$$x_s + length_s * o_s + width_s * (1 - o_s) \leq L \qquad \forall s \in S \quad (A9)$$

$$y_s + width_s * o_s + length_s * (1 - o_s) \leq W \qquad \forall s \in S \quad (A10)$$

$$t_s + storageTime_s \leq T \qquad \forall s \in S \quad (A11)$$

$$t_s \geq nomTime_s \qquad \forall s \in S \quad (A12)$$

$$t_s + storageTime_s \leq arrTime_s \qquad \forall s \in S \quad (A13)$$

$$traveledDist_s \geq |(ConnectX - x_s) + (ConnectY - y_s)| \qquad \forall s \in S \quad (A14)$$

$$x_s, y_s, t_s \geq 0 \qquad \forall s \in S \quad (A15)$$

$$a_{sk}, b_{sk}, c_{sk}, d_{sk}, e_{sk}, f_{sk} \in \{0,1\} \qquad \forall s, k \in S \quad (A16)$$

Our objective is to minimize the distance to the quay side for all stockpiles and minimize the total idle time of stockpiles. Since these two quantities are different in units, (i.e. distance vs. time) we associate these quantities with their respective costs and obtain a single monetary objective. Therefore, the objective function (A0) minimizes the cost of assigning a location to a stockpile where the assignment costs are calculated regarding to the cost of transferring each stockpile to the quay and the dwelling cost of stockpiles. Constraint sets (A1) – (A4) are non-overlapping constraints. They ensure that none of the stockpiles are assigned to on top of each other at the stockyard. Constraint sets (A5) and (A6) restrict building stockpiles that overlaps in time. Constraint set (A7) and (A8) establish the relationship between two stockpiles. Constraint sets (A9), (A10) and (A11) prevent solutions which reach out of the boundaries of the stockyard and time period. Constraint sets (A12) and (A13) ensure that processing of a stockpile is between the arrival and the departure time of the vessel. Constraint set (A14) provides the travel time of the rectilinear distance from port to a location at the stockyard as the cost of storing the pile on that location. We can generalize the model by adding this constraint set to the model, where $ConnectX$ and $ConnectY$ are the coordinates of the connection points of conveyor belts to the ship loaders on the X and Y axis, respectively. Lastly, remaining constraints define the range of each decision variable used in the model.

Linearization of constraint set (A14)

Constraint set (A14):

$$traveledDist_s \geq |(ConnectX - x_s) + (ConnectY - y_s)| \qquad \forall s \in S$$

calculates the rectilinear distance traveled for each stockpile from its storage location to the ship loader. To linearize (A14) the absolute value should be expressed in another way.

Thus, if the following constraint sets are replaced with (A14), the CSA model will be linear.

$$traveledDist_s \geq (ConnectX - x_s) + (ConnectY - y_s) \qquad \forall s \in S \quad \text{(A14a)}$$
$$traveledDist_s \geq (ConnectX - x_s) + (ConnectY + y_s) \qquad \forall s \in S \quad \text{(A14b)}$$
$$traveledDist_s \geq (ConnectX + x_s) + (ConnectY - y_s) \qquad \forall s \in S \quad \text{(A14c)}$$
$$traveledDist_s \geq (ConnectX + x_s) + (ConnectY + y_s) \qquad \forall s \in S \quad \text{(A14d)}$$

Constraint sets (A14a), (A14b), (A14c) and (A14d) ensure that the traveled distance is greater than or equal to the sum of distances between the coordinates of the connection points and coordinates of the stockpile on the yard.

One of the strengths of this model is the flexibility to decide the orientation of the stockpile. However, along with this flexibility, the model performs slower when compared to other models proposed in the following sections of this chapter. Different than the forthcoming models, the distance traveled by each stockpile represents the rectilinear distance from bottom-left corner of each stockpile to a ship loader connection point on the quay side. Thus, a better representation of the real life problem of yard assignment is achieved.

The size of CSA model in terms of the number of constraints, and the number of binary and continuous variables depending on the system parameters is as follows:

Assume that n is the number of stockpiles. Therefore, number of constraints is $(9n^2 + 3n)$, number of binary variables is $(n^2 - n)$ and number of continuous variables is $(3n)$.

4.2 Continuous Yard Allocation Model in the Presence of Pads (CSAP)

As a second step, in addition to the assumptions made in the first model, we presume that there are pads at the stockyard where stockpiles can only be placed within the boundaries of them. In other words, pads determine the widths of these piles and we can only decide the location of a pile along a path. The second and third models are based on this assumption. Thus, one dimension which was width can be omitted in these models. The new version of the problem resembles to two-dimensional packing problems. Hence, the constraints of the following models are similar to the two-dimensional packing models' constraints.

Most of the assumptions are the same as CSAP except there are conveyor belts which were built at the stockyard. These conveyor belts divide the stockyard into smaller parts which are called pads. Therefore, the width of each stockpile is predetermined by the width of the pads. Hence, the decision space of this model can be limited to the location of each pile along the length of each pad and the start time of building that stockpile in a continuous space.

Since most of the bulk port terminals have pads on their stockyards, this assumption makes the model more realistic. An overview of a stockyard with pads can be seen in Figure 9.

In this section, first, the sets, parameters, the decision variables are defined. Secondly, the continuous stockyard allocation model is presented. Finally, the objective function of the model and the constraint sets are explained.

CSAP

Sets and Parameters

S	Set of stockpiles.
P	Set of pads.
L	Length of the stockyard.
T	Maximum time period.

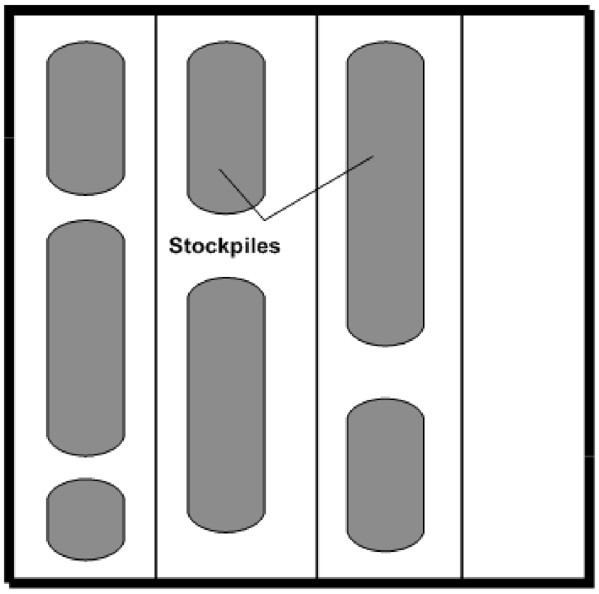

Figure 9. An overview of a stockyard with pads

ct	Cost of transferring a stockpile per second.
cw	Cost of dwelling a stockpile per hour.
at_s	Arrival time of stockpile $s \in S$.
dt_s	Departure time of vessel that stockpile $s \in S$ will be uploaded into.
l_s	Length of stockpile $s \in S$.
p_s	Processing time of stockpile $s \in S$.

Decision Variables

X_s	Position of stockpile $s \in S$ on the x-axis of a pad.
T_s	Start time of building stockpile $s \in S$.

$$Y_{si} = \begin{cases} 1, & \text{if stockpile } s \in S \text{ is on pad } i \in P, \\ 0, & \text{otherwise.} \end{cases}$$

$$A_{sk} = \begin{cases} 1, \text{if stockpile } s \in S \text{ is on the left hand side of } k \in S \text{ and they are on the same pad,} \\ 0, \text{otherwise.} \end{cases}$$

$$B_{sk} = \begin{cases} 1, \text{if stockpile } s \in S \text{ and } k \in S \text{ are on the same pad and } k \text{ is stored after } s, \\ 0, \text{otherwise.} \end{cases}$$

Model

$$minimize \sum_{s} ct * X_s - \sum_{s}\sum_{i} ct * (i-1)\, L\, Y_{si} + \sum_{s} cw * T_s \qquad (B0)$$

$Subject\ to$:

$$X_s + l_s \le X_k + |P|L(1 - A_{sk}) \qquad \forall\, s,k \in S: s \ne k \quad (B1)$$

$$dt_s \le T_k + T(1 - B_{sk}) \qquad \forall\, s,k \in S: s \ne k \quad (B2)$$

$$T_s + p_s \le dt_s \qquad \forall\, s \in S\ (B3)$$

$$A_{sk} + B_{sk} + A_{ks} + B_{ks} = 1 \qquad \forall\, s,k \in S: s \ne k \quad (B4)$$

$$A_{ss} + B_{ss} = 0 \qquad \forall\, s,k \in S: s \ne k \quad (B5)$$

$$X_s + l_s \le \sum_{i} i\, L\, Y_{si} \qquad \forall\, s \in S \quad (B6)$$

$$X_s \ge \sum_{i} (i-1)L\, Y_{si} \qquad \forall\, s \in S \quad (B7)$$

$$T_s + p_s \le T \qquad \forall\, s \in S \quad (B8)$$

$$T_s \ge at_s \qquad \forall\, s \in S \quad (B9)$$

$$\sum_{i} Y_{si} = 1 \qquad \forall\, s \in S \quad (B10)$$

$$X_s \ge 0 \qquad \forall\, s \in S \quad (B11)$$

$$Y_{si} \in \{0,1\} \qquad \forall\, s \in S, \quad \forall\, i \in P\ (B12)$$

$$T_s \ge 0 \qquad \forall\, s \in S \quad (B13)$$

$$A_{sk} \in \{0,1\} \qquad \forall\, s,k \in S: s \neq k \quad (B14)$$

$$B_{sk} \in \{0,1\} \qquad \forall\, s,k \in S: s \neq k \quad (B15)$$

Similar to the CSA model, there are still two objectives in this model that minimize the total cost of distance travelled at the stockyard for all stockpiles and minimize the total cost of dwelling stockpiles. The objective function (B0) minimizes the cost of assigning a location to a pad where the assignment costs are calculated regarding the cost of distance traveled from the location of a pile to the ship loader and the cost of dwelling a stockpile. Constraint sets (B1) are non-overlapping constraints. They ensure that none of the stockpiles are assigned to on top of each other at the stockyard. Since the widths of stockpiles are determined by pads, we can reduce one-dimension in this model when compared to CSA. Constraint sets (A1) - (A4), and (A9) and (A10) corresponds to sets (B1) together with (B6). Similar to constraint sets (A5), (A6) and (A11) - (A13) in CSA, constraint set (B2) restricts building stockpiles that overlaps in time. Constraint set (B3) ensures that storing a stockpile is completed before the corresponding vessel of that load departs. Constraint sets (B4) and (B5) establish the relationship between two stockpiles. Constraint sets (B6) and (B7) prevent solutions which reach out of the boundaries of the stockyard and time period. Constraint sets (B8) and (B9) ensure that processing of a stockpile is between the nomination time of the corresponding vessel and the planning time horizon. Constraint set (B10) ensures that each stockpile must be placed onto exactly one pad. Lastly, remaining constraints define the range of each decision variable.

Since most of the bulk port terminals have pads on their stockyards, this model may be preferred when compared to the CSA model.

The size of CSAP model in terms of the number of constraints, and the number of binary and continuous variables depending on the system parameters is as follows:

Assume that n is the number of stockpiles and p is the number of pads. Therefore, number of constraints is $(6n^2 + 2n + np)$, number of binary variables is $(2n^2 - 2n + np)$ and number of continuous variables is $(2n)$.

4.3 Discrete Stockyard Allocation Model in the Presence of Pads (DSAP)

In addition to the continuous models which are presented in Section 4.2 and 4.3, a discretized stockyard assignment model will be provided in this section. The assumption of having the conveyor belts at the stockyard is also valid in DSAP model. Additionally, the stockyard is assumed to be consisting of small squares which will be allocated to the piles with respect to their areas. The discretized stockyard can be seen in Figure 10. The preliminary results of CSA and CSAP indicate that for some data sets the models may not able to obtain an upper bound within a reasonable time. Therefore, we propose a time indexed model where we can impose the restricted times and spaces for each stockpile.

In this section, first, the sets, parameters, the decision variables are defined. Secondly, the continuous stockyard allocation model is presented. Finally, the objective function of the model and the constraint sets are explained.

DSAP

Sets and Parameters

I	Set of stockpiles.
P	Set of pads.
L	Set of locations on a pad.
T	Set of time points on a planning horizon.
ct	Cost of transferring a stockpile per second.

cw	Cost of dwelling a stockpile per hour
a_i	Arrival time of stockpile $i \in I$.
d_i	Departure time of vessel that stockpile $i \in I$ will be uploaded into.
l_i	Length of stockpile $i \in I$.
p_i	Processing time of stockpile $i \in I$.
L^0	Maximum length of the stockyard.
T^0	Maximum planning horizon.

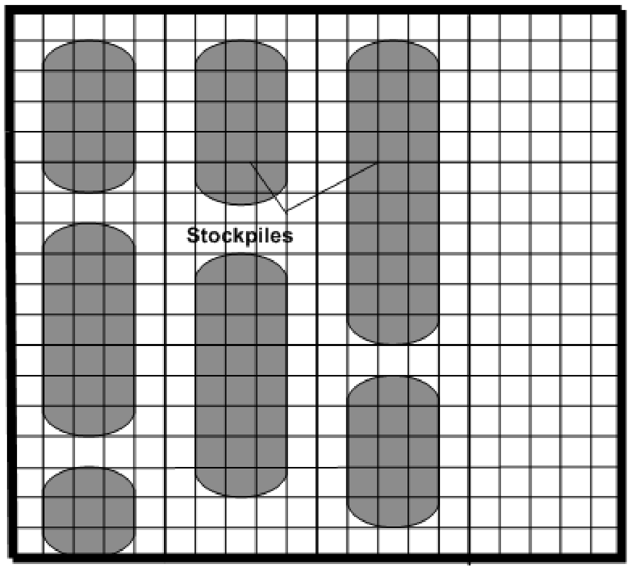

Figure 10. An example of discrete stockyard at time t

Decision Variable

$$S_{ixtp} = \begin{cases} 1, \text{ if stockpile } i \text{ is assigned to position } x \text{ at time } t \text{ on pad } p, \forall i \in I, \forall x \in L, \forall t \in T, \forall p \in P \\ 0, \text{ otherwise.} \end{cases}$$

Model

$$minimize \sum_{i}\sum_{x}\sum_{t}\sum_{p}(ct * x + cw * t)\, S_{ixtp} \qquad (C0)$$

$Subject\ to$:

$$\sum_{x=1}^{L^0-l_i+1}\ \sum_{t=a_i}^{d_i-p_i}\ \sum_{p} S_{ixtp} = 1 \qquad \forall i \in I\ \ (C1)$$

$$\sum_{\substack{x=\min\{L^0,\ L^0-l_i+2\}\\ :\, l_i>1}}^{L^0}\ \sum_{t=1}^{T^0}\ \sum_{p=1}^{|P|} S_{ixtp} + \sum_{x=1}^{L^0}\ \sum_{\substack{t=1\\ a_i>1}}^{a_i-1}\ \sum_{p=1}^{|P|} S_{ixtp}$$
$$+ \sum_{x=1}^{L^0}\ \sum_{\substack{t=\min\{T^0,\ d_i-p_i+1\}\\ :\, p_i>1}}^{T^0}\ \sum_{p=1}^{|P|} S_{ixtp} = 0 \qquad \forall i \in I\ (C2)$$

$$\sum_{i=1}^{|I|}\ \sum_{k=\max\{x-l_i+1,\ 1\}}^{x}\ \sum_{l=\max\{t-p_i+1,\ 1\}}^{t} S_{iklp} \leq 1 \qquad \forall x \in L, \forall t \in T, \forall p \in P\ \ (C3)$$

$$S_{ixtp} \in \{1,0\} \qquad \forall i \in I,\quad \forall x \in L,\quad \forall t \in T,\quad \forall p \in P\ (C4)$$

In DSAP model the objective function (C0), similar to the previous models, minimizes the total cost of transferring stockpiles at the stockyard and the total cost of dwelling stockpiles. Constraint set (C1) determines the boundaries of the stockyard as well as available time periods to build a stockpile and ensures that each stockpile must be placed onto exactly one pad. Since we know the restricted space and time, over all stockpiles we can ensure that no stockpile is placed out of the boundaries. Therefore, similar to constraint sets (B6) - (B8) in CSAP model, constraint set (C2) restricts solutions which lie along the outside of boundaries of the stockyard and available time period. Hence, this constraint reduces the search space of the model. Constraint set (C3) contains non-overlapping conditions. Constraint set (C3) ensures that none of the stockpiles are assigned to on top of each other at the stockyard and also it restricts building stockpiles that overlaps in time. Additionally, this constraint set ensures that building a stockpile is completed before the corresponding vessel of that load arrives. Constraint sets (A1) – (A6) in CSA and sets (B1), (B2), (B6) and (B7) in CSAP corresponds to (C3). Lastly, remaining constraints define the range of the decision variables.

The size of DSAP model in terms of the number of constraints, and the number of binary and continuous variables depending on the system parameters is as follows:

Assume that n is the number of stockpiles, l is the number of locations on a pad, t is the number of time periods in a planning horizon and p is the number of pads. Therefore, number of constraints is $(nltp + ltp + 2n)$ and number of binary variables is $(nltp)$.

4.4 Comparison of the proposed models

One of the advantages of CSA model is that, the orientation of the stockpiles can be determined and CSA is suitable to solve YAP in stockyards where there is no pads. On the other hand, with the growth in world maritime transportation, the need for larger stockyards is increased. In most of the large-sized stockyards there are conveyor belts, thus there are pads. Therefore, CSAP and DSAP models are proposed to meet the physical conditions of most of the stockyards.

Chapter 5

HYBRIDIZED TABU SEARCH AND VARIABLE NEIGHBORHOOD SEARCH

Some of the physical constraints of YAP, such as restricting placing stockpiles on top of each other, are achieved through non-overlapping constraints of multi-dimensional packing problems (MDPP). Since MDPP are NP-hard, an MILP models may not find the optimal solutions for large instances in a reasonable time.

In this chapter, a tabu search (YATS) and a tabu search hybridized with variable neighborhood search (YATS_VNS) algorithms which uses a bottom-left-fill (BLF) like heuristic are proposed. Detailed explanations of the algorithms and the components of proposed YATS and YATS_VNS are provided in Section 5.1 and Section 5.2, respectively. Hence, the chosen representation of the problem, proposed neighborhood definitions, move operators, the tabu rule, tabu tenure, the stopping criterion and the aspiration criterion are discussed. The pseudocode of the heuristic algorithm is presented in order to interpret the proposed representation. Along with these definitions, the feasibility check mechanism of the algorithm is explained in Section 5.1.

5.1 Implementation of the TS

In this section the implementation of the TS to solve the yard allocation problem is discussed. Initially, the solution representation and how to obtain the initial solution is explained. Secondly, the move operator and the corresponding neighborhood structure are provided and the search procedure is explained. Then, the tabu list and the tabu tenure, and the stopping and the aspiration criteria are established. Finally, a bottom-left-fill like heuristic (YA_BLFLH) which is the decoding and encoding routine of the algorithm and the feasibility check procedure are presented.

Solution Representation:

According to Hopper and Turton's (2000, 2001) categorization which is provided in Chapter 3, some metaheuristic algorithms represent the solution space in a more basic form. For example, with respect to the packing problems, metaheuristics can be only used to determine the sequence of packing. Then, using a decoding-encoding routine, the allocation of items on the object is obtained.

Since a complex encoding and decoding routine is adopted in the proposed algorithm, it is sufficient to express the solution as a permutation of pile numbers. Moreover, algorithms like BLF take the sequence of the items as an input to construct the final solution. Therefore, in YATS permutation of stockpile numbers such as $[s_1, s_2, \dots, s_n]$ is used as the solution representation. The order of the piles represents the placing order in which the YA_BLFLH utilizes. For example, if five stockpiles will be located at the stockyard and if the represented solution is [3, 1, 4, 5, 2] then, YA_BLFLH first places pile 3 then pile 1 and continues with the order.

Initial Solution:

Different starting points may result in different best solutions. In order to select a better initial solution, two different approaches are adopted. In some of the applications of packing heuristics, items may be sorted according to their dimensions. One of the approaches that are presented is to sort the items according to their arrival times. For example, if piles arrive in the order of 2, 3, 1, 4 and 5 then the initial solution representation is [2, 3, 1, 4, 5]. Additionally, random starting solutions can be accepted. First, the piles are randomly sorted and then a placing procedure can be applied. The preliminary results are discussed in Chapter 6 to indicate the robustness of the algorithms in terms of the initial solution.

Neighborhood Structure:

Since a complex decoding routine is adopted, for simplicity of the algorithm a swap operator is selected to create the neighborhood. This operator interchanges the order of two selected piles. For example, assume that the current solution is [3, 4, 5, 2, 1]. First, two random integers are generated between 1 and the number of piles; assume that 2 and 5 are generated. Then, the 2nd and the 5th piles are swapped such that the new order becomes [3, 1, 5, 2, 4].

Neighborhood Search Procedure:

In order to move to a new solution, the following search procedure is applied. If s is the current solution, the admissible (non-tabu or allowed by aspiration criterion) neighborhood of s, $\overline{N}(s)$ is searched systematically to find a better solution s'. In order to explore the different solutions in the search space, instead of visiting all solutions in $\overline{N}(s)$, the number of replications for one solution is predetermined and this number is referred to as $nrep$. Therefore, the diversification of the search is increased whereas the intensification is sacrificed. In this thesis a steepest descent (best improvement) strategy is selected to search the neighborhood and accept the new solutions. The pseudocode of the steepest descent strategy with swap operator is provided in Figure 11. The details of YA_BLFLH algorithm used in Step 4 of the search are explained in latter parts of Section 5.1.

Search and Acceptance Procedure: Steepest Descent with Swap

```
 1: Input: $nrep$: number of swaps in each iteration
           $f(s)$: current objective value
 2: for 1 to $nrep$
 3:     $s' =$ Swap two items in the order
 4:     Run YA_BLFLH
 5:     Calculate $f(s')$
 6:     if $f(s') < f(s)$ then
 7:         Update $s = s'$ and $f(s) = f(s')$
 8:     end if
 9: end for
10: Return s
```

Figure 11. Search and acceptance procedure: Steepest Descent with Swap

Number of Replications

In this procedure, the neighborhood is searched for a predetermined number of replications. Randomly selected two stockpiles in the sequence are swapped and if an improvement occurs, the solution and the objective values are recorded. The search continues until the number of replications, and the best obtained solution are returned. Preliminary tests are conducted in order to decide the number of swap operations performed per iteration. These preliminary computational experiments are presented in Section 6.3 in detail.

Tabu List and Tabu Tenure:

Tabu lists can consist of whole solution, part of a solution or an attribute of a solution. In YATS, the pile numbers swapped are recorded in the tabu list so that in the following iteration, these two piles cannot be swapped. In the previous example, 2nd and 5th items which correspond to 4th and 1st piles were swapped. Therefore, 4th and 1st piles are kept in the tabu list. In the following iterations, either 4th or 1st piles cannot be swapped for a number of iterations. The number of iterations for which swapping is forbidden is known as tabu tenure. Tabu tenure is decided to be 5 after some preliminary computational experiments. The details of these preliminary computational experiments are given in Section 6.3.

Aspiration Criterion:

In some cases, a solution is accepted even though it is in the tabu list. According to a predetermined rule, the tabu status of that solution is ignored and the solution becomes the new current solution. The aforethought rule is referred to as aspiration criterion. In this algorithm, best solution so far is chosen as an aspiration criterion. In other words, if the best solution so far is achieved by swapping one of the piles in the tabu list then this swap can occur.

Stopping Criterion:

The algorithm is designed to terminate when the objective function cannot be improved for a certain number of iterations (say, T_1). In other words, the stopping criterion is chosen to be number of iterations without improvement. Different T_1 values are examined to avoid early convergence or waiting too long even after the acceptable solutions. On the other hand, in order to avoid a possible infinite loop in case the algorithm fails to find a feasible solution, a threshold value (say, T_2) that limits the maximum number of iterations is implemented. Thus, the algorithm stops if it can find a feasible solution and cannot improve it for T_1 iterations, or when it reaches T_2 iterations. In order not to cause early convergence, T_2 is set to a high number such as 5000.

A bottom-left-fill like heuristic for yard allocation (YA_BLFLH):

A bottom-left-fill like heuristic (YA_BLFLH) algorithm is presented which is used to decode the solution representation of YATS. YA_BLFLH tries to find a good feasible place for the bottom left corner of each stockpile. Starting from the (0, 0) coordinate of the first pad, all piles are placed regarding to their arrival times and the availability of the pad. For example, assume that the sequence of piles is [1, 5, 9, 2, 6, 4, …] and there are two pads at the stockyard. In Figure 12, the x-axis represents planning time horizon and intervals are given in terms of days, whereas on the y-axis the length of pads is shown. Each box corresponds to a stockpile where the x-dimension is the storage time of the stockpile and y-dimension is the length of the stockpile. The arrival time of a stockpile is denoted by letter A and the stockpile number, such that A1 is the arrival time of stockpile 1. Additionally, letter C denotes a corner point. Moreover, the schedule of stockpiles 1, 5, 9, 2 and 6 can be seen in Figure 12. Thus, YA_BLFLH will assign stockpile 4 to a place after evaluating admissible (feasible points) corner points in each pad, individually. For example, in Figure 12, C2-C6, C10, C11, C15-C18 are admissible corner points for stockpile 4. YA_BLFLH choses among these points to place the bottom-left corner of pile 4 while minimizing the cost of distance travelled and the cost of dwelling times over all stockpiles. Moreover, the pseudocode of the proposed heuristic YA_BLFLH can be seen in Figure 13.

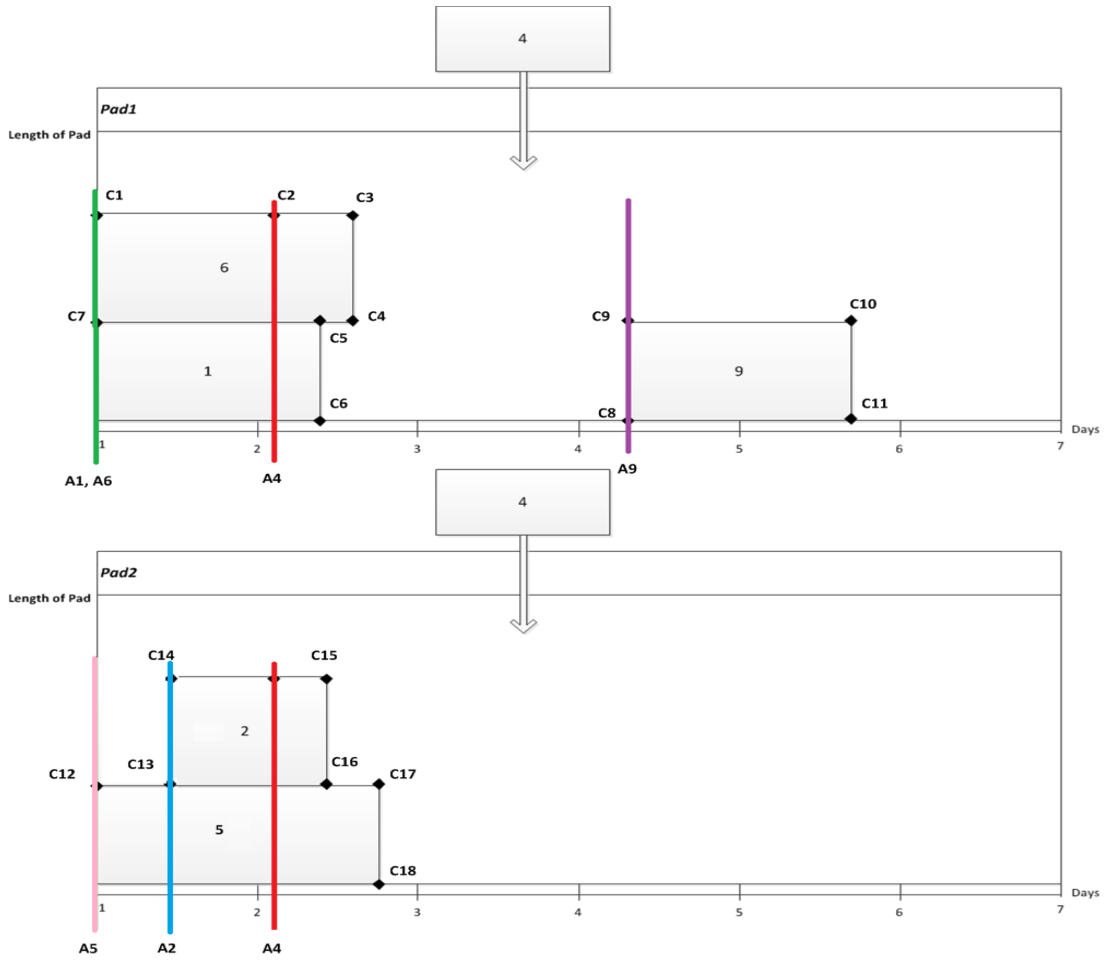

Figure 12. An instance of YA_BLFLH algorithm

Notation

I: Set of stockpiles.

$\bar{I}$: Set of stockpiles which are not placed at the stockyard yet.

P: Set of pads.

XT: Set of corner points and start time of building each stockpile on the yard.

$\varphi(i, x, t)$: the partial objective value of s when placing pile i to the yard.

$xt^*(i)$: the best known (x, t) coordinates for pile i.

Algorithm: YA_BLFLH

```
 1: Input: $s, I, P, \bar{I}$
 2: while $\bar{I} \neq \emptyset$ do
 3:     for 1 to $|P|$
 4:         for $\forall (x, t) \in XT$
 5:             if placing $i \in \bar{I}$ on $(x, t)$ is feasible then
 6:                 if $\varphi(i, x, t) < \varphi(i, xt^*(i))$ then
 7:                     Place stockpile $i$ on $(x, t)$
 8:                     Update $xt^*(i)$ and $\varphi(i, xt^*(i))$
 9:                 end if
10:             else
11:                 Place stockpile $i$ on a predetermined infeasible location
12:                 Update $xt^*(i)$ and $\varphi(i, xt^*(i)) = +\infty$
13:             end if
14:         end for
15:     end for
16: end while
```

Figure 13. A bottom-left-fill-like heuristic to solve YAP

The YATS algorithm utilizes YA_BLFLH algorithm and the components of TS to solve yard allocation problem. The pseudocode of YATS is provided in Figure 14.

Notation

s_0: the initial solution.

s: the current solution.

s^*: the best known solution.

$f(s)$: the objective value of s.

$N(s)$: the neighborhood of s.

$\overline{N}(s)$: the admissible (non-tabu or allowed by aspiration criterion) neighborhood of s.

$\bar{s}$: the best solution $\in \overline{N}(s)$.

$nrep$: the number of search replications in one iteration.

Algorithm: YATS

```
 1: Initialize: $f(s) = +\infty, TabuTenure, TabuLists = \emptyset, nrep, maxIter$
 2: Generate: $s_0$: sort piles according to arrival times
 3: $s \leftarrow s_0$
 4: **Run** YA_BLFLH
 5: $f(s) \leftarrow f(s_0)$
 6: **while** termination criterion is not met **do**
 7:     **for** 1 to $nrep$
 8:         Construct $N(s)$
 9:         Construct $\overline{N}(s)$
10:         Find $\bar{s} \in \overline{N}(s)$
11:         **Run** YA_BLFLH
12:         **if** $f(\bar{s}) < f(s)$ **then**
13:             Update $s^*$ and $f(s^*)$
14:             Update the $TabuLists$
15:         **end if**
16:     **end for**
17: **end while**
```

Figure 14. Pseudocode of the YATS Algorithm

5.2 Implementation of the hybridized algorithm YATS_VNS

The solution representation, initial solution, neighborhood structure, tabu list and tabu tenure, search procedure, aspiration criterion, stopping criterion, feasibility check mechanism and decoding routine YA_BLFLH components of YATS which are provided in Section 5.1 are also applied for the hybridized heuristic referred to as YATS_VNS. Moreover, in this section, first a further neighborhood structure and a new tabu list are provided. Then, the pseudocode of the YATS_VNS algorithm is presented.

Additional Neighborhood Structure

In addition to the swap operator, in this section insertion operator is considered.
Insertion: This operator achieves a new solution by inserting a pile between two other piles in the permutation. For example, assume that the current solution is [3, 5, 2, 4, 1]. Two random integers are generated between 1 and the number of piles; assume that 1 and 3 are generated. First generated value is the number which is inserted; second number determines the insertion location. In our example, the 1st pile will be inserted next to 5th such that the new order becomes [5, 2, 3, 4, 1].

Additional Tabu List

In the first tabu list, the numbers of piles which are swapped are kept. In addition to the first list, in the second list the number of pile which will be inserted as well as numbers of piles next to it are stored. From the previous example, there is [5, 2, 3, 4, 1] solution. If this solution results in better objective value and is accepted as the new solution, then, 3rd, 2nd and 4th piles are recorded in the tabu list and cannot be moved for the next iterations. The reason keeping all these piles is that, if that order results in a good solution, it can be saved for some iterations. Therefore, during a predetermined number of iterations, these three piles cannot be moved unless an aspiration criterion occurs.

The YATS_VNS utilizes YA_BLFLH algorithm and the components of TS as well as VNS to solve yard allocation problem. The pseudocode of YATS_VNS is provided in Figure 15.

Pseudocode of YATS_VNS

Notation

I: Set of stockpiles.

P: Set of pads.

$N_k,\ (k = 1,2)$: set of preselected neighborhood structures.

s_0: the initial solution.

s: the current solution.

s^*: the best known solution.

$f(s)$: the objective value of s.

$N_k(s)$: the k^{th} neighborhood of s.

$\overline{N}_k(s)$: the admissible solutions of the (non-tabu or allowed by aspiration criterion) k^{th} neighborhood of s.

$\bar{s}$: the best solution $\in \overline{N}_k(s)$.

$nrep$: the number of search replications in one iteration.

$maxIter$: the maximum number of iterations.

Algorithm: YATS_VNS

```
1:  Initialize: $f(s) = +\infty, TabuTenure, TabuLists = \emptyset, nrep, maxIter$
2:  Generate: $s_0$: sort piles according to arrival times
3:  $s \leftarrow s_0$
4:  Run YA_BLFLH
5:  $f(s) \leftarrow f(s_0)$
6:  while termination criterion is not met do
7:      for 1 to $nrep$
8:          Set $k \leftarrow 1$
9:          Construct $N_k(s)$
10:         Find $\bar{s} \in \overline{N}_k(s)$
11:         Run YA_BLFLH
12:         if $f(\bar{s}) < f(s)$ then
13:             Update $s^*$ and $f(s^*)$
14:             Update the $TabuLists$
15:             Continue search with $k \leftarrow 1$
16:         else
17:             $k \leftarrow k + 1$
18:         end if
19:     end for
20: end while
```

Figure 15. Pseudocode of the Hybridized Tabu Search and Variable Neighborhood Search Algorithm

Chapter 6

COMPUTATIONAL EXPERIMENTS

The computational experiments are conducted in order to test the success of both the mathematical models and the metaheuristics. We analyze the performance of both the mathematical models and the metaheuristic algorithm with four sets of generated data including instances based on real life data. In this chapter, first, the design of the computational experiments is provided. Then, the preliminary tests for the parameter settings of metaheuristics are presented. Finally, the results of the models are analyzed and the comparisons are evaluated.

6.1 Design of the Computational Experiments

To test and compare the performance of the models, the same instances are solved with all models. Since one of the mathematical models has a discrete space, to be able to solve the same instances with all models concurrently, the parameters are generated consisting of only integer numbers.

In CSAP and DSAP there are pads at the stockyard, therefore no stockpile can be placed on two pads such that some proportion of the pile is on one pad whereas the remaining part is on the other pad. In other words, a stockpile can only be built on one pad such that the width of a stockpile must be equal to the width of that pad. In order to be able to compare these models to the CSA which does not have a pad structure, the widths of stockpiles are set to the width of pads which is 50 meters.

With each model, the same scenarios and for each scenario 5 random instances are solved using CPLEX 12.5 with a desktop having 3.40 GHz processor and 8 GB RAM. The running time limit for CPLEX is set to half an hour for sets A, B and D and one hour for set C.

Additionally, stockyard dimensions and arrival times of vessels are taken from a port in Australia. Although storage times and dimensions of stockpiles are randomly generated, it is important to emphasize that they are agreeable with real life data. Therefore, scenarios which represent real life values are tested. Four different data sets are used in order to analyze the performance of the algorithms. The generated scenarios can be seen in Table 1.

Table 1. Data Set

	Set A	**Set B**	**Set C***	**Set D**
S (number of stockpiles)	12, 14, 16	16, 18, 20, 24, 27, 29	16, 18, 20, 24, 27, 29	48, 54, 58
L (length of the stockyard)	10, 16	10, 24	24	20
Stockpile length intervals (100 mts)	[3,5], [1,5], [1,3]	[3,5], [1,5], [1,3], [8,10], [2,10], [2,6]	[8,10], [2,10], [4,8], 6	[8,10], [2,10], [4,8], 6
Stockpile storage times (hrs)	[12,24]	[24,48]	[24,48]	[36,48]
Cost of dwelling (per hr)	$100	$100	$100	$100
Cost of transferring (per mt)	$1	$1	$1	$1
T (hrs)	120	168, 336	168	336, 504

* The arrival and departure times of stockpiles are assumed to be 1 and end of the planning horizon (T), respectively

The explanation of tables, calculation of performance measures and the representation of the scenarios are as follows:

In Tables 6-19, under the Scenario column, the scenarios are given as $\mathrm{N\_L\_(l_1\_l_2)\_X}$ where N is the number of stockpiles, L is the length of the stockyard, $\mathrm{l_1\_l_2}$ are the length interval of stockpiles. Finally, X denotes the time length between arrival of a stockpile and the departure of the vessel which will be loaded with that stockpile. If an S appears, it means that there is a small gap between arrival of a stockpile and the departure of the vessel; it is 48-60 hours. On the other hand, if an L appears, there can be 48-136 hours. Under the "Status" column, "Opt." column shows how many instances can be solved to optimality within 30 minutes, "Term." column shows how many instances are terminated due to time limit. In order to measure the performance of the models, the gap between the best obtained solution and the lower bound is calculated for the terminated instances. The formula to calculate the gap is provided in Equation 6.1.

$$\text{Gap} = \frac{(UB-LB)}{LB} \times 100 \tag{6.1}$$

The minimum, the maximum and the average gaps are calculated for every instance in each scenario over 5 replicas. Root gaps as well as gaps after 30 minutes are provided in the tables. If there is no gap between the upper and lower bounds, the instance is solved to optimality and (-) denotes zero gap.

6.2 Preliminary Tests for Parameter Settings for YATS_VNS

In Chapter 5 we present the components of the TS and the VNS algorithms. We provide different initial solutions, tabu tenure, number of replications and stopping criterion. In order to obtain better results we need to find a good combination of the presented parameters. The preliminary tests are conducted for the selection of the parameters of the YATS_VNS algorithm. We conduct preliminary tests for different scenarios from different data sets. There is no notable difference in behavior between different data sets. Hence for the sake of simplicity, the results of the preliminary tests for 36 instances from Set C are presented.

Initial Solution:

In Table 2, the average percentage deviation and CPU times can be seen. The compared initial solutions are: (1) the sorted numbers according to the arrival times, (2) the randomly ordered piles.

The percentage deviations (PD) of objective values are compared for different initial solutions. PD is calculated according to formula given in (6.2). $BestValue_1$ corresponds to the best obtained value starting with a sequence of sorted stockpiles according to their arrival times. $BestValue_2$ is the best obtained value when the search stars with a random sequence.

$$\text{PD} = \frac{(BestSol_1 - BestSol_2)}{BestSol_2} \tag{6.2}$$

In order to decide initial solution, for different initialization procedures the objective values and the running times of the algorithm is compared. Table 2 shows that if piles are sorted according to their arrival times, then the algorithm provides better objective values when compared to the random initial solution. On the other hand, sorting procedure increases the CPU time approximately 1 second on average. Despite the increase in running time, the better objective values are preferred and sorting according to arrival times is chosen as an initial solution. The detailed results of this computational experiment can be seen in Appendix 1.

Table 2: Average CPU times and percent deviation with different initial solutions for data Set C

	Best Solution with (1)	Best Solution with (2)	
	CPU Secs	CPU Secs	PD %
Average	15.85	14.53	0.32

Tabu Tenure:

The obtained objective values ($HeurValue$) of the preliminary tests are compared with optimal values ($OptimalValue$) and the gap (Gap_{heur}) is calculated according to formula provided in (6.3). The average CPU times and the average gaps are presented in Table 3 and the best results are given in bold.

$$Gap_{heur} = \frac{(HeurValue - OptimalValue)}{OptimalValue} \tag{6.3}$$

Table 3 indicates that the average gaps and the average CPU seconds are very close to each other. Hence, we can conclude that tabu tenure value does not affect the performance of TS very much. Based on the results, we choose tabu tenure as 5. The detailed results are presented in Appendix 3.

Table 3: Average CPU times and average gaps with different tabu tenures for data Set C

	Tabu Tenure 3		Tabu Tenure 5		Tabu Tenure 10	
	CPU Sec	Gap_{heur}	CPU Sec	Gap_{heur}	CPU Sec	Gap_{heur}
Max	35.28	10.76%	**29.78**	**10.19%**	45.06	10.96%
Avg	16.00	3.41%	**15.85**	**3.21%**	17.83	3.31%
Min	3.60	**0.00%**	3.60	**0.00%**	**3.28**	**0.00%**

Number of replications:

The number of solutions visited in an admissible neighborhood is denoted by the number of replications ($nrep$). The results of the preliminary tests are provided in Table 4. In order to decide the number of swap operations performed per iteration the maximum, the average and the minimum of the CPU times and gaps are compared for $nrep = 5$ and $nrep = 15$.

Table 4: Average CPU times and gaps with different number of replications for data Set C

	$\boldsymbol{nrep}$ **5**		$\boldsymbol{nrep}$ **15**	
	CPU Sec	**Gap$_{heur}$**	**CPU Sec**	**Gap$_{heur}$**
Max	**16.75**	12.48%	29.78	**10.19%**
Avg	**5.40**	3.39%	15.85	**3.21%**
Min	**1.16**	0.00%	3.60	**0.00%**

Although 5 replications provide faster results, these results are on the average worse than 15 replications. Additionally, 5 replications may result in an early convergence. Therefore, $nrep = 15$ is chosen for medium- and large-sized instances. On the other hand, for small-sized instances, since the algorithm obtains optimum results in a very short time, $nrep = 5$ is used. For example, to illustrate the behavior of the YATS_VNS with different $nrep$ values, the convergence graphs are provided in Figure 16 and Figure 17. The detailed results of the computational experiments can be seen in Appendix 2. We can conclude that, when $nrep = 15$ we obtain better objective values within less number of iterations when compared to $nrep = 5$.

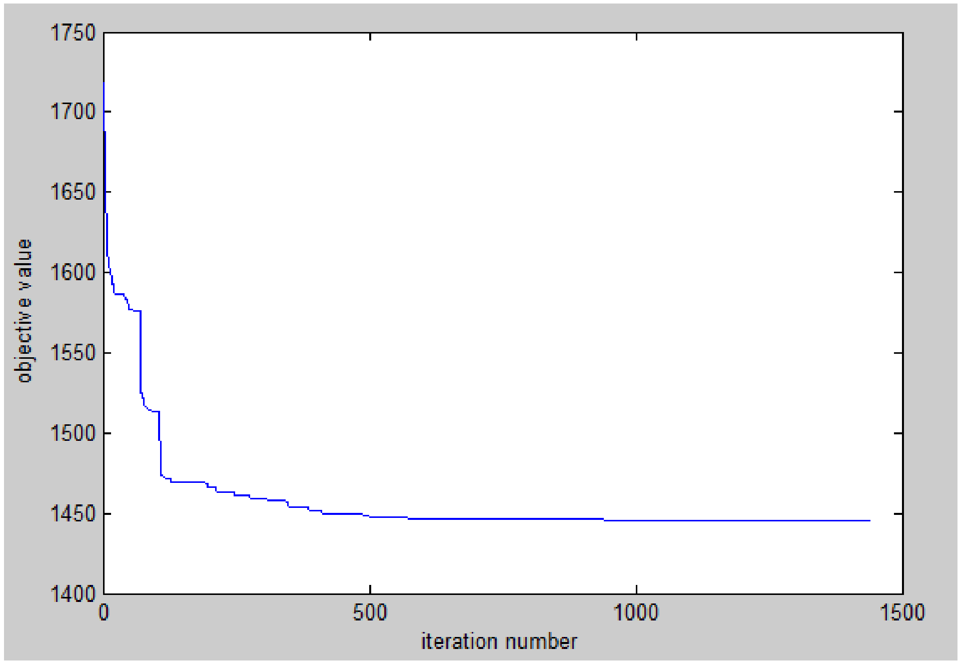

Figure 16. Convergence graph of an instance from Set C where $nrep = 5$

Stopping Criterion:

In Table 5, the results are presented where the number of iterations without improvement is considered as 200, 500 and 1000.

Table 5. Average CPU times and average gaps with different number of iteration for stopping criterion for data Set C

	Stopping 200		Stopping 500		Stopping 1000	
	CPU Sec	**Gap$_{heur}$%**	**CPU Sec**	**Gap$_{heur}$%**	**CPU Sec**	**Gap$_{heur}$%**
Max	**15.44**	14.70	29.78	**11.21**	66.31	11.34
Avg	**5.89**	3.96	13.15	**3.02**	15.67	3.43
Min	**1.61**	0.00	3.60	**0.00**	3.60	0.00

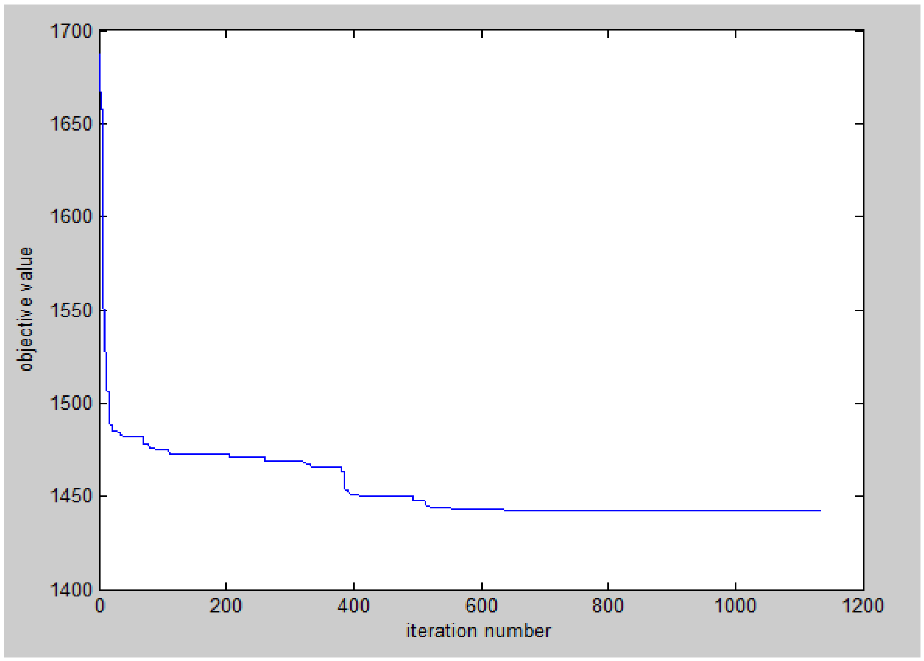

Figure 17. Convergence graph of an instance from Set C where $nrep = 15$

When the stopping criterion is set to 200 iterations without improvement, YATS_VNS reaches the result quickly. Although it is fast, the percent deviation from the optimal value is the greatest with 200 iterations. 500 and 1000 iterations give similar results in terms of gap but sometimes waiting 1000 iterations may not result any improvement and while CPU time increases the gap is not decreasing. Therefore we choose the stopping criterion as 500 iterations. The detailed results of the experiments are given in Appendix 4.

6.3 Results of MILP Models with Instance Set A

The results of the computational experiments of CSA, CSAP and DSAP models are shown in Table 6, Table 7 and Table 8, respectively.

As it can be seen from the results, DSAP model managed to solve all of the instances optimally. On the other hand, CSA can solve 50% of the instances to optimality. For CSAP model, a less number of instances from half of the scenarios had to be terminated after half an hour before the model can obtain the optimal solution.

Results of the models individually show that CSA solved 90 out of 180 instances to optimality. An investigation into the characteristics of the instances which are terminated due to time limit indicates that most of them have 14 and 16 stockpiles. In addition, one important characteristic of the solutions is that although CSA could decide the orientation of the stockpiles, it gave the similar solutions to other problems. In other words, for small-sized stockpiles, even though the model could place the piles parallel to the quay side, in all of the obtained optimal solutions, the piles were placed perpendicular to the quay side as if there are pads. Although, there are instances which CSA cannot solve optimally, Table 6 shows that the maximum gap among all instances is 11.61%. Additionally, in most of the scenarios the maximum gap is less than 5%. Out of 25 scenarios in which there are terminated instances, only 6 scenarios has more than 3% average gap.

CSAP model managed to solve 132 out of 180 instances optimally. Similar to CSA, when there are 14 and 16 stockpiles, the model may not be able to obtain the optimal result within half an hour. Even though one of the instances in specific scenarios cannot be solved to optimality, other instances might be solved very quickly in the same scenarios. For example, the results of 18th, 24th or 30th scenarios in Table 7 indicate that other than one terminated instance, there are instances which are solved within 5 seconds. The average performance with respect to CPU time shows that CSAP model results are better than CSA model results.

Since all of the instances are solved to optimality with DSAP, the optimality gaps are not included in Table 8. In addition, the root gaps for every instance are also zero except the 32nd scenario which has less than 1% for maximum gap. The model spends most of the time solving the root node and when it finds the root bounds, the gap is almost 0. Then the model obtains the optimal solution instantly. Because constraint sets (C1) and (C2) differentiate available and unavailable places for each stockpile at the stockyard, the discrete model can perform faster than others. For DSAP model, the maximum CPU time is less than 10 seconds for every instance in Set A. On the other hand, for scenarios 3, 9 and 12 the average CPU time of CSAP is better than that of DSAP. For the rest of the instances, DSAP outperforms CSAP and CSA.

In addition to the MILP models, these instances are also solved with YATS and YATS_VNS. The results are given in Table 9 and Table 10. Since the instance set A is very small-sized, YATS and YATS_VNS can find the optimal solution in all instances. Moreover, they can obtain optimal solutions almost in the first iterations. Therefore, the performances of the algorithms are tested when the stopping criterion, which is the number of iterations without improvement, is decreased to 10 for this data set. Number of iterations without improvement is set to 10 and the results are presented under the "Stopping criterion (1)" (SC1) column and original parameters under the "Stopping criterion (2)" (SC2) column in the tables. It can be concluded that, although all instances are solved to optimality with SC2, the computation time is approximately 10 times SC1's running time. On the other hand, with SC1, in 7 out of 180 instances the optimal results cannot be achieved. However, the optimality gap is very low on average; hence, the maximum gap is 0.41%.

Results in Table 9 and Table 10 indicate that YATS and YATS_VNS can achieve optimal solutions in average 5-6 seconds. Additionally, if the number of iterations without improvement is decreased, they solve every instance in set A in less than a second without sacrificing optimality in most of the cases.

Table 6: CSA Model Results with Instance Set A

		Status		Root Gap %			Gap %			CPU Time (sec)		
No.	Scenario	Opt.	Term.	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	12_10_(3_5)_S	4	1	100.00	100.00	100.00	0.68	0.14	0.00	1800.58	386.31	22.93
2	12_10_(3_5)_L	5	-	100.00	100.00	100.00	-	-	-	308.87	86.28	2.32
3	12_10_(1_5)_S	5	-	100.00	82.29	49.23	-	-	-	103.40	29.08	3.87
4	12_10_(1_5)_L	5	-	100.00	64.10	8.87	-	-	-	51.06	15.71	1.47
5	12_10_(1_3)_S	4	1	100.00	81.24	6.18	0.37	0.08	0.00	1800.70	383.20	1.75
6	12_10_(1_3)_L	5	-	100.00	65.73	13.79	-	-	-	17.08	6.43	1.92
7	12_16_(6_8)_S	4	1	100.00	88.21	69.12	0.71	0.15	0.01	1800.66	448.11	20.83
8	12_16_(6_8)_L	5	-	100.00	100.00	100.00	-	-	-	284.97	89.64	2.29
9	12_16_(2_6)_S	5	-	100.00	66.24	12.58	-	-	-	17.44	10.10	1.11
10	12_16_(2_6)_L	5	-	100.00	78.89	16.00	-	-	-	72.48	23.31	1.50
11	12_16_(2_4)_S	5	-	100.00	40.51	18.24	-	-	-	61.86	18.59	3.07
12	12_16_(2_4)_L	5	-	100.00	57.86	0.00	-	-	-	355.90	78.81	0.56
13	14_10_(3_5)_S	1	4	100.00	100.00	100.00	8.10	2.90	0.01	1801.53	1488.41	237.31
14	14_10_(3_5)_L	1	4	100.00	100.00	100.00	5.53	2.56	0.01	1801.23	1572.79	660.24
15	14_10_(1_5)_S	3	2	100.00	100.00	100.00	1.84	0.47	0.01	1801.15	774.80	13.56
16	14_10_(1_5)_L	4	1	100.00	100.00	100.00	4.53	0.91	0.01	1800.61	779.93	90.25
17	14_10_(1_3)_S	3	2	100.00	100.00	100.00	2.16	0.60	0.01	1804.74	1104.37	98.31
18	14_10_(1_3)_L	4	1	100.00	100.00	100.00	0.85	0.18	0.00	1800.83	793.59	3.56
19	14_16_(6_8)_S	1	4	100.00	100.00	100.00	8.01	4.40	0.01	1803.03	1625.91	923.31
20	14_16_(6_8)_L	1	4	100.00	100.00	100.00	11.61	4.29	0.01	1801.41	1462.61	108.67
21	14_16_(2_6)_S	4	1	100.00	100.00	100.00	3.69	0.74	0.00	1800.94	512.97	4.67
22	14_16_(2_6)_L	2	3	100.00	100.00	100.00	4.36	2.05	0.01	1801.29	1184.31	126.50
23	14_16_(2_4)_S	1	4	100.00	100.00	100.00	3.93	1.48	0.01	1801.19	1602.45	808.35
24	14_16_(2_4)_L	2	3	100.00	66.57	8.86	3.13	1.06	0.00	1801.36	1091.97	3.92
25	16_10_(3_5)_S	-	5	100.00	100.00	100.00	4.36	3.85	2.48	1801.26	1801.11	1800.75
26	16_10_(3_5)_L	-	5	100.00	100.00	100.00	5.38	4.00	2.04	1802.11	1801.57	1801.32
27	16_10_(1_5)_S	-	5	100.00	100.00	100.00	1.91	0.78	0.24	1801.53	1801.39	1801.25
28	16_10_(1_5)_L	-	5	100.00	100.00	100.00	1.42	0.85	0.32	1801.80	1801.48	1801.19
29	16_10_(1_3)_S	-	5	100.00	100.00	100.00	1.54	0.79	0.33	1801.69	1801.49	1801.36
30	16_10_(1_3)_L	1	4	100.00	100.00	100.00	1.18	0.61	0.01	1801.57	1445.97	24.27
31	16_16_(6_8)_S	-	5	100.00	100.00	100.00	10.75	8.19	4.83	1801.35	1801.14	1800.67
32	16_16_(6_8)_L	-	5	100.00	100.00	100.00	7.55	6.09	3.26	1801.35	1801.26	1801.14
33	16_16_(2_6)_S	-	5	100.00	100.00	100.00	3.46	2.12	0.52	1801.55	1801.39	1801.23
34	16_16_(2_6)_L	2	3	100.00	100.00	100.00	3.24	1.64	0.01	1822.10	1281.80	470.61
35	16_16_(2_4)_S	1	4	100.00	100.00	100.00	2.10	1.40	0.01	1801.36	1447.91	34.95
36	16_16_(2_4)_L	2	3	100.00	100.00	100.00	2.13	0.78	0.01	1801.31	1303.52	544.69

Table 7: CSAP Model Results with Instance Set A

		Status		**Root Gap %**			**Gap %**			**CPU Time (sec)**		
No.	**Scenario**	**Opt.**	**Term.**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
1	12_10_(3_5)_S	5	-	17.01	8.59	4.81	-	-	-	42.51	11.60	2.54
2	12_10_(3_5)_L	5	-	32.94	11.78	3.91	-	-	-	30.42	8.22	1.84
3	12_10_(1_5)_S	5	-	21.00	10.18	2.18	-	-	-	6.94	2.97	1.14
4	12_10_(1_5)_L	5	-	12.92	6.88	4.62	-	-	-	2.64	1.76	1.48
5	12_10_(1_3)_S	5	-	10.91	5.62	2.65	-	-	-	7.22	2.64	1.28
6	12_10_(1_3)_L	5	-	9.89	3.93	1.03	-	-	-	3.67	1.84	1.20
7	12_16_(6_8)_S	5	-	17.51	12.80	7.59	-	-	-	951.84	212.58	5.57
8	12_16_(6_8)_L	5	-	15.64	10.49	5.19	-	-	-	190.13	43.50	1.50
9	12_16_(2_6)_S	5	-	14.98	9.86	4.51	-	-	-	4.96	2.61	1.36
10	12_16_(2_6)_L	5	-	19.34	9.83	5.71	-	-	-	28.97	9.31	1.39
11	12_16_(2_4)_S	5	-	24.93	11.85	4.08	-	-	-	5.07	3.63	1.48
12	12_16_(2_4)_L	5	-	30.00	12.90	0.00	-	-	-	3.25	2.34	0.56
13	14_10_(3_5)_S	4	1	29.54	16.01	6.60	6.09	1.22	0.00	1800.67	783.24	11.87
14	14_10_(3_5)_L	3	2	14.94	9.94	6.13	3.78	0.93	0.00	1801.21	1114.54	29.91
15	14_10_(1_5)_S	5	-	7.47	6.07	5.45	-	-	-	621.51	230.19	3.85
16	14_10_(1_5)_L	4	1	7.48	5.89	3.30	3.29	0.67	0.00	1800.75	386.20	13.25
17	14_10_(1_3)_S	4	1	48.22	15.91	4.47	1.81	0.37	0.00	1800.72	467.29	3.99
18	14_10_(1_3)_L	4	1	7.08	4.64	2.32	0.64	0.13	0.00	1800.88	377.83	1.16
19	14_16_(6_8)_S	2	3	19.89	15.60	10.14	6.18	2.38	0.00	1801.17	1261.56	74.21
20	14_16_(6_8)_L	3	2	15.76	13.34	8.56	9.67	2.33	0.00	1800.83	933.28	14.10
21	14_16_(2_6)_S	4	1	14.15	10.71	8.65	2.97	0.60	0.00	1800.85	486.62	3.79
22	14_16_(2_6)_L	3	2	14.90	9.85	6.08	3.68	1.08	0.00	1801.38	996.84	10.20
23	14_16_(2_4)_S	3	2	10.73	7.29	5.84	3.49	0.89	0.00	1801.03	782.67	21.39
24	14_16_(2_4)_L	4	1	41.11	14.56	2.85	1.17	0.24	0.00	1800.58	676.25	2.26
25	16_10_(3_5)_S	-	5	21.31	11.85	6.84	3.33	2.02	0.24	1801.44	1801.20	1800.70
26	16_10_(3_5)_L	-	5	12.09	10.42	7.67	2.41	1.67	0.25	1801.52	1801.40	1801.22
27	16_10_(1_5)_S	4	1	8.41	6.64	5.45	0.85	0.18	0.00	1800.94	664.93	154.10
28	16_10_(1_5)_L	5	-	12.97	10.58	7.59	-	-	-	1340.53	537.79	237.68
29	16_10_(1_3)_S	4	1	13.62	6.80	1.88	0.89	0.18	0.00	1801.05	610.54	44.29
30	16_10_(1_3)_L	4	1	6.63	4.74	3.30	0.29	0.07	0.00	1800.83	893.55	4.18
31	16_16_(6_8)_S	-	5	24.48	18.02	11.62	8.33	5.01	1.51	1801.78	1801.41	1800.81
32	16_16_(6_8)_L	-	5	22.37	17.90	11.51	6.50	4.07	1.15	1801.89	1801.64	1801.29
33	16_16_(2_6)_S	2	3	23.10	13.33	6.31	2.96	1.20	0.00	1804.72	1561.19	956.04
34	16_16_(2_6)_L	3	2	42.62	15.89	7.22	4.57	1.28	0.00	1801.41	980.29	2.09
35	16_16_(2_4)_S	3	2	14.68	9.94	6.25	0.80	0.22	0.00	1800.75	1100.13	15.32
36	16_16_(2_4)_L	4	1	10.43	7.86	3.89	1.53	0.31	0.00	1800.72	692.73	48.27

Table 8: DSAP Model Results with Instance Set A

		Status		Root Gap %			CPU Time (sec)		
No.	**Scenario**	**Optimal**	**Terminated**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
1	12_10_(3_5)_S	5	-	-	-	-	2.59	2.49	2.29
2	12_10_(3_5)_L	5	-	-	-	-	2.70	2.52	2.31
3	12_10_(1_5)_S	5	-	-	-	-	2.43	2.14	1.89
4	12_10_(1_5)_L	5	-	-	-	-	2.61	2.30	1.86
5	12_10_(1_3)_S	5	-	-	-	-	1.87	1.80	1.65
6	12_10_(1_3)_L	5	-	-	-	-	1.83	1.58	1.40
7	12_16_(6_8)_S	5	-	-	-	-	6.16	5.65	4.96
8	12_16_(6_8)_L	5	-	-	-	-	5.85	5.51	5.02
9	12_16_(2_6)_S	5	-	-	-	-	4.42	3.75	3.06
10	12_16_(2_6)_L	5	-	-	-	-	4.38	4.05	3.64
11	12_16_(2_4)_S	5	-	-	-	-	4.02	3.39	2.92
12	12_16_(2_4)_L	5	-	-	-	-	3.18	2.90	2.62
13	14_10_(3_5)_S	5	-	-	-	-	4.20	3.22	2.76
14	14_10_(3_5)_L	5	-	-	-	-	3.81	3.24	3.03
15	14_10_(1_5)_S	5	-	-	-	-	3.40	2.83	2.12
16	14_10_(1_5)_L	5	-	-	-	-	3.65	3.01	2.61
17	14_10_(1_3)_S	5	-	-	-	-	2.29	2.17	2.03
18	14_10_(1_3)_L	5	-	-	-	-	2.15	1.98	1.84
19	14_16_(6_8)_S	5	-	-	-	-	8.86	7.11	6.41
20	14_16_(6_8)_L	5	-	-	-	-	7.43	6.91	6.61
21	14_16_(2_6)_S	5	-	-	-	-	6.72	5.53	4.56
22	14_16_(2_6)_L	5	-	-	-	-	6.94	5.44	4.76
23	14_16_(2_4)_S	5	-	-	-	-	4.79	4.37	4.10
24	14_16_(2_4)_L	5	-	-	-	-	4.17	3.38	2.90
25	16_10_(3_5)_S	5	-	-	-	-	4.20	3.66	3.21
26	16_10_(3_5)_L	5	-	-	-	-	4.13	3.76	3.46
27	16_10_(1_5)_S	5	-	-	-	-	3.56	3.21	2.81
28	16_10_(1_5)_L	5	-	-	-	-	3.57	3.27	3.01
29	16_10_(1_3)_S	5	-	-	-	-	3.04	2.64	2.45
30	16_10_(1_3)_L	5	-	-	-	-	2.76	2.42	2.18
31	16_16_(6_8)_S	5	-	-	-	-	9.17	8.59	7.55
32	16_16_(6_8)_L	5	-	0.47	0.09	0.00	9.08	8.23	7.86
33	16_16_(2_6)_S	5	-	-	-	-	8.03	6.32	4.81
34	16_16_(2_6)_L	5	-	-	-	-	6.76	5.88	4.60
35	16_16_(2_4)_S	5	-	-	-	-	5.54	5.20	4.65
36	16_16_(2_4)_L	5	-	-	-	-	4.77	4.38	3.98

Table 9: YATS model results compared with DSAP results for instance Set A

		Stopping criterion (1)							Stopping criterion (2)			
		Status	Gap %			CPU Time (sec)			Status	CPU Time (sec)		
No.	Scenario	Opt.	Max	Avg	Min	Max	Avg	Min	Opt.	Max	Avg	Min
1	12_10_(3_5)_S	5	-	-	-	0.44	0.38	0.34	5	4.52	3.80	3.43
2	12_10_(3_5)_L	5	-	-	-	0.44	0.39	0.36	5	4.58	3.95	3.74
3	12_10_(1_5)_S	5	-	-	-	0.48	0.43	0.39	5	4.82	4.25	3.93
4	12_10_(1_5)_L	5	-	-	-	0.51	0.43	0.37	5	4.75	4.22	3.70
5	12_10_(1_3)_S	5	-	-	-	0.49	0.42	0.37	5	4.98	4.30	3.85
6	12_10_(1_3)_L	5	-	-	-	0.50	0.43	0.38	5	4.92	4.44	3.85
7	12_16_(6_8)_S	4	0.34	0.07	0.00	0.65	0.44	0.34	5	4.60	4.04	3.47
8	12_16_(6_8)_L	5	-	-	-	0.49	0.42	0.36	5	4.34	3.90	3.67
9	12_16_(2_6)_S	5	-	-	-	0.53	0.43	0.38	5	4.98	4.28	3.76
10	12_16_(2_6)_L	5	-	-	-	0.44	0.42	0.38	5	4.73	4.24	3.74
11	12_16_(2_4)_S	5	-	-	-	0.48	0.44	0.38	5	4.85	4.35	3.95
12	12_16_(2_4)_L	5	-	-	-	0.52	0.44	0.38	5	5.13	4.40	3.75
13	14_10_(3_5)_S	4	0.41	0.08	0.00	0.69	0.59	0.49	5	5.79	5.30	4.84
14	14_10_(3_5)_L	5	-	-	-	0.79	0.60	0.50	5	5.70	5.39	4.83
15	14_10_(1_5)_S	5	-	-	-	0.73	0.63	0.50	5	6.97	6.28	5.12
16	14_10_(1_5)_L	5	-	-	-	0.69	0.65	0.58	5	6.54	6.10	5.43
17	14_10_(1_3)_S	5	-	-	-	0.83	0.66	0.55	5	6.89	6.26	5.47
18	14_10_(1_3)_L	5	-	-	-	1.05	0.78	0.66	5	7.53	6.79	6.04
19	14_16_(6_8)_S	4	0.38	0.08	0.00	1.13	0.70	0.50	5	6.33	5.41	4.87
20	14_16_(6_8)_L	5	-	-	-	0.75	0.59	0.51	5	5.81	5.25	4.59
21	14_16_(2_6)_S	4	0.21	0.04	0.00	0.90	0.70	0.59	5	8.72	6.88	5.88
22	14_16_(2_6)_L	5	-	-	-	0.81	0.66	0.54	5	6.99	6.25	5.44
23	14_16_(2_4)_S	4	0.40	0.08	0.00	0.79	0.68	0.60	5	7.13	6.48	5.93
24	14_16_(2_4)_L	5	-	-	-	0.89	0.70	0.63	5	7.24	6.67	6.25
25	16_10_(3_5)_S	5	-	-	-	0.87	0.76	0.65	5	8.21	7.39	6.59
26	16_10_(3_5)_L	4	0.47	0.09	0.00	0.75	0.71	0.62	5	7.82	6.90	5.95
27	16_10_(1_5)_S	5	-	-	-	0.97	0.82	0.69	5	8.66	7.93	7.09
28	16_10_(1_5)_L	5	-	-	-	1.20	0.87	0.73	5	8.55	7.91	7.18
29	16_10_(1_3)_S	5	-	-	-	1.02	0.91	0.83	5	9.55	8.48	7.57
30	16_10_(1_3)_L	5	-	-	-	1.39	0.99	0.75	5	9.73	8.66	7.83
31	16_16_(6_8)_S	5	-	-	-	0.87	0.74	0.59	5	7.34	6.72	6.01
32	16_16_(6_8)_L	5	-	-	-	1.45	0.95	0.61	5	7.80	7.00	6.00
33	16_16_(2_6)_S	4	0.23	0.05	0.00	0.96	0.84	0.70	5	9.24	8.40	7.10
34	16_16_(2_6)_L	5	-	-	-	1.44	1.05	0.80	5	8.93	8.16	7.67
35	16_16_(2_4)_S	5	-	-	-	1.35	0.95	0.78	5	9.16	8.52	7.77
36	16_16_(2_4)_L	5	-	-	-	1.33	1.00	0.77	5	9.94	8.66	7.50

Table 10: YATS_VNS model results compared with DSAP results for instance Set A

		Stopping criterion (1)							Stopping criterion (2)			
		Status	Gap %			CPU Time (sec)			Status	CPU Time (sec)		
No.	Scenario	Opt.	Max	Avg	Min	Max	Avg	Min	Opt.	Max	Avg	Min
1	12_10_(3_5)_S	5	-	-	-	0.45	0.40	0.35	5	4.45	3.81	3.48
2	12_10_(3_5)_L	5	-	-	-	0.47	0.41	0.37	5	4.54	3.97	3.70
3	12_10_(1_5)_S	5	-	-	-	0.51	0.44	0.40	5	4.88	4.24	3.90
4	12_10_(1_5)_L	5	-	-	-	0.51	0.45	0.41	5	4.77	4.26	3.73
5	12_10_(1_3)_S	5	-	-	-	0.51	0.44	0.40	5	4.98	4.28	3.81
6	12_10_(1_3)_L	5	-	-	-	0.55	0.46	0.40	5	4.94	4.47	3.88
7	12_16_(6_8)_S	4	0.34	0.07	0.00	0.45	0.41	0.37	5	7.04	4.61	3.48
8	12_16_(6_8)_L	5	-	-	-	0.45	0.41	0.39	5	4.37	3.96	3.67
9	12_16_(2_6)_S	5	-	-	-	0.52	0.45	0.40	5	5.07	4.32	3.75
10	12_16_(2_6)_L	5	-	-	-	0.53	0.45	0.37	5	4.77	4.27	3.78
11	12_16_(2_4)_S	5	-	-	-	0.52	0.45	0.41	5	4.96	4.36	3.88
12	12_16_(2_4)_L	5	-	-	-	0.50	0.46	0.39	5	5.17	4.42	3.73
13	14_10_(3_5)_S	5	-	-	-	0.68	0.61	0.56	5	5.90	5.33	4.79
14	14_10_(3_5)_L	5	-	-	-	0.89	0.63	0.52	5	5.70	5.43	4.95
15	14_10_(1_5)_S	5	-	-	-	0.79	0.66	0.52	5	7.07	6.26	5.19
16	14_10_(1_5)_L	5	-	-	-	0.70	0.65	0.56	5	6.64	6.24	5.80
17	14_10_(1_3)_S	5	-	-	-	0.79	0.67	0.59	5	7.13	6.43	5.58
18	14_10_(1_3)_L	5	-	-	-	0.80	0.70	0.63	5	7.38	6.81	6.18
19	14_16_(6_8)_S	4	0.57	0.11	0.00	0.68	0.58	0.51	5	6.41	5.68	4.96
20	14_16_(6_8)_L	5	-	-	-	0.59	0.54	0.48	5	5.93	5.28	4.60
21	14_16_(2_6)_S	4	0.21	0.04	0.00	0.83	0.74	0.60	5	11.05	7.44	5.93
22	14_16_(2_6)_L	4	0.21	0.04	0.00	0.71	0.65	0.59	5	6.93	6.27	5.56
23	14_16_(2_4)_S	5	-	-	-	1.39	0.79	0.61	5	7.16	6.51	5.93
24	14_16_(2_4)_L	5	-	-	-	0.86	0.72	0.62	5	7.54	6.75	6.26
25	16_10_(3_5)_S	5	-	-	-	1.35	0.91	0.66	5	9.06	7.46	6.35
26	16_10_(3_5)_L	4	-	-	-	0.82	0.75	0.63	5	7.75	6.91	6.11
27	16_10_(1_5)_S	5	-	-	-	1.38	0.93	0.69	5	8.99	7.97	6.93
28	16_10_(1_5)_L	5	-	-	-	0.99	0.83	0.72	5	8.82	8.02	7.13
29	16_10_(1_3)_S	5	-	-	-	1.02	0.88	0.80	5	9.53	8.59	7.65
30	16_10_(1_3)_L	5	-	-	-	1.21	0.99	0.81	5	9.63	8.74	7.92
31	16_16_(6_8)_S	5	-	-	-	0.85	0.71	0.60	5	7.30	6.60	5.86
32	16_16_(6_8)_L	4	0.29	0.06	0.00	0.81	0.74	0.67	5	7.96	7.03	6.11
33	16_16_(2_6)_S	4	0.23	0.05	0.00	1.41	0.97	0.77	5	13.60	9.27	7.01
34	16_16_(2_6)_L	5	0.25	0.05	0.00	1.41	0.96	0.79	5	10.18	8.74	7.70
35	16_16_(2_4)_S	5	-	-	-	1.12	0.98	0.88	5	9.27	8.59	7.93
36	16_16_(2_4)_L	4	0.23	0.05	0.00	1.09	0.96	0.84	5	10.01	8.74	7.62

6.4 Results of MILP Models with Instance Set B

The results of the computational experiments of instance set B solved with CSAP and DSAP models are shown in Table 11 and Table 14, respectively. Since preliminary results showed that CSA model cannot solve any medium-sized instance optimally, the comparisons will be provided with CSAP and DSAP models. Additionally, for instance set B, the comparisons of the solutions of MILP models and the metaheuristics are presented in this section. Table 11 shows that CSAP model can solve 91 out of 360 instances to optimality when the time limit is half an hour. Although the maximum gap is 24.78% for an instance from a scenario where there are 29 vessels, the average of maximum gaps for all instances is 3.56%. Additionally, the average gap for all instances is 2.12%.

Moreover, Table 12 shows the results of instance set B where the time limit is 120 seconds. As it can be seen in Table 12, the maximum gap is 26.34% when the model stops after 120 seconds. Thus the average of maximum gaps is 4.22% and the average gap for all instances is 2.72%. The results indicate that CSAP model can find good solutions within a short time. In fact, when the upper bounds obtained from CSAP are compared with the optimal values, it can be asserted that in most of the cases they are equal. Table 13 shows the percentage deviation of the best solution of CSAP and the optimal solution obtained with DSAP model. The deviation is calculated by the Equation 5.1.

The results of DSAP model shows that out of 72 scenarios, 42 scenarios are solved to optimality within a short time. However, after 43rd scenario, the model gives out of memory errors on a desktop having 8 GB RAM. Table 14 indicates that the maximum CPU time is 63.74 seconds out of fist 210 instances. In order to provide the best found values for Table 13, the rest of the scenarios are solved with a laptop having 16 GB RAM. Since a CPU time comparison is not fair with different computers having different configurations, these results are not included in the table.

In addition to the MILP models, instance set B is solved with YATS and YATS_VNS. Table 15 and Table 16 show the results of these metaheuristics.

First, in Table 15 YATS model is compared with DSAP model. The bold numbers under the CPU time columns show the superior results of YATS over DSAP model with respect to average CPU times. Under the gap column, "-" means that there is no gap between the best found solution of metaheuristic and the optimal solution obtained by DSAP. Most of the cases, YATS model achieves optimal result in shorter time when compared to DSAP. Moreover, in Table 16 the solutions of YATS_VNS model are compared with DASP model.

YATS cannot obtain optimal solutions for only 5 scenarios and within these scenarios totally 8 instances are terminated due to time limit. In addition, for 37th scenario, where there are 24 stockpiles with lengths between (3, 5) and the length of pads is 10, in one instance the model cannot find any feasible solution and terminates due to the maximum number of iterations.

6.5 Results of MILP Models with Instance Set C

This data set is a special case of set B where there are only 16, 18 and 20 stockpiles. The planning horizon is again 1 week. The major difference is that the arrival times of stockpiles are $t = 1$ and departure times of vessels are $t = T$, meaning that all stockpiles are ready at the beginning of the time horizon and there is 1 week time to serve vessels, hence to store stockpiles.

Since the preliminary results indicate that CSA and CSAP cannot obtain any root bounds for instance set C within a reasonable time, the data set C will be tested with DSAP model, YATS and YATS_VNS algorithms.

Table 11: CSAP Model Results with Instance Set B

		Status		Root Gap %			Gap %			CPU Time (sec)		
No.	Scenario	Opt.	Term	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	16_10_(3_5)_S	4	1	5.03	3.53	2.77	1.11	0.23	0.01	1800.40	451.67	20.67
2	16_10_(3_5)_L	4	1	6.57	3.43	2.45	0.49	0.11	0.01	1804.27	461.62	8.47
3	16_10_(1_5)_S	5	-	4.36	3.37	2.00	-	-	-	218.03	100.12	5.04
4	16_10_(1_5)_L	5	-	4.63	3.22	2.00	-	-	-	422.01	138.84	4.93
5	16_10_(1_3)_S	5	-	2.76	1.84	1.01	0.00	0.00	0.00	732.52	202.10	1.93
6	16_10_(1_3)_L	4	1	3.23	2.53	1.34	0.17	0.04	0.00	1800.82	380.20	1.95
7	16_24_(8_10)_S	4	1	12.54	7.65	5.12	1.70	0.35	0.01	1800.91	526.48	23.67
8	16_24_(8_10)_L	3	2	10.98	7.64	5.31	1.72	0.55	0.01	1800.60	828.03	15.35
9	16_24_(2_10)_S	5	-	8.65	5.21	3.78	-	-	-	346.17	88.52	3.62
10	16_24_(2_10)_L	4	1	11.93	6.10	3.84	0.17	0.04	0.00	1800.69	424.38	4.10
11	16_24_(2_6)_S	5	-	8.51	4.49	2.57	-	-	-	936.01	224.66	4.26
12	16_24_(2_6)_L	5	-	8.54	5.77	4.50	-	-	-	195.24	40.41	0.94
13	18_10_(3_5)_S	1	4	5.49	4.27	3.25	0.92	0.29	0.01	1800.75	1541.42	504.54
14	18_10_(3_5)_L	2	3	4.38	3.86	3.04	1.06	0.55	0.01	1801.23	1363.00	446.13
15	18_10_(1_5)_S	2	3	5.01	3.03	1.98	0.90	0.22	0.01	1801.13	1200.17	24.02
16	18_10_(1_5)_L	4	1	3.63	2.92	2.01	0.31	0.07	0.01	1800.81	730.12	8.14
17	18_10_(1_3)_S	3	2	3.00	2.58	2.15	0.11	0.04	0.01	1804.19	864.31	8.22
18	18_10_(1_3)_L	3	2	4.93	2.93	1.41	0.44	0.18	0.00	1800.81	933.60	19.11
19	18_24_(8_10)_S	-	5	10.19	8.78	6.75	1.55	0.91	0.28	1801.05	1800.83	1800.66
20	18_24_(8_10)_L	1	4	10.16	9.02	7.33	2.88	1.29	0.01	1804.16	1575.58	671.35
21	18_24_(2_10)_S	2	3	6.71	5.57	4.66	0.83	0.34	0.01	1800.95	1356.59	346.26
22	18_24_(2_10)_L	1	4	8.07	6.08	4.84	0.97	0.39	0.01	1801.04	1460.69	99.44
23	18_24_(2_6)_S	1	4	7.66	4.52	2.80	1.05	0.46	0.01	1800.91	1525.53	424.39
24	18_24_(2_6)_L	3	2	6.78	4.25	2.40	1.27	0.33	0.01	1801.28	1024.62	32.73
25	20_10_(3_5)_S	-	5	13.50	6.27	3.98	2.01	0.93	0.34	1804.14	1801.27	1800.48
26	20_10_(3_5)_L	-	5	9.59	6.80	3.37	2.81	0.97	0.34	1800.80	1800.65	1800.52
27	20_10_(1_5)_S	1	4	5.59	4.01	2.71	0.79	0.55	0.01	1804.06	1513.42	359.72
28	20_10_(1_5)_L	2	3	6.40	4.20	1.73	0.91	0.26	0.01	1801.00	1227.00	70.54
29	20_10_(1_3)_S	1	4	4.53	3.07	2.21	1.87	0.57	0.01	1802.35	1733.07	1458.26
30	20_10_(1_3)_L	3	2	5.22	3.98	2.06	0.80	0.24	0.01	1800.89	1093.45	182.83
31	20_24_(8_10)_S	-	5	16.19	10.24	7.39	4.07	2.33	1.02	1801.28	1801.16	1800.97
32	20_24_(8_10)_L	-	5	13.10	9.65	6.55	7.06	3.20	1.53	1801.25	1801.13	1800.98
33	20_24_(2_10)_S	2	3	10.47	7.08	4.65	2.25	0.67	0.01	1801.34	1294.21	515.82
34	20_24_(2_10)_L	1	4	7.69	6.61	5.41	1.75	0.80	0.01	1801.72	1514.66	367.96
35	20_24_(2_6)_S	2	3	8.13	5.91	4.14	1.60	0.70	0.01	1802.03	1587.81	1231.64
36	20_24_(2_6)_L	2	3	9.94	6.79	4.33	2.33	0.76	0.00	1801.48	1269.08	23.09

		Status		Root Gap %			Gap %			CPU Time (sec)		
No.	Scenario	Opt.	Term	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
37	24_10_(3_5)_S	-	5	100.00	28.28	8.58	5.32	2.83	0.12	1801.32	1801.07	1800.59
38	24_10_(3_5)_L	-	5	100.00	26.49	6.54	5.41	3.46	1.95	1801.23	1801.07	1800.89
39	24_10_(1_5)_S	-	5	19.22	8.07	4.21	2.60	1.31	0.44	1801.33	1801.08	1800.74
40	24_10_(1_5)_L	-	5	17.01	8.77	4.95	2.70	1.96	1.30	1801.39	1801.25	1801.15
41	24_10_(1_3)_S	-	5	10.41	6.25	4.24	2.13	1.46	0.82	1804.49	1801.75	1800.66
42	24_10_(1_3)_L	-	5	17.17	6.30	2.45	2.79	1.16	0.38	1802.29	1801.71	1800.86
43	24_24_(8_10)_S	-	5	14.50	13.53	11.72	7.48	6.36	5.43	1800.92	1800.75	1800.60
44	24_24_(8_10)_L	-	5	100.00	46.18	9.88	5.61	5.09	4.59	1801.15	1801.04	1800.90
45	24_24_(2_10)_S	-	5	16.61	11.78	5.13	6.22	4.27	2.00	1801.92	1801.58	1801.00
46	24_24_(2_10)_L	-	5	16.65	9.51	6.13	7.40	3.29	1.58	1801.61	1801.28	1800.98
47	24_24_(2_6)_S	-	5	13.53	8.82	6.25	6.33	3.13	2.00	1801.81	1801.40	1800.65
48	24_24_(2_6)_L	-	5	23.98	14.53	5.09	3.87	2.75	1.85	1801.61	1801.42	1801.10
49	27_10_(3_5)_S	-	5	100.00	82.49	12.43	6.09	5.23	4.58	1801.11	1800.95	1800.84
50	27_10_(3_5)_L	-	5	15.54	13.32	10.57	7.69	5.02	3.59	1801.24	1801.06	1800.85
51	27_10_(1_5)_S	-	5	11.66	7.93	3.76	2.40	1.73	0.76	1801.45	1801.21	1800.81
52	27_10_(1_5)_L	-	5	6.94	5.32	3.26	2.24	1.80	0.81	1801.38	1801.09	1800.54
53	27_10_(1_3)_S	-	5	7.56	4.58	2.50	2.65	1.49	1.00	1801.47	1801.31	1801.03
54	27_10_(1_3)_L	-	5	7.62	4.84	2.94	1.93	1.37	0.72	1801.58	1801.37	1801.28
55	27_24_(8_10)_S	-	5	100.00	65.77	13.50	9.18	7.35	2.91	3601.13	2520.81	1800.55
56	27_24_(8_10)_L	-	5	23.27	18.11	12.79	7.92	6.57	4.41	1801.27	1801.20	1801.06
57	27_24_(2_10)_S	-	5	19.93	13.09	7.37	6.68	4.77	2.77	1803.69	1801.49	1800.65
58	27_24_(2_10)_L	-	5	22.57	13.72	8.12	6.42	4.00	3.20	1802.17	1801.59	1800.86
59	27_24_(2_6)_S	-	5	9.76	7.57	5.19	3.45	2.90	2.23	1801.99	1801.45	1801.15
60	27_24_(2_6)_L	-	5	16.44	9.65	5.57	5.63	2.92	1.67	1801.58	1801.29	1800.58
61	29_10_(3_5)_S	-	5	100.00	28.55	9.60	5.51	4.75	4.29	1801.20	1800.94	1800.70
62	29_10_(3_5)_L	-	5	100.00	29.05	2.58	5.40	3.32	0.04	1804.19	1802.30	1800.89
63	29_10_(1_5)_S	-	5	15.40	10.14	5.83	2.82	2.06	1.24	1801.36	1801.02	1800.67
64	29_10_(1_5)_L	-	5	19.40	10.95	4.93	4.15	1.99	0.88	1801.99	1801.54	1801.30
65	29_10_(1_3)_S	-	5	12.55	7.27	4.20	2.36	1.77	1.26	1803.90	1801.46	1800.75
66	29_10_(1_3)_L	-	5	8.62	5.21	3.84	1.97	1.30	1.03	1804.18	1802.28	1801.39
67	29_24_(8_10)_S	1	4	100.00	51.63	16.93	0.12	0.09	0.06	1801.11	1440.91	0.55
68	29_24_(8_10)_L	-	5	100.00	50.88	14.20	24.78	10.45	5.54	1801.12	1801.06	1800.98
69	29_24_(2_10)_S	-	5	20.27	13.24	8.04	7.09	4.43	2.68	1802.18	1801.66	1801.24
70	29_24_(2_10)_L	-	5	16.10	13.69	8.69	8.07	4.94	3.33	1801.56	1801.22	1800.91
71	29_24_(2_6)_S	-	5	17.22	12.62	9.04	5.64	3.75	2.83	1802.53	1801.96	1800.56
72	29_24_(2_6)_L	-	5	19.96	11.96	8.63	4.24	2.82	2.25	1801.64	1801.52	1801.45

Table 12: CSAP Model Results with Instance Set B terminated after 120 sec

		Status		Root Gap %			Gap %			CPU Time (sec)		
No.	Scenario	Opt.	Term	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	16_10_(3_5)_S	2	3	5.03	3.53	2.77	1.77	0.63	0.01	121.09	83.22	20.08
2	16_10_(3_5)_L	1	4	6.57	3.43	2.45	1.34	0.57	0.01	120.92	98.36	8.80
3	16_10_(1_5)_S	3	2	4.36	3.37	2.00	0.34	0.13	0.01	120.74	69.35	5.60
4	16_10_(1_5)_L	3	2	4.63	3.22	2.00	0.38	0.1	0.01	120.90	56.15	5.37
5	16_10_(1_3)_S	3	2	2.76	1.84	1.01	0.38	0.15	0.00	120.81	55.91	2.46
6	16_10_(1_3)_L	4	1	3.23	2.53	1.34	1.01	0.21	0	120.62	44.50	2.36
7	16_24_(8_10)_S	1	4	12.54	7.65	5.12	2.68	1.32	0.01	121.14	101.49	24.20
8	16_24_(8_10)_L	2	3	10.98	7.64	5.31	3.20	1.58	0.01	121.21	83.32	15.88
9	16_24_(2_10)_S	4	1	8.65	5.21	3.78	0.39	0.08	0	120.78	43.60	4.28
10	16_24_(2_10)_L	4	1	11.93	6.10	3.84	2.55	0.61	0	120.78	57.04	4.70
11	16_24_(2_6)_S	4	1	8.51	4.49	2.57	0.89	0.18	0	120.71	61.79	4.26
12	16_24_(2_6)_L	4	1	8.54	5.77	4.50	0.33	0.07	0	120.65	25.57	1.20
13	18_10_(3_5)_S	-	5	5.49	4.27	3.25	1.83	1.19	0.53	121.10	120.91	120.56
14	18_10_(3_5)_L	-	5	4.38	3.86	3.04	2.15	1.65	0.48	121.07	120.97	120.89
15	18_10_(1_5)_S	1	4	5.01	3.03	1.98	1.19	0.55	0.01	121.34	101.58	23.70
16	18_10_(1_5)_L	1	4	3.63	2.92	2.01	1.32	0.40	0.01	121.20	98.33	7.86
17	18_10_(1_3)_S	1	4	3.00	2.58	2.15	0.64	0.27	0.01	121.26	98.61	8.77
18	18_10_(1_3)_L	1	4	4.93	2.93	1.41	0.75	0.33	0.00	121.24	100.77	19.73
19	18_24_(8_10)_S	-	5	10.19	8.78	6.75	4.28	3.31	2.11	121.14	120.98	120.76
20	18_24_(8_10)_L	-	5	10.16	9.02	7.33	4.59	3.75	2.99	121.23	120.98	120.84
21	18_24_(2_10)_S	-	5	6.71	5.57	4.66	1.54	1.17	0.37	121.21	121.15	121.10
22	18_24_(2_10)_L	1	4	8.07	6.08	4.84	1.66	1.12	0.01	121.15	116.16	96.89
23	18_24_(2_6)_S	-	5	7.66	4.52	2.80	1.83	0.92	0.42	121.34	121.22	121.10
24	18_24_(2_6)_L	1	4	6.78	4.25	2.40	2.05	0.75	0.01	121.54	103.35	32.03
25	20_10_(3_5)_S	-	5	13.50	6.27	3.98	3.10	1.92	1.34	121.14	121.02	120.89
26	20_10_(3_5)_L	-	5	9.59	6.80	3.37	3.80	2.10	1.49	121.10	120.98	120.81
27	20_10_(1_5)_S	-	5	5.59	4.01	2.71	1.62	1.07	0.38	121.14	121.09	121.06
28	20_10_(1_5)_L	1	4	6.40	4.20	1.73	1.75	0.73	0.01	121.26	110.99	70.59
29	20_10_(1_3)_S	-	5	4.53	3.07	2.21	1.91	0.77	0.17	121.42	121.30	121.15
30	20_10_(1_3)_L	-	5	5.22	3.98	2.06	1.16	0.44	0.09	121.28	121.17	120.93
31	20_24_(8_10)_S	-	5	16.19	10.24	7.39	6.02	4.11	2.52	120.99	120.89	120.79
32	20_24_(8_10)_L	-	5	13.10	9.65	6.55	8.26	4.89	3.61	121.12	121.00	120.89
33	20_24_(2_10)_S	-	5	10.47	7.08	4.65	2.70	1.51	0.81	121.28	121.15	120.95
34	20_24_(2_10)_L	-	5	7.69	6.61	5.41	2.69	1.79	0.81	121.32	121.09	120.95
35	20_24_(2_6)_S	-	5	8.13	5.91	4.14	2.53	1.47	0.67	121.38	121.19	121.03
36	20_24_(2_6)_L	1	4	9.94	6.79	4.33	2.61	1.07	0.00	121.21	100.83	19.92

		Status		Root Gap %			Gap %			CPU Time (sec)		
No.	**Scenario**	**Opt.**	**Term**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
37	24_10_(3_5)_S	-	5	100.00	28.28	8.58	6.41	4.68	3.29	120.96	120.79	120.57
38	24_10_(3_5)_L	-	5	100.00	26.49	6.54	7.64	4.43	2.68	121.03	120.85	120.68
39	24_10_(1_5)_S	-	5	19.22	8.07	4.21	3.60	1.93	1.20	121.09	120.96	120.75
40	24_10_(1_5)_L	-	5	17.01	8.77	4.95	3.65	2.64	1.75	121.12	121.00	120.81
41	24_10_(1_3)_S	-	5	10.41	6.25	4.24	2.23	1.55	0.82	121.15	121.01	120.89
42	24_10_(1_3)_L	-	5	17.17	6.30	2.45	2.95	1.35	0.57	121.28	121.16	121.01
43	24_24_(8_10)_S	-	5	14.50	13.53	11.72	8.24	7.32	6.49	121.07	120.92	120.81
44	24_24_(8_10)_L	-	5	100.00	46.18	9.88	8.25	6.74	5.63	121.04	120.90	120.79
45	24_24_(2_10)_S	-	5	16.61	11.78	5.13	7.61	4.85	2.33	121.09	120.95	120.84
46	24_24_(2_10)_L	-	5	16.65	9.51	6.13	7.81	3.94	2.34	121.18	121.01	120.93
47	24_24_(2_6)_S	-	5	13.53	8.82	6.25	6.33	3.33	2.19	121.32	121.17	120.98
48	24_24_(2_6)_L	-	5	23.98	14.53	5.09	4.45	3.23	1.98	121.20	121.09	120.93
49	27_10_(3_5)_S	-	5	100.00	82.49	12.43	6.67	5.90	5.39	135.10	123.72	120.78
50	27_10_(3_5)_L	-	5	15.54	13.32	10.57	8.12	5.74	4.29	121.10	121.00	120.93
51	27_10_(1_5)_S	-	5	11.66	7.93	3.76	2.62	2.15	1.47	121.17	121.05	120.95
52	27_10_(1_5)_L	-	5	6.94	5.32	3.26	2.52	2.06	0.99	121.28	121.16	121.04
53	27_10_(1_3)_S	-	5	7.56	4.58	2.50	2.65	1.50	1.02	121.29	121.19	121.07
54	27_10_(1_3)_L	-	5	7.62	4.84	2.94	2.00	1.52	0.85	121.24	121.16	120.96
55	27_24_(8_10)_S	-	5	100.00	65.77	13.50	10.08	9.39	9.03	121.04	120.96	120.89
56	27_24_(8_10)_L	-	5	23.27	18.11	12.79	8.80	7.38	4.82	121.10	121.00	120.84
57	27_24_(2_10)_S	-	5	19.93	13.09	7.37	6.91	5.07	3.10	121.28	121.12	120.99
58	27_24_(2_10)_L	-	5	22.57	13.72	8.12	7.41	4.41	3.24	121.29	121.06	120.84
59	27_24_(2_6)_S	-	5	9.76	7.57	5.19	3.66	3.06	2.39	121.48	121.29	121.15
60	27_24_(2_6)_L	-	5	16.44	9.65	5.57	5.88	3.02	1.69	121.21	121.10	120.95
61	29_10_(3_5)_S	-	5	100.00	28.55	9.60	6.16	5.27	4.85	121.32	121.09	120.95
62	29_10_(3_5)_L	-	5	100.00	48.53	10.54	5.84	4.86	4.11	121.12	120.97	120.65
63	29_10_(1_5)_S	-	5	15.40	10.14	5.83	3.00	2.26	1.48	121.14	121.01	120.90
64	29_10_(1_5)_L	-	5	19.40	10.95	4.93	4.56	2.43	1.17	121.28	121.14	121.04
65	29_10_(1_3)_S	-	5	12.55	7.27	4.20	2.40	1.85	1.31	121.28	121.15	120.98
66	29_10_(1_3)_L	-	5	8.62	5.21	3.84	2.11	1.34	1.06	121.42	121.22	120.96
67	29_24_(8_10)_S	1	4	100.00	51.63	16.93	13.35	9.77	6.51	121.49	97.05	0.55
68	29_24_(8_10)_L	-	5	100.00	50.88	14.20	26.34	11.44	6.19	121.26	121.09	120.89
69	29_24_(2_10)_S	-	5	20.27	13.24	8.04	7.23	4.61	2.83	121.23	121.04	120.84
70	29_24_(2_10)_L	-	5	16.10	13.69	8.69	8.58	5.13	3.61	121.28	121.11	120.87
71	29_24_(2_6)_S	-	5	17.22	12.62	9.04	5.64	3.87	3.03	121.29	121.08	120.92
72	29_24_(2_6)_L	-	5	19.96	11.96	8.63	4.59	2.98	2.34	121.40	121.15	121.01

Table 13: The average percent deviation of CSAP model results from DSAP model for instance Set B

No.	Scenario	Avg PD %	No.	Scenario	Avg PD %
1	16_10_(3_5)_S	**0.00**	37	24_10_(3_5)_S	0.09
2	16_10_(3_5)_L	**0.00**	38	24_10_(3_5)_L	0.06
3	16_10_(1_5)_S	**0.00**	39	24_10_(1_5)_S	0.06
4	16_10_(1_5)_L	**0.00**	40	24_10_(1_5)_L	0.07
5	16_10_(1_3)_S	**0.00**	41	24_10_(1_3)_S	0.05
6	16_10_(1_3)_L	**0.00**	42	24_10_(1_3)_L	0.02
7	16_24_(8_10)_S	**0.00**	43	24_24_(8_10)_S	0.04
8	16_24_(8_10)_L	**0.00**	44	24_24_(8_10)_L	0.09
9	16_24_(2_10)_S	**0.00**	45	24_24_(2_10)_S	0.44
10	16_24_(2_10)_L	**0.00**	46	24_24_(2_10)_L	0.17
11	16_24_(2_6)_S	**0.00**	47	24_24_(2_6)_S	0.21
12	16_24_(2_6)_L	**0.00**	48	24_24_(2_6)_L	0.11
13	18_10_(3_5)_S	**0.00**	49	27_10_(3_5)_S	0.30
14	18_10_(3_5)_L	**0.00**	50	27_10_(3_5)_L	0.19
15	18_10_(1_5)_S	**0.00**	51	27_10_(1_5)_S	0.16
16	18_10_(1_5)_L	**0.00**	52	27_10_(1_5)_L	0.11
17	18_10_(1_3)_S	**0.00**	53	27_10_(1_3)_S	0.04
18	18_10_(1_3)_L	**0.00**	54	27_10_(1_3)_L	0.10
19	18_24_(8_10)_S	**0.00**	55	27_24_(8_10)_S	0.05
20	18_24_(8_10)_L	**0.00**	56	27_24_(8_10)_L	0.55
21	18_24_(2_10)_S	**0.00**	57	27_24_(2_10)_S	0.26
22	18_24_(2_10)_L	0.04	58	27_24_(2_10)_L	0.32
23	18_24_(2_6)_S	0.02	59	27_24_(2_6)_S	0.26
24	18_24_(2_6)_L	**0.00**	60	27_24_(2_6)_L	0.16
25	20_10_(3_5)_S	**0.00**	61	29_10_(3_5)_S	0.22
26	20_10_(3_5)_L	**0.00**	62	29_10_(3_5)_L	0.18
27	20_10_(1_5)_S	**0.00**	63	29_10_(1_5)_S	0.11
28	20_10_(1_5)_L	**0.00**	64	29_10_(1_5)_L	0.15
29	20_10_(1_3)_S	**0.00**	65	29_10_(1_3)_S	0.20
30	20_10_(1_3)_L	**0.00**	66	29_10_(1_3)_L	0.06
31	20_24_(8_10)_S	**0.00**	67	29_24_(8_10)_S	0.67
32	20_24_(8_10)_L	0.07	68	29_24_(8_10)_L	0.53
33	20_24_(2_10)_S	0.04	69	29_24_(2_10)_S	0.38
34	20_24_(2_10)_L	0.07	70	29_24_(2_10)_L	0.46
35	20_24_(2_6)_S	**0.00**	71	29_24_(2_6)_S	0.24
36	20_24_(2_6)_L	0.02	72	29_24_(2_6)_L	0.24

Table 14: DSAP Model Results with Instance Set B

		Status		Root Gap %			CPU Time (sec)		
No.	**Scenario**	**Opt.**	**Term.**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
1	16_10_(3_5)_S	5	-	0.00	0.00	0.00	6.74	6.10	5.54
2	16_10_(3_5)_L	5	-	0.00	0.00	0.00	6.77	6.09	5.71
3	16_10_(1_5)_S	5	-	0.00	0.00	0.00	5.63	5.25	4.57
4	16_10_(1_5)_L	5	-	0.00	0.00	0.00	5.73	5.34	4.68
5	16_10_(1_3)_S	5	-	0.00	0.00	0.00	4.40	4.04	3.60
6	16_10_(1_3)_L	5	-	0.00	0.00	0.00	4.62	3.83	3.01
7	16_24_(8_10)_S	5	-	0.00	0.00	0.00	63.74	37.41	28.36
8	16_24_(8_10)_L	5	-	0.00	0.00	0.00	43.68	36.61	31.00
9	16_24_(2_10)_S	5	-	0.00	0.00	0.00	28.53	24.41	22.46
10	16_24_(2_10)_L	5	-	0.00	0.00	0.00	39.81	28.64	22.56
11	16_24_(2_6)_S	5	-	0.00	0.00	0.00	21.48	18.02	15.27
12	16_24_(2_6)_L	5	-	0.00	0.00	0.00	22.71	19.91	16.79
13	18_10_(3_5)_S	5	-	0.00	0.00	0.00	15.63	9.64	7.02
14	18_10_(3_5)_L	5	-	0.00	0.00	0.00	12.50	9.65	7.68
15	18_10_(1_5)_S	5	-	0.00	0.00	0.00	8.17	7.45	6.27
16	18_10_(1_5)_L	5	-	0.00	0.00	0.00	9.16	6.90	5.76
17	18_10_(1_3)_S	5	-	0.00	0.00	0.00	6.65	5.98	5.06
18	18_10_(1_3)_L	5	-	0.00	0.00	0.00	4.54	4.17	3.64
19	18_24_(8_10)_S	5	-	0.00	0.00	0.00	43.88	37.90	33.34
20	18_24_(8_10)_L	5	-	0.00	0.00	0.00	50.11	42.12	33.15
21	18_24_(2_10)_S	5	-	0.00	0.00	0.00	49.11	32.66	24.77
22	18_24_(2_10)_L	5	-	0.00	0.00	0.00	36.86	29.35	21.17
23	18_24_(2_6)_S	5	-	0.00	0.00	0.00	24.82	21.87	17.85
24	18_24_(2_6)_L	5	-	0.00	0.00	0.00	27.58	21.50	17.71
25	20_10_(3_5)_S	5	-	0.00	0.00	0.00	12.23	10.65	8.47
26	20_10_(3_5)_L	5	-	0.00	0.00	0.00	8.49	7.80	7.54
27	20_10_(1_5)_S	5	-	0.00	0.00	0.00	7.21	6.43	5.79
28	20_10_(1_5)_L	5	-	0.00	0.00	0.00	6.58	6.33	5.73
29	20_10_(1_3)_S	5	-	0.00	0.00	0.00	5.48	4.84	4.40

		Status		Root Gap %			CPU Time (sec)		
No.	Scenario	Opt.	Term.	Max	Avg	Min	Max	Avg	Min
30	20_10_(1_3)_L	5	-	0.00	0.00	0.00	5.18	4.62	4.09
31	20_24_(8_10)_S	5	-	0.00	0.00	0.00	47.75	43.71	39.30
32	20_24_(8_10)_L	5	-	0.00	0.00	0.00	56.15	50.78	48.41
33	20_24_(2_10)_S	5	-	0.00	0.00	0.00	36.02	32.48	25.04
34	20_24_(2_10)_L	5	-	0.00	0.00	0.00	45.18	36.81	28.58
35	20_24_(2_6)_S	5	-	0.00	0.00	0.00	28.49	25.33	21.83
36	20_24_(2_6)_L	5	-	0.00	0.00	0.00	25.62	22.11	15.73
37	24_10_(3_5)_S	5	-	2.86	0.77	0.00	36.54	23.31	17.16
38	24_10_(3_5)_L	5	-	0.94	0.33	0.00	26.54	21.13	18.13
39	24_10_(1_5)_S	5	-	0.00	0.00	0.00	18.88	16.70	12.21
40	24_10_(1_5)_L	5	-	0.10	0.02	0.00	24.24	20.75	16.47
41	24_10_(1_3)_S	5	-	0.00	0.00	0.00	15.16	12.62	9.27
42	24_10_(1_3)_L	5	-	0.00	0.00	0.00	13.29	12.27	11.09

Table 15: YATS Model Results when compared with DSAP results for Instance Set B

		YATS								DSAP Model		
					Gap%		CPU Time (sec)			CPU Time (sec)		
No.	Scenario	Opt	Term	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	16_10_(3_5)_S	5		-	-	-	5.11	**4.69**	4.32	6.74	6.10	5.54
2	16_10_(3_5)_L	5		-	-	-	5.11	**4.60**	4.05	6.77	6.09	5.71
3	16_10_(1_5)_S	5		-	-	-	5.70	5.49	5.00	5.63	**5.25**	4.57
4	16_10_(1_5)_L	5		-	-	-	5.43	**5.25**	5.03	5.73	5.34	4.68
5	16_10_(1_3)_S	5		-	-	-	6.13	5.70	5.24	4.40	**4.04**	3.60
6	16_10_(1_3)_L	5		-	-	-	6.38	6.02	5.67	4.62	**3.83**	3.01
7	16_24_(8_10)_S	5		-	-	-	4.90	**4.54**	4.01	63.74	37.41	28.36
8	16_24_(8_10)_L	5		-	-	-	5.09	**4.62**	4.18	43.68	36.61	31.00
9	16_24_(2_10)_S	5		-	-	-	5.86	**5.45**	4.87	28.53	24.41	22.46
10	16_24_(2_10)_L	5		-	-	-	5.71	**5.47**	5.17	39.81	28.64	22.56
11	16_24_(2_6)_S	5		-	-	-	6.02	**5.82**	5.49	21.48	18.02	15.27
12	16_24_(2_6)_L	5		-	-	-	6.29	**6.07**	5.66	22.71	19.91	16.79
13	18_10_(3_5)_S	4	1	0.07	0.01	0.00	6.67	6.14	5.58	15.63	9.64	7.02
14	18_10_(3_5)_L	5		-	-	-	7.58	**6.44**	5.33	12.50	9.65	7.68
15	18_10_(1_5)_S	5		-	-	-	7.22	**6.72**	6.16	8.17	7.45	6.27

				YATS						DSAP Model		
				Gap%			CPU Time (sec)			CPU Time (sec)		
No.	Scenario	Opt	Term	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
16	18_10_(1_5)_L	5		-	-	-	7.10	**6.79**	5.93	9.16	6.90	5.76
17	18_10_(1_3)_S	5		-	-	-	7.58	7.30	7.13	6.65	**5.98**	5.06
18	18_10_(1_3)_L	5		-	-	-	8.06	7.88	7.64	4.54	**4.17**	3.64
19	18_24_(8_10)_S	5		-	-	-	6.31	**5.80**	5.30	43.88	37.90	33.34
20	18_24_(8_10)_L	5		-	-	-	6.24	**5.69**	5.28	50.11	42.12	33.15
21	18_24_(2_10)_S	5		-	-	-	7.09	**6.80**	6.35	49.11	32.66	24.77
22	18_24_(2_10)_L	5		-	-	-	7.30	**7.05**	6.52	36.86	29.35	21.17
23	18_24_(2_6)_S	5		-	-	-	7.87	**7.07**	6.51	24.82	21.87	17.85
24	18_24_(2_6)_L	5		-	-	-	8.54	**7.93**	7.45	27.58	21.50	17.71
25	20_10_(3_5)_S	5		-	-	-	7.37	**6.99**	6.52	12.23	10.65	8.47
26	20_10_(3_5)_L	5		-	-	-	10.13	**7.22**	5.60	8.49	7.80	7.54
27	20_10_(1_5)_S	5		-	-	-	14.80	10.21	7.92	7.21	**6.43**	5.79
28	20_10_(1_5)_L	4	1	0.11	0.02	0.00	10.08	8.97	8.53	6.58	**6.33**	5.73
29	20_10_(1_3)_S	5		-	-	-	9.70	9.16	8.60	5.48	**4.84**	4.40
30	20_10_(1_3)_L	5		-	-	-	10.66	10.26	10.07	5.18	**4.62**	4.09
31	20_24_(8_10)_S	5		-	-	-	8.20	**7.44**	6.61	47.75	43.71	39.30
32	20_24_(8_10)_L	4	1	0.85	0.17	0.00	8.72	**7.15**	5.99	56.15	50.78	48.41
33	20_24_(2_10)_S	5		-	-	-	10.12	**9.52**	9.14	36.02	32.48	25.04
34	20_24_(2_10)_L	5		-	-	-	14.80	**10.14**	8.66	45.18	36.81	28.58
35	20_24_(2_6)_S	5		-	-	-	9.87	**9.19**	8.50	28.49	25.33	21.83
36	20_24_(2_6)_L	5		-	-	-	12.28	**10.47**	9.89	25.62	22.11	15.73
37	24_10_(3_5)_S	4	1*	-	-	-	10.28	**10.28**	10.28	36.54	23.31	17.16
38	24_10_(3_5)_L	1	4	0.74	0.30	0.00	41.16	20.69	13.97	26.54	21.13	18.13
39	24_10_(1_5)_S	5		-	-	-	25.42	**16.53**	11.65	18.88	16.70	12.21
40	24_10_(1_5)_L	5		-	-	-	13.71	**12.47**	9.69	24.24	20.75	16.47
41	24_10_(1_3)_S	5		-	-	-	15.55	14.16	13.47	15.16	**12.62**	9.27
42	24_10_(1_3)_L	5		-	-	-	15.99	15.36	14.84	13.29	**12.27**	11.09

* YATS model could not find any feasible solution in one instance

Table 16: YATS_VNS Model Results when compared with DSAP results for Instance Set B

				YATS_VNS						**DSAP Model**		
					Gap%			**CPU Time (sec)**			**CPU Time (sec)**	
No.	**Scenario**	**Opt**	**Term**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
1	16_10_(3_5)_S	5		-	-	-	4.78	**4.54**	4.31	6.74	6.10	5.54
2	16_10_(3_5)_L	4	1	0.17	0.03	0.00	5.14	**4.60**	3.94	6.77	6.09	5.71
3	16_10_(1_5)_S	5		-	-	-	5.49	5.40	5.20	5.63	**5.25**	4.57
4	16_10_(1_5)_L	5		-	-	-	5.34	**5.16**	4.88	5.73	5.34	4.68
5	16_10_(1_3)_S	5		-	-	-	5.95	5.62	5.22	4.40	**4.04**	3.60
6	16_10_(1_3)_L	5		-	-	-	6.23	5.94	5.74	4.62	**3.83**	3.01
7	16_24_(8_10)_S	5		-	-	-	6.85	**5.22**	3.99	63.74	37.41	28.36
8	16_24_(8_10)_L	5		-	-	-	4.88	**4.46**	4.01	43.68	36.61	31.00
9	16_24_(2_10)_S	5		-	-	-	5.75	**5.41**	4.97	28.53	24.41	22.46
10	16_24_(2_10)_L	5		-	-	-	5.64	**5.32**	4.93	39.81	28.64	22.56
11	16_24_(2_6)_S	5		-	-	-	6.03	**5.67**	5.42	21.48	18.02	15.27
12	16_24_(2_6)_L	5		-	-	-	6.15	**5.88**	5.37	22.71	19.91	16.79
13	18_10_(3_5)_S	4	1	0.07	0.01	0.00	6.64	**5.98**	5.55	15.63	9.64	7.02
14	18_10_(3_5)_L	4	1	0.07	0.01	0.00	6.20	**5.49**	4.92	12.50	9.65	7.68
15	18_10_(1_5)_S	5		-	-	-	7.33	**6.79**	5.96	8.17	7.45	6.27
16	18_10_(1_5)_L	5		-	-	-	7.16	**6.73**	5.87	9.16	6.90	5.76
17	18_10_(1_3)_S	5		-	-	-	7.27	7.08	6.78	6.65	**5.98**	5.06
18	18_10_(1_3)_L	5		-	-	-	7.90	7.68	7.42	4.54	**4.17**	3.64
19	18_24_(8_10)_S	5		-	-	-	6.26	**5.40**	4.95	43.88	37.90	33.34
20	18_24_(8_10)_L	5		-	-	-	5.66	**5.36**	4.88	50.11	42.12	33.15
21	18_24_(2_10)_S	5		-	-	-	8.33	**7.29**	6.61	49.11	32.66	24.77
22	18_24_(2_10)_L	5	1	0.07	0.01	0.00	7.64	**7.20**	6.84	36.86	29.35	21.17
23	18_24_(2_6)_S	5		-	-	-	7.67	**7.07**	6.51	24.82	21.87	17.85
24	18_24_(2_6)_L	5		-	-	-	8.60	**8.01**	7.38	27.58	21.50	17.71
25	20_10_(3_5)_S	5		-	-	-	7.84	**7.05**	6.01	12.23	10.65	8.47
26	20_10_(3_5)_L	5		-	-	-	11.02	**7.64**	6.27	8.49	7.80	7.54
27	20_10_(1_5)_S	4	1	0.10	0.02	0.00	9.89	8.89	7.73	7.21	**6.43**	5.79
28	20_10_(1_5)_L	4	1	0.11	0.02	0.00	9.84	8.97	8.38	6.58	**6.33**	5.73
29	20_10_(1_3)_S	5		-	-	-	12.49	9.79	8.67	5.48	**4.84**	4.40
30	20_10_(1_3)_L	5		-	-	-	10.43	10.13	9.94	5.18	**4.62**	4.09
31	20_24_(8_10)_S	4	1	0.32	0.06	0.00	8.44	**7.25**	6.57	47.75	43.71	39.30
32	20_24_(8_10)_L	4	1	0.53	0.12	0.00	9.78	**7.44**	6.61	56.15	50.78	48.41

				YATS_VNS						DSAP Model		
				Gap%			CPU Time (sec)			CPU Time (sec)		
No.	**Scenario**	**Opt**	**Term**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
33	20_24_(2_10)_S	5		-	-	-	16.99	**11.54**	8.90	36.02	32.48	25.04
34	20_24_(2_10)_L	5		0.11	0.02	0.00	9.55	**9.11**	8.54	45.18	36.81	28.58
35	20_24_(2_6)_S	5		-	-	-	9.69	**9.25**	8.81	28.49	25.33	21.83
36	20_24_(2_6)_L	5		-	-	-	17.78	**11.46**	9.72	25.62	22.11	15.73
37	24_10_(3_5)_S	4	1*	0.15	0.03	0.00	172.04	43.17	10.52	36.54	**23.31**	17.16
38	24_10_(3_5)_L	4	1	0.95	0.35	0.00	14.14	**11.07**	8.76	26.54	21.13	18.13
39	24_10_(1_5)_S	4	1	0.23	0.05	0.00	15.65	**13.77**	11.35	18.88	16.70	12.21
40	24_10_(1_5)_L	5		-	-	-	16.71	**13.64**	9.94	24.24	20.75	16.47
41	24_10_(1_3)_S	5		-	-	-	16.47	14.33	13.45	15.16	**12.62**	9.27
42	24_10_(1_3)_L	5		-	-	-	19.75	16.11	14.94	13.29	**12.27**	11.09

* YATS_VNS model could not find any feasible solution in one instance

Moreover, to obtain better bounds, DSAP model is tested with a time limit of 1 hour. According to Table 17, within an hour, 137 out of 180 instances can be solved to optimality. For other instances the maximum gap is 3.95% which is not a large gap. On the other hand, the average CPU time is approximately 1060 seconds. Therefore, a metaheuristic is needed to obtain fast and good quality results.

Additionally, The YATS and YATS_VNS heuristics are compared with the best obtained solutions of DSAP. The results are provided in Table 18 and Table 19.

Although YATS can achieve as good results as DSAP in 61 instances, in others the gaps are also not very large. For example, in Table 18 the maximum gap is 14.41% whereas, the overall average gap is 5.39%. On the other hand, the CPU times are decreased in considerable amount. In 138 out of 180 instances, DSAP can achieve the optimum results.

In Table 19, YATS_VNS also exhibits similar results. The overall average gap is decreased to 4.37% while the overall average CPU time remains the same with YATS_VNS. Additionally, the maximum gap is decreased to 12.48%.

Table 17: DSAP Model Results with Instance Set C

		Status		Root Gap %			Gap %			CPU Time (sec)		
No.	Scenario	Opt	Term	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	16_10_(8_10)_S_336	5	-	8.83	3.92	0.37	-	-	-	223.58	115.97	44.18
2	16_10_(8_10)_L_336	5	-	3.35	1.80	0.00	-	-	-	720.43	182.50	38.67
3	16_10_(2_10)_S_336	5	-	10.29	5.77	0.00	-	-	-	91.01	49.68	24.91
4	16_10_(2_10)_L_336	5	-	15.63	9.57	4.29	-	-	-	94.13	58.80	35.77
5	16_10_(4_8)_S_336	5	-	0.00	0.00	0.00	-	-	-	23.29	18.91	15.88
6	16_10_(4_8)_L_336	5	-	0.00	0.00	0.00	-	-	-	20.87	16.94	14.32
7	16_10_(6_6)_S_336	5	-	3.68	2.24	0.65	-	-	-	170.96	128.90	51.34
8	16_10_(6_6)_L_336	5	-	9.78	6.78	3.14	-	-	-	283.92	159.11	84.33
9	16_10_(8_10)_S_504	5	-	24.67	14.31	4.19	-	-	-	681.88	406.98	138.64
10	16_10_(8_10)_L_504	5	-	18.27	12.76	7.70	-	-	-	567.44	273.49	53.07
11	16_10_(2_10)_S_504	5	-	0.00	0.00	0.00	-	-	-	26.13	19.82	15.63
12	16_10_(2_10)_L_504	5	-	0.00	0.00	0.00	-	-	-	19.10	17.83	16.90
13	16_10_(4_8)_S_504	4	1	5.99	2.72	0.74	0.19	0.04	0.00	3600.15	848.49	89.28
14	16_10_(4_8)_L_504	5	-	7.80	3.84	1.53	-	-	-	245.50	152.20	74.79
15	16_10_(6_6)_S_504	5	-	15.69	9.49	4.26	-	-	-	904.99	566.60	248.21
16	16_10_(6_6)_L_504	5	-	18.04	13.29	8.96	-	-	-	885.28	477.10	240.46
17	18_10_(8_10)_S_336	5	-	0.00	0.00	0.00	-	-	-	24.15	19.96	17.22
18	18_10_(8_10)_L_336	5	-	0.00	0.00	0.00	-	-	-	32.23	26.98	21.64
19	18_10_(2_10)_S_336	3	2	4.98	3.06	1.24	1.46	0.39	0.00	3601.83	2433.15	323.92
20	18_10_(2_10)_L_336	4	1	6.09	4.00	0.40	0.84	0.17	0.00	3602.11	1227.84	117.45
21	18_10_(4_8)_S_336	1	4	21.21	11.79	8.02	1.73	0.85	0.00	3600.83	3482.41	3009.93
22	18_10_(4_8)_L_336	-	5	16.00	12.28	9.12	1.38	0.90	0.52	3600.89	3600.60	3600.33
23	18_10_(6_6)_S_336	5	-	0.00	0.00	0.00	-	-	-	33.29	28.77	22.34
24	18_10_(6_6)_L_336	5	-	0.00	0.00	0.00	-	-	-	33.13	27.10	21.51
25	18_10_(8_10)_S_504	2	3	6.01	3.48	2.09	1.22	0.53	0.00	3601.72	2420.26	340.25
26	18_10_(8_10)_L_504	4	1	4.71	2.25	0.55	2.07	0.41	0.00	3600.42	1256.79	241.02
27	18_10_(2_10)_S_504	-	5	14.35	11.14	7.89	2.55	1.64	0.72	3601.15	3600.62	3600.24
28	18_10_(2_10)_L_504	-	5	18.98	13.79	9.99	3.60	2.55	1.83	3601.34	3600.61	3600.22
29	18_10_(4_8)_S_504	5	-	0.00	0.00	0.00	-	-	-	50.73	37.50	23.82
30	18_10_(4_8)_L_504	5	-	0.00	0.00	0.00	-	-	-	38.58	30.98	25.37
31	18_10_(6_6)_S_504	2	3	3.65	2.50	1.22	1.37	0.40	0.00	3601.85	3064.64	1352.92
32	18_10_(6_6)_L_504	2	3	6.70	4.46	3.29	2.11	0.55	0.00	3600.72	2624.20	529.31
33	20_10_(8_10)_S_336	-	5	14.87	10.70	8.52	3.95	2.86	1.58	3602.01	3600.66	3600.25
34	20_10_(8_10)_L_336	-	5	15.73	12.69	3.55	2.76	2.12	0.95	3602.19	3600.76	3600.08
35	20_10_(2_10)_S_336	5	-	0.00	0.00	0.00	-	-	-	54.23	44.72	34.21
36	20_10_(2_10)_L_336	5	-	0.00	0.00	0.00	-	-	-	55.44	45.96	32.60

Table 18: YATS Model Results when compared with DSAP results for Instance Set C

		YATS						DSAP Model		
		Gap%			CPU Time (sec)			CPU Time (sec)		
No.	Scenario	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	16_24_(8_10)_S	5.30	2.35	0.00	8.99	5.44	3.27	223.58	115.97	44.18
2	16_24_(8_10)_L	11.06	4.35	0.68	4.33	3.85	3.36	720.43	182.5	38.67
3	16_24_(2_10)_S	5.30	3.92	2.46	7.68	6.11	4.63	91.01	49.68	24.91
4	16_24_(2_10)_L	11.34	5.74	2.93	6.26	5.59	5.14	94.13	58.8	35.77
5	16_24_(6_6)_S	-	-	-	5.35	5.25	5.20	23.29	18.91	15.88
6	16_24_(6_6)_L	-	-	-	5.35	5.26	5.19	20.87	16.94	14.32
7	18_24_(8_10)_S	3.50	2.01	0.56	10.05	6.07	4.86	170.96	128.9	51.34
8	18_24_(8_10)_L	5.71	2.92	0.41	7.16	5.07	4.29	283.92	159.11	84.33
9	18_24_(2_10)_S	9.57	7.07	3.91	8.26	6.60	5.59	681.88	406.98	138.64
10	18_24_(2_10)_L	13.06	5.92	0.59	8.71	7.05	5.94	567.44	273.49	53.07
11	18_24_(6_6)_S	-	-	-	7.39	7.07	6.94	26.13	19.82	15.63
12	18_24_(6_6)_L	-	-	-	7.47	7.08	6.96	19.1	17.83	16.9
13	20_24_(8_10)_S	4.34	2.39	0.29	9.68	6.97	5.67	3600.15	848.49	89.28
14	20_24_(8_10)_L	5.03	3.50	0.87	15.26	9.13	5.70	245.5	152.2	74.79
15	20_24_(2_10)_S	7.39	5.83	3.77	21.38	12.98	8.68	904.99	566.6	248.21
16	20_24_(2_10)_L	10.63	4.89	1.42	17.09	10.69	7.41	885.28	477.1	240.46
17	20_24_(6_6)_S	-	-	-	9.51	9.23	8.94	24.15	19.96	17.22
18	20_24_(6_6)_L	-	-	-	9.05	8.92	8.78	32.23	26.98	21.64
19	24_24_(8_10)_S	10.96	3.67	0.32	20.76	12.87	7.46	3601.83	2433.15	323.92
20	24_24_(8_10)_L	11.34	4.55	1.90	17.87	11.04	9.03	3602.11	1227.84	117.45
21	24_24_(2_10)_S	14.41	10.26	6.27	19.14	14.83	12.51	3600.83	3482.41	3009.93
22	24_24_(2_10)_L	10.35	8.74	7.93	29.69	19.49	11.44	3600.89	3600.6	3600.33
23	24_24_(6_6)_S	-	-	-	13.43	13.11	12.79	33.29	28.77	22.34
24	24_24_(6_6)_L	-	-	-	13.34	12.95	12.44	33.13	27.1	21.51
25	27_24_(8_10)_S	10.08	3.85	1.06	36.56	20.47	11.86	3601.72	2420.26	340.25
26	27_24_(8_10)_L	13.30	4.17	0.34	17.96	14.44	11.20	3600.42	1256.79	241.02
27	27_24_(2_10)_S	12.52	7.61	5.83	36.87	27.41	15.29	3601.15	3600.62	3600.24
28	27_24_(2_10)_L	9.81	8.74	7.57	36.99	24.76	16.67	3601.34	3600.61	3600.22
29	27_24_(6_6)_S	-	-	-	20.57	18.47	17.07	50.73	37.5	23.82
30	27_24_(6_6)_L	-	-	-	18.44	17.99	17.64	38.58	30.98	25.37
31	29_24_(8_10)_S	7.93	5.12	0.92	26.45	20.11	15.81	3601.85	3064.64	1352.92
32	29_24_(8_10)_L	9.74	5.18	2.83	31.25	21.84	15.78	3600.72	2624.2	529.31
33	29_24_(2_10)_S	12.97	8.83	4.94	33.94	29.52	25.94	3602.01	3600.66	3600.25
34	29_24_(2_10)_L	10.29	7.65	4.90	66.31	43.85	27.41	3602.19	3600.76	3600.08
35	29_24_(6_6)_S	-	-	-	22.69	21.40	20.43	54.23	44.72	34.21
36	29_24_(6_6)_L	-	-	-	28.49	23.94	20.43	55.44	45.96	32.6

Table 19: YATS_VNS Model Results compared with DSAP results for Instance Set C

		YATS_VNS						DSAP Model		
		Gap%			CPU Time (sec)			CPU Time (sec)		
No.	Scenario	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
1	16_24_(8_10)_S	1.69	0.84	0.22	4.52	4.11	3.80	223.58	115.97	44.18
2	16_24_(8_10)_L	4.40	2.28	0.68	4.59	3.94	3.60	720.43	182.5	38.67
3	16_24_(2_10)_S	6.01	2.51	0.00	16.21	8.83	5.67	91.01	49.68	24.91
4	16_24_(2_10)_L	7.17	3.14	0.47	7.23	5.91	4.76	94.13	58.8	35.77
5	16_24_(6_6)_S	-	-	-	5.49	5.36	5.23	23.29	18.91	15.88
6	16_24_(6_6)_L	-	-	-	5.95	5.52	5.28	20.87	16.94	14.32
7	18_24_(8_10)_S	3.70	2.68	1.57	5.64	5.22	4.76	170.96	128.9	51.34
8	18_24_(8_10)_L	4.76	2.26	0.41	9.12	6.18	4.66	283.92	159.11	84.33
9	18_24_(2_10)_S	5.56	4.86	3.72	12.05	8.79	6.19	681.88	406.98	138.64
10	18_24_(2_10)_L	7.10	3.27	0.41	9.84	8.17	5.77	567.44	273.49	53.07
11	18_24_(6_6)_S	-	-	-	7.57	7.32	7.21	26.13	19.82	15.63
12	18_24_(6_6)_L	-	-	-	7.29	7.15	7.01	19.1	17.83	16.9
13	20_24_(8_10)_S	4.90	2.15	0.29	14.42	9.00	5.81	3600.15	848.49	89.28
14	20_24_(8_10)_L	4.60	3.02	0.46	10.95	7.31	5.93	245.5	152.2	74.79
15	20_24_(2_10)_S	8.37	5.99	0.00	24.60	14.71	7.26	904.99	566.6	248.21
16	20_24_(2_10)_L	10.26	6.23	2.82	18.31	11.34	8.86	885.28	477.1	240.46
17	20_24_(6_6)_S	-	-	-	9.57	9.30	8.87	24.15	19.96	17.22
18	20_24_(6_6)_L	-	-	-	9.17	9.02	8.85	32.23	26.98	21.64
19	24_24_(8_10)_S	5.65	1.86	0.72	16.74	11.99	8.75	3601.83	2433.15	323.92
20	24_24_(8_10)_L	6.39	2.98	0.75	14.46	11.25	9.65	3602.11	1227.84	117.45
21	24_24_(2_10)_S	12.48	7.58	4.83	28.93	19.49	12.11	3600.83	3482.41	3009.93
22	24_24_(2_10)_L	11.59	8.18	6.11	19.76	17.80	12.17	3600.89	3600.6	3600.33
23	24_24_(6_6)_S	-	-	-	13.30	13.15	12.85	33.29	28.77	22.34
24	24_24_(6_6)_L	-	-	-	13.87	13.33	12.86	33.13	27.1	21.51
25	27_24_(8_10)_S	7.90	3.58	0.97	27.55	20.95	14.26	3601.72	2420.26	340.25
26	27_24_(8_10)_L	3.14	1.48	0.00	21.79	17.09	13.68	3600.42	1256.79	241.02
27	27_24_(2_10)_S	10.97	8.02	4.24	30.91	25.08	20.26	3601.15	3600.62	3600.24
28	27_24_(2_10)_L	10.61	8.69	6.16	31.69	24.73	15.93	3601.34	3600.61	3600.22
29	27_24_(6_6)_S	-	-	-	19.73	18.50	17.48	50.73	37.5	23.82
30	27_24_(6_6)_L	-	-	-	18.40	18.06	17.50	38.58	30.98	25.37
31	29_24_(8_10)_S	8.97	4.37	1.08	32.56	25.35	17.90	3601.85	3064.64	1352.92
32	29_24_(8_10)_L	3.78	2.52	0.34	43.50	26.32	20.29	3600.72	2624.2	529.31
33	29_24_(2_10)_S	10.39	6.90	4.36	43.10	29.78	18.48	3602.01	3600.66	3600.25
34	29_24_(2_10)_L	12.48	9.52	4.76	35.55	25.23	18.34	3602.19	3600.76	3600.08
35	29_24_(6_6)_S	-	-	-	23.11	21.32	20.59	54.23	44.72	34.21
36	29_24_(6_6)_L	-	-	-	28.49	23.94	20.43	55.44	45.96	32.6

6.6 Results of MILP Models with Instance Set D

This data set is used to generate large-sized instances where there are 48 to 58 stockpiles arriving to the stockyard in a planning horizon of 2 to 3 weeks. Since the dimensions of the stockyard as well as the planning horizon are large, the DSAP model gave out of memory errors for all instances. Since DSAP model has 1,317,216 constraints for the smallest problem. However, the largest problem size in set D that we can solve with CSAP model has 20,532 constraints, and 6,844 binary and 116 continuous variables.

We compare the CSAP model results with metaheuristics YATS and YATS_VNS for instance Set D. First, the results of CSAP model with a time limit of half an hour are presented in Table 20. Although for all instances in Set D, CSAP model is terminated due to time limit, the optimality gap over all scenarios is 3.58%. The maximum optimality gap is 12.54% whereas the minimum gap is 0.80%. Therefore, the results indicate that although the model cannot obtain optimal solutions within 30 minutes, the ability to find a good solution is acceptable. Moreover, in order to assess the performance of CSAP heuristic, we compare the objective values at different time intervals such that 30, 60 and 120 seconds. Table 21 shows that the model can obtain good solutions even within 30 seconds. Additionally, instance set D is solved with YATS and YATS_VNS heuristics and the results are provided in Table 22 and Table 23, respectively. The results indicate that, YATS can solve within 197 seconds on average over all instances. Additionally, it can obtain better objective values than the CSAP model within 1800 seconds. On average -0.15% gap is attained over all instances. The * indicates that, in the 3rd scenario CSAP is not able to find any feasible solution in one instance. Moreover, in Table 23, YATS_VNS heuristic results are compared with CSAP model. Similar to YATS results, YATS_VNS can obtain better results than CSAP model. In almost all instances, YATS_VNS improves the objective value when compared to CSAP. However, in the 2nd and 3rd scenarios the model cannot find any feasible solution in one instance each.

Table 20. CSAP Model results with instance Set D

		Root Gap %			Gap %			CPU Time (sec)		
No.	**Scenario**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**	**Max**	**Avg**	**Min**
1	48_20_(8_10)_L_336	100.00	84.74	23.68	9.31	7.63	6.16	1804.964	1802.333	1801.442
2	48_20_(8_10)_S_336	100.00	83.37	16.85	12.54	7.79	4.62	1802.471	1801.758	1801.436
3	48_20_(8_10)_M_336	25.92	21.29	15.05	4.86	4.21	3.71	1801.496	1801.34	1801.24
4	48_20_(2_10)_L_336	27.84	22.31	12.04	6.88	4.08	3.03	1801.636	1801.496	1801.308
5	48_20_(2_10)_S_336	27.61	19.42	13.01	8.97	4.02	2.29	1804.811	1802.249	1801.426
6	48_20_(2_10)_M_336	29.13	25.06	21.21	5.68	3.70	2.64	1801.504	1801.365	1801.182
7	48_20_(6_8)_L_336	100.00	34.70	15.79	5.17	3.76	2.82	1801.784	1801.457	1801.284
8	48_20_(6_8)_S_336	26.07	18.53	11.97	5.21	3.09	2.11	1802.608	1801.675	1801.33
9	48_20_(6_8)_M_336	27.91	18.13	9.92	3.47	2.76	2.18	1804.351	1801.983	1801.235
10	48_20_(4_8)_L_336	100.00	69.60	21.51	6.88	5.82	5.31	1801.513	1801.418	1801.339
11	48_20_(4_8)_S_336	100.00	37.66	16.73	5.68	4.19	3.57	1801.655	1801.522	1801.345
12	48_20_(4_8)_M_336	100.00	41.88	23.39	5.94	4.00	3.02	1801.701	1801.527	1801.255
13	54_20_(8_10)_L_336	100.00	100.00	100.00	11.27	9.53	8.24	1804.662	1802.226	1801.386
14	54_20_(8_10)_S_336	100.00	84.43	22.16	11.27	8.21	6.71	1801.972	1801.762	1801.52
15	54_20_(8_10)_M_336	100.00	84.30	21.51	8.77	7.16	4.54	1813.882	1804.585	1801.59
16	54_20_(4_8)_L_336	100.00	38.38	19.23	5.71	4.55	4.12	1802.112	1801.721	1801.596
17	54_20_(4_8)_S_336	100.00	39.45	21.95	5.31	4.45	3.77	1801.641	1801.439	1801.142
18	54_20_(4_8)_M_336	24.19	22.50	20.96	4.25	3.35	2.81	1801.823	1801.552	1801.326
19	54_20_(6_8)_L_504	21.20	18.00	12.08	2.37	1.84	1.14	1801.637	1801.517	1801.359
20	54_20_(6_8)_S_504	22.93	14.45	2.99	1.49	1.27	1.06	1801.78	1801.583	1801.326
21	54_20_(6_8)_M_504	20.33	12.97	2.58	1.60	1.18	0.80	1807.943	1802.839	1801.432
22	54_20_(2_10)_L_504	22.05	18.99	14.57	3.55	2.54	1.98	1801.55	1801.471	1801.34
23	54_20_(2_10)_S_504	22.36	17.75	7.65	2.47	2.09	1.60	1801.575	1801.457	1801.348
24	54_20_(2_10)_M_504	24.80	19.54	14.56	1.84	1.69	1.49	1801.762	1801.484	1801.25
25	58_20_(8_10)_L_504	100.00	35.22	12.31	4.68	3.65	2.74	1801.581	1801.384	1801.033
26	58_20_(8_10)_S_504	100.00	34.82	11.51	4.48	3.55	3.08	1801.58	1801.438	1801.354
27	58_20_(8_10)_M_504	23.23	16.54	9.40	4.07	2.41	1.67	1801.762	1801.464	1801.109
28	58_20_(2_10)_L_504	23.75	19.88	15.34	2.45	1.94	1.59	1801.768	1801.62	1801.496
29	58_20_(2_10)_S_504	21.31	16.42	5.59	1.85	1.55	1.33	1801.738	1801.546	1801.44
30	58_20_(2_10)_M_504	25.73	19.37	13.46	2.30	1.62	1.13	1801.595	1801.453	1801.339
31	58_20_(6_8)_L_504	26.00	21.93	16.41	2.22	1.69	1.46	1801.691	1801.54	1801.293
32	58_20_(6_8)_S_504	25.77	18.33	5.51	1.66	1.47	1.40	1801.829	1801.628	1801.466
33	58_20_(6_8)_M_504	25.48	19.81	12.48	1.41	1.21	0.83	1801.832	1801.572	1801.466
34	58_20_(4_8)_L_504	22.09	18.11	10.12	3.40	2.65	1.94	1801.671	1801.502	1801.373
35	58_20_(4_8)_S_504	23.37	18.43	9.30	2.73	2.30	1.92	1801.666	1801.529	1801.371
36	58_20_(4_8)_M_504	23.83	22.25	20.16	2.05	1.85	1.58	1801.514	1801.439	1801.409

Table 21. Comparison of CSAP model results in terms of the average optimality gap with different time limits

	Time Limit:	**1800 sec**	**120 sec**	**60 sec**	**30 sec**
No.	**Scenario**	**Avg Gap%**	**Avg Gap%**	**Avg Gap %**	**Avg Gap %**
1	48_20_(8_10)_L_336	7.63	8.18	8.32	8.55
2	48_20_(8_10)_S_336	7.79	8.27	8.42	8.91
3	48_20_(8_10)_M_336	4.21	4.52	4.67	4.98
4	48_20_(2_10)_L_336	4.08	4.31	4.52	4.64
5	48_20_(2_10)_S_336	4.02	4.34	4.54	4.61
6	48_20_(2_10)_M_336	3.70	3.90	4.03	4.09
7	48_20_(6_8)_L_336	3.76	4.22	4.42	4.57
8	48_20_(6_8)_S_336	3.09	3.29	3.52	3.73
9	48_20_(6_8)_M_336	2.76	2.99	3.08	3.18
10	48_20_(4_8)_L_336	5.82	6.20	6.86	7.10
11	48_20_(4_8)_S_336	4.19	4.58	4.76	4.92
12	48_20_(4_8)_M_336	4.00	4.44	4.62	4.83
13	54_20_(8_10)_L_336	9.53	10.25	10.63	10.80
14	54_20_(8_10)_S_336	8.21	8.80	9.10	9.15
15	54_20_(8_10)_M_336	7.16	8.06	8.46	8.64
16	54_20_(4_8)_L_336	4.55	5.02	5.28	5.39
17	54_20_(4_8)_S_336	4.45	4.73	5.11	5.24
18	54_20_(4_8)_M_336	3.35	3.69	3.92	3.93
19	54_20_(6_8)_L_504	1.84	1.97	2.08	2.11
20	54_20_(6_8)_S_504	1.27	1.39	1.42	1.44
21	54_20_(6_8)_M_504	1.18	1.28	1.33	1.38
22	54_20_(2_10)_L_504	2.54	2.70	2.84	2.93
23	54_20_(2_10)_S_504	2.09	2.23	2.26	2.39
24	54_20_(2_10)_M_504	1.69	1.75	1.82	1.85
25	58_20_(8_10)_L_504	3.65	3.78	4.03	4.15
26	58_20_(8_10)_S_504	3.55	3.74	3.93	4.00
27	58_20_(8_10)_M_504	2.41	2.52	2.56	2.61
28	58_20_(2_10)_L_504	1.94	2.07	2.19	2.19
29	58_20_(2_10)_S_504	1.55	1.64	1.71	1.71
30	58_20_(2_10)_M_504	1.62	1.78	1.83	1.85
31	58_20_(6_8)_L_504	1.69	1.82	1.99	2.02
32	58_20_(6_8)_S_504	1.47	1.60	1.64	1.66
33	58_20_(6_8)_M_504	1.21	1.31	1.35	1.35
34	58_20_(4_8)_L_504	2.65	2.88	3.07	3.17
35	58_20_(4_8)_S_504	2.30	2.46	2.54	2.68
36	58_20_(4_8)_M_504	1.85	1.98	2.10	2.13

Table 22. Comparison of YATS model results with the best obtained solutions of CSAP 1800 sec

		YATS						
		Gap%			CPU Time (sec)			
No.	Scenario	Max	Avg	Min	Max	Avg	Min	
1	48_20_(8_10)_L_336	-0.09	-0.25	-0.32	204.66	144.76	97.77	
2	48_20_(8_10)_S_336	-0.09	-0.27	-0.57	169.27	114.66	86.88	
3	48_20_(8_10)_M_336	0.02	-0.15	-0.28	165.23	145.81	113.92	*
4	48_20_(2_10)_L_336	-0.02	-0.20	-0.71	250.44	160.85	97.68	
5	48_20_(2_10)_S_336	-0.02	-0.23	-0.85	176.42	144.24	88.68	
6	48_20_(2_10)_M_336	-0.06	-0.28	-0.54	304.27	184.01	127.48	
7	48_20_(6_8)_L_336	-0.02	-0.13	-0.29	147.14	124.68	100.63	
8	48_20_(6_8)_S_336	0.13	-0.07	-0.27	180.01	155.69	130.14	
9	48_20_(6_8)_M_336	-0.04	-0.11	-0.23	208.60	160.34	135.53	
10	48_20_(4_8)_L_336	-0.10	-0.45	-0.67	215.55	131.21	75.99	
11	48_20_(4_8)_S_336	0.19	-0.10	-0.37	150.56	106.61	72.57	
12	48_20_(4_8)_M_336	0.00	-0.12	-0.40	223.56	151.77	123.97	
13	54_20_(8_10)_L_336	-0.27	-0.42	-0.58	216.73	188.42	166.02	
14	54_20_(8_10)_S_336	-0.27	-0.35	-0.42	241.68	170.19	107.27	
15	54_20_(8_10)_M_336	-0.30	-0.41	-0.69	404.56	267.82	174.79	
16	54_20_(4_8)_L_336	-0.06	-0.24	-0.41	228.08	213.57	190.49	
17	54_20_(4_8)_S_336	-0.16	-0.30	-0.47	374.14	274.36	169.98	
18	54_20_(4_8)_M_336	-0.19	-0.25	-0.35	364.54	261.84	151.36	
19	54_20_(6_8)_L_504	-0.01	-0.10	-0.22	283.35	226.53	184.90	
20	54_20_(6_8)_S_504	-0.01	-0.04	-0.07	217.79	175.08	139.34	
21	54_20_(6_8)_M_504	0.01	-0.01	-0.04	288.67	234.39	160.30	
22	54_20_(2_10)_L_504	0.01	-0.12	-0.25	368.58	211.14	131.50	
23	54_20_(2_10)_S_504	-0.01	-0.04	-0.08	220.55	173.81	115.47	
24	54_20_(2_10)_M_504	-0.01	-0.04	-0.09	228.76	196.68	172.21	
25	58_20_(8_10)_L_504	0.00	-0.06	-0.14	219.78	179.38	151.77	
26	58_20_(8_10)_S_504	0.02	-0.08	-0.20	257.01	204.43	171.19	
27	58_20_(8_10)_M_504	-0.01	-0.09	-0.20	369.33	215.67	157.03	
28	58_20_(2_10)_L_504	-0.02	-0.12	-0.21	325.45	263.84	176.47	
29	58_20_(2_10)_S_504	0.04	-0.06	-0.13	368.78	281.27	206.93	
30	58_20_(2_10)_M_504	-0.05	-0.07	-0.09	293.29	259.16	202.20	
31	58_20_(6_8)_L_504	0.05	-0.01	-0.11	301.33	229.00	162.26	
32	58_20_(6_8)_S_504	0.01	-0.06	-0.19	312.55	209.40	164.45	
33	58_20_(6_8)_M_504	0.03	-0.04	-0.08	335.89	257.04	174.36	
34	58_20_(4_8)_L_504	-0.03	-0.11	-0.28	321.48	234.87	160.71	
35	58_20_(4_8)_S_504	-0.02	-0.12	-0.27	288.91	235.38	176.25	
36	58_20_(4_8)_M_504	0.00	-0.06	-0.12	327.07	220.01	145.11	

Table 23. Comparison of YATS model results with the best obtained solutions of CSAP 1800 sec

		YATS_VNS						
		Gap%			CPU Time (sec)			
No.	Scenario	Max	Avg	Min	Max	Avg	Min	
1	48_20_(8_10)_L_336	-0.02	-0.22	-0.32	130.48	114.52	94.20	
2	48_20_(8_10)_S_336	-0.04	-0.12	-0.32	141.66	107.52	81.52	*
3	48_20_(8_10)_M_336	0.02	-0.16	-0.32	203.36	139.89	106.81	*
4	48_20_(2_10)_L_336	-0.02	-0.20	-0.62	222.29	139.17	99.57	
5	48_20_(2_10)_S_336	-0.02	-0.20	-0.83	284.18	149.98	93.71	
6	48_20_(2_10)_M_336	0.02	-0.23	-0.46	376.56	188.47	114.96	
7	48_20_(6_8)_L_336	-0.03	-0.13	-0.23	184.80	158.39	119.33	
8	48_20_(6_8)_S_336	0.04	-0.08	-0.23	278.78	153.00	81.43	
9	48_20_(6_8)_M_336	-0.04	-0.14	-0.27	251.71	203.38	132.02	
10	48_20_(4_8)_L_336	-0.31	-0.49	-0.60	212.50	165.47	129.93	
11	48_20_(4_8)_S_336	0.04	-0.07	-0.37	207.34	133.27	87.76	
12	48_20_(4_8)_M_336	0.09	-0.12	-0.55	221.83	150.17	86.83	
13	54_20_(8_10)_L_336	-0.25	-0.47	-0.63	327.06	226.94	143.45	
14	54_20_(8_10)_S_336	-0.20	-0.37	-0.58	292.47	193.03	85.21	
15	54_20_(8_10)_M_336	-0.15	-0.34	-0.56	267.81	189.96	109.44	
16	54_20_(4_8)_L_336	-0.13	-0.23	-0.38	259.31	198.24	134.80	
17	54_20_(4_8)_S_336	-0.15	-0.30	-0.44	235.96	217.65	203.94	
18	54_20_(4_8)_M_336	-0.17	-0.26	-0.41	314.58	218.91	152.27	
19	54_20_(6_8)_L_504	-0.03	-0.13	-0.29	259.30	218.97	151.93	
20	54_20_(6_8)_S_504	-0.01	-0.04	-0.07	277.78	213.48	145.25	
21	54_20_(6_8)_M_504	0.03	0.00	-0.02	326.32	217.13	153.92	
22	54_20_(2_10)_L_504	-0.03	-0.12	-0.24	307.50	187.03	127.73	
23	54_20_(2_10)_S_504	0.01	-0.04	-0.08	320.97	227.77	121.60	
24	54_20_(2_10)_M_504	-0.01	-0.02	-0.05	224.57	208.64	183.97	
25	58_20_(8_10)_L_504	0.04	-0.06	-0.15	302.53	225.56	163.29	
26	58_20_(8_10)_S_504	-0.04	-0.11	-0.23	326.67	243.92	170.84	
27	58_20_(8_10)_M_504	0.00	-0.08	-0.20	511.05	305.92	136.83	
28	58_20_(2_10)_L_504	0.00	-0.11	-0.21	321.03	241.05	191.86	
29	58_20_(2_10)_S_504	0.01	-0.06	-0.13	443.20	311.42	185.05	
30	58_20_(2_10)_M_504	-0.02	-0.06	-0.08	360.11	267.00	180.65	
31	58_20_(6_8)_L_504	0.01	-0.01	-0.05	249.14	217.09	181.38	
32	58_20_(6_8)_S_504	0.02	-0.05	-0.18	379.25	289.38	174.06	
33	58_20_(6_8)_M_504	-0.01	-0.06	-0.11	357.47	237.70	170.16	
34	58_20_(4_8)_L_504	-0.04	-0.10	-0.22	301.66	238.37	190.61	
35	58_20_(4_8)_S_504	-0.03	-0.11	-0.24	371.80	260.80	195.77	
36	58_20_(4_8)_M_504	0.00	-0.07	-0.14	340.94	234.45	168.67	

Chapter 7

CONCLUSION AND FUTURE RESEARCH

In this thesis, we studied yard allocation problem (YAP) in bulk port terminals. YAP allocates each stockpile to a place at the stockyard and each stockpile is stored at the stockyard starting from its arrival time till the corresponding vessel's departure time. In the thesis, we studied both the continuous YAP and the discrete YAP. We considered minimizing the total travelled distance at the stockyard as well as minimizing the total dwelling time over all stockpiles. Since these two quantities are different in units, (i.e. distance vs. time), we associated these quantities with their respective costs and obtained a single monetary objective.

We presented three mixed integer mathematical models (CSA, CSAP and DSAP) inspired by multi-dimensional packing problems. While, the first (CSA) and the second (CSAP) models proposed in this thesis addressed YAP in continuous time and continuous space, the third (DSAP) model we propose discussed YAP in discrete time and discrete space.

In our models, we considered two types of stockyards: with pads and without pads. In the absence of pads, the stockpiles can be stored freely provided that they are parallel to the edges of the stockyard. Therefore, the orientations of piles are determined by CSA model. On the other hand, we solved YAP in stockyards with pads in CSAP and DSAP models.

Since the underlying multi-dimensional packing problems are NP-hard, developing a computationally efficient mathematical model for YAP was challenging. Thus, a tabu search (TS) algorithm which utilizes a well-known bottom-left-fill like heuristic was presented (YATS). Then, to improve the quality of the solutions, we combined TS with Variable Neighborhood Search algorithms (YATS_VNS). YATS_VNS algorithm was used to solve YAP in large-scaled bulk port terminals. Computational experiments were conducted and the models were validated with four data sets.

We analyzed the performance of both the mathematical models and the metaheuristic algorithm with four sets of generated data including instances based on real life data. Mathematical models as well as heuristics could solve small-sized instances of YAP, which are in data Set A, to optimality within a reasonable time. For large data sets, we managed to obtain small gaps at least with one of the mathematical models. Moreover, since stockyard planning is a tactical level operation and the decisions were given on a weekly or on a monthly basis, the running times of MILP models were acceptable. Additionally, we compared the metaheuristic results with the best obtained solutions. The results indicate that we improved the solution quality and running time with YATS_VNS in almost every instance in set D. Hence, the YATS_VNS can be suggested as a decision support system to the port managers.

Although the proposed methodologies successfully addressed the problem, they can be improved by excluding some of the assumptions. First, we assumed that departure and arrival times are deterministic. One possible improvement can be making these events stochastic. Secondly, since there are various components of TS and VNS algorithms, the performance of YATS and YATS_VNS algorithms can be improved by changing neighborhoods or adding new neighborhoods. Moreover, the solution quality can be improved by adding local search procedures to the algorithms. Finally, since storage facilities need to be designed in a way to implement several safety measures and precautions, for future work, those specifications can be investigated and incorporated into the models.

BIBLIOGRAPHY

Arahori, Y., Imamichi, T., and Nagamochi, H. (2012). An exact strip packing algorithm based on canonical forms. *Computers and Operations Research*, 39(12), 2991-3011.

Baker, B. S., Coffman, Jr, E. G., and Rivest, R. L. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4), 846-855.

Bay, M., Crama, Y., Langer, Y., and Rigo, P. (2010). Space and time allocation in a shipyard assembly hall. *Annals of Operations Research*, 179(1), 57-76.

Behnamian, J., Zandieh, M., and Ghomi, S. F. (2009). Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm. *Expert Systems with Applications*, 36(6), 9637-9644.

Beltrán, J. D., Calderón, J. E., Cabrera, R. J., Moreno-Pérez, J. A., and Moreno-Vega, J. M. (2004). GRASP-VNS hybrid for the Strip Packing Problem. In *Hybrid metaheuristics 2004*, 79-90.

Bierwirth, C., & Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals. European Journal of Operational Research, 202(3), 615-627.Boland, N., Gulczynski, D., and Savelsbergh, M. (2012). A stockyard planning problem. *EURO Journal on Transportation and Logistics*, 1(3), 197-236.

Bortfeldt, A., and Gehring, H. (1998). Applying tabu search to container loading problems. In *Operations Research Proceedings* 1997 (pp. 533-538). Springer Berlin Heidelberg.

Carlo, H. J., Vis, I. F., and Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235(2), 412-430.

Castro, P. M., and Oliveira, J. F. (2011). Scheduling inspired models for two-dimensional packing problems. *European Journal of Operational Research*, 215(1), 45-56.

Chen, C. S., Lee, S. M., and Shen, Q. S. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1), 68-76.

Clarkson Research Services (2013). Dry Bulk Trade Outlook. July.

Clarkson Research Services (2014a). Dry Bulk Trade Outlook. June.

Coffman, Jr., E., Garey, M., and Johnson, D. (1983). Dynamic Bin Packing. *SIAM Journal on Computing*, 12(2), 227-258.

Johnson, D. S. (1973). Near-optimal bin packing algorithms (Doctoral dissertation, Massachusetts Institute of Technology).

Epstein, L., and Levy, M. (2010). Dynamic multi-dimensional bin packing. *Journal Of Discrete Algorithms*, 8(4), 356-372.

Glover, F. (1989). Tabu search-part I. *ORSA Journal on computing*, 1(3), 190-206.

Glover, F. (1990). Tabu search—part II. *ORSA Journal on computing*, 2(1), 4-32.

Glover, F., and Laguna, M. (2013). Tabu Search (pp. 3261-3362). *Springer New York*.

Glover, F., and Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41(1), 1-28.

Han, X., Peng, C., Ye, D., Zhang, D., and Lan, Y. (2010). Dynamic bin packing with unit fraction items revisited. *Information Processing Letters*, 110(23), 1049-1054. doi:10.1016/j.ipl.2010.09.002

Hopper, E. B. C. H., and Turton, B. C. (2001). An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1), 34-57.

Hu, D., and Yao, Z. (2010, August). Stacker-reclaimer scheduling for raw material yard operation. *In Advanced Computational Intelligence (IWACI),* 2010 Third International Workshop on (pp. 432-436). IEEE. ISO 690

Jain, S., and Gea, H. C. (1998). Two-dimensional packing problems using genetic algorithms. *Engineering with Computers*, 14(3), 206-213.

Junior, V. V., Santos, R. G., Costa, T. S., and De Oliveira, A. C. M. (2012, September). Greedy Heuristic for Berth Allocation in Tidal Bulk Ports. *In Jornada de Informática* *do Maranhão* (JIM 2012)/Escola Regional de Computação dos Estados do Ceará, Maranhão e Piauí (ERCEMAPI 2012).

Liu, S. Q., and Kozan, E. (2009). Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers and Operations Research*, 36(10), 2840-2852.

Lodi, A., Martello, S., and Monaci, M. (2002a). Two-dimensional packing problems: A survey. *European Journal Of Operational Research*, 141(2), 241-252. doi:10.1016/s0377-2217(02)00123-6

Lodi, A., Martello, S., and Vigo, D. (1999). Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. *INFORMS Journal On Computing*, 11(4), 345-357. doi:10.1287/ijoc.11.4.345

Lodi, A., Martello, S., and Vigo, D. (2002b). Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1), 379-396.

Martello, S., and Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management science*, 44(3), 388-399.

Mladenović, N., and Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24(11), 1097-1100.

Pisinger, D., and Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2), 154-167.

Robenek, T., Umang, N., Bierlaire, M., and Ropke, S. (2014). A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports. *European Journal of Operational Research*, 235(2), 399-411.

Schott, D. L., and Lodewijks, G. (2007). Analysis of dry bulk terminals: Chances for exploration. *Particle and Particle Systems Characterization*, 24(4 - 5), 375-380.

Stahlbock, R., & Voß, S. (2008). Operations research at container terminals: a literature update. *OR Spectrum*, 30(1), 1-52.

"Stockpile." Merriam-Webster.com. Accessed July 19, 2015. http://www.merriam-webster.com/dictionary/stockpile.

UNCTAD (2014). *Review of maritime transport*. United Nations conference on trade and development. <http://www.unctad.org>.

Talbi E. Metaheuristics: from design to implementation. Hoboken, New Jersey, USA: John Wiley & Sons; 2009.

Appendix 1: Presents the computational results for the parameter settings of the tabu search with data set C. The best solutions obtained by the YATS algorithm with random initial solution, denoted by R, and sorted according to arrival times of pads, denoted by S.

No.	Best Solution with (R)	Secs	Best Solution with (S)	Secs	Gap %
1	440	3.60	440	3.61	0.00
2	444	4.33	441	4.01	0.68
3	252	6.36	247	6.64	1.98
4	211	5.23	212	6.19	-0.47
5	285	5.24	285	5.26	0.00
6	273	5.32	273	5.48	0.00
7	580	4.86	580	5.91	0.00
8	507	4.80	526	5.67	-3.75
9	354	5.62	351	7.82	0.85
10	340	8.71	343	9.73	-0.88
11	375	6.97	375	7.20	0.00
12	366	6.96	366	7.25	0.00
13	634	6.38	645	5.54	-1.74
14	689	7.45	663	9.86	3.77
15	475	9.44	462	8.23	2.74
16	406	7.41	381	9.15	6.16
17	431	9.28	431	9.57	0.00
18	459	8.80	459	9.73	0.00
19	1066	16.00	1049	21.80	1.59
20	1011	17.87	951	10.74	5.93
21	609	14.51	598	23.37	1.81
22	636	11.44	647	21.65	-1.73
23	609	12.79	609	15.93	0.00
24	594	12.92	594	14.23	0.00
25	1183	26.54	1173	30.28	0.85
26	1221	17.96	1222	14.74	-0.08
27	763	29.78	803	20.94	-5.24
28	761	16.67	742	25.94	2.50
29	712	20.57	712	21.06	0.00
30	739	18.19	739	19.33	0.00
31	1422	26.45	1337	21.98	5.98
32	1506	15.78	1551	14.98	-2.99
33	914	27.19	949	30.31	-3.83
34	750	26.31	770	41.18	-2.67
35	890	20.43	890	23.82	0.00
36	893	25.25	893	23.88	0.00
Average		15.85		14.53	0.32

Appendix 2: Presents the computational experiments conducted to decide number of replications per instance for the YATS algorithm. The tests are conducted with data set C.

No.	Best Solution with (15)	Secs	Gap %	Best Solution with (5)	Secs	Gap %
1	440	3.60	0.23%	447	1.16	1.82%
2	444	4.33	0.68%	444	1.44	0.68%
3	252	6.36	4.76%	249	2.93	3.75%
4	211	5.23	2.84%	214	3.25	4.39%
5	285	5.24	0.00%	285	1.85	0.00%
6	273	5.32	0.00%	273	1.85	0.00%
7	580	4.86	1.38%	579	2.46	1.22%
8	507	4.80	1.78%	503	2.19	1.00%
9	354	5.62	6.78%	363	2.42	10.00%
10	340	8.71	0.59%	352	4.46	4.14%
11	375	6.97	0.00%	375	2.81	0.00%
12	366	6.96	0.00%	366	2.70	0.00%
13	634	6.38	2.52%	635	2.26	2.75%
14	689	7.45	4.79%	688	1.96	4.88%
15	475	9.44	6.53%	459	4.01	3.38%
16	406	7.41	9.61%	404	4.55	10.08%
17	431	9.28	0.00%	431	3.32	0.00%
18	459	8.80	0.00%	459	3.16	0.00%
19	1066	16.00	2.81%	1050	5.97	1.35%
20	1011	17.87	10.19%	976	4.94	7.49%
21	609	14.51	9.20%	597	6.10	7.96%
22	636	11.44	8.02%	618	7.40	5.64%
23	609	12.79	0.00%	609	5.43	0.00%
24	594	12.92	0.00%	594	5.01	0.00%
25	1183	26.54	3.72%	1181	11.19	3.69%
26	1221	17.96	4.91%	1204	5.91	3.70%
27	763	29.78	5.50%	811	7.68	12.48%
28	761	16.67	8.94%	752	7.71	8.51%
29	712	20.57	0.00%	712	7.04	0.00%
30	739	18.19	0.00%	739	7.57	0.00%
31	1422	26.45	7.10%	1362	10.16	3.10%
32	1506	15.78	3.19%	1500	7.70	2.88%
33	914	27.19	4.70%	951	16.75	9.18%
34	750	26.31	4.67%	771	11.93	7.83%
35	890	20.43	0.00%	890	8.11	0.00%
36	893	25.25	0.00%	893	8.95	0.00%
Average		13.15	3.21%		5.40	3.39%

Appendix 3: The preliminary test results of data set C to decide the tabu tenure.

	Tabu Tenure 3			Tabu Tenure 5			Tabu Tenure 10		
No.	**Obj Value**	**Secs**	**Gap**	**Obj Value**	**Secs**	**Gap**	**Obj Value**	**Secs**	**Gap**
1	447	3.60	1.79%	440	3.60	0.23%	446	3.28	1.57%
2	444	3.93	0.68%	444	4.33	0.68%	444	3.68	0.68%
3	252	5.88	4.76%	252	6.36	4.76%	244	5.60	1.64%
4	212	5.70	3.30%	211	5.23	2.84%	219	4.93	6.39%
5	285	5.16	0.00%	285	5.24	0.00%	285	5.30	0.00%
6	273	5.26	0.00%	273	5.32	0.00%	273	5.42	0.00%
7	577	5.38	0.87%	580	4.86	1.38%	577	6.50	0.87%
8	509	5.60	2.16%	507	4.80	1.78%	510	4.59	2.35%
9	361	6.56	8.59%	354	5.62	6.78%	343	13.28	3.79%
10	343	13.40	1.46%	340	8.71	0.59%	348	11.63	2.87%
11	375	6.94	0.00%	375	6.97	0.00%	375	7.11	0.00%
12	366	7.12	0.00%	366	6.96	0.00%	366	7.13	0.00%
13	634	8.25	2.52%	634	6.38	2.52%	633	10.47	2.37%
14	686	6.78	4.37%	689	7.45	4.79%	668	8.87	1.80%
15	486	11.71	8.64%	475	9.44	6.53%	474	6.56	6.33%
16	393	18.65	6.62%	406	7.41	9.61%	398	8.20	7.79%
17	431	9.21	0.00%	431	9.28	0.00%	431	9.53	0.00%
18	459	9.17	0.00%	459	8.80	0.00%	459	9.33	0.00%
19	1050	10.03	1.33%	1066	16.00	2.81%	1112	11.17	6.83%
20	1011	10.81	10.19%	1011	17.87	10.19%	1011	13.71	10.19%
21	584	17.38	5.31%	609	14.51	9.20%	597	29.06	7.37%
22	644	15.23	9.16%	636	11.44	8.02%	632	13.16	7.44%
23	609	12.85	0.00%	609	12.79	0.00%	609	13.56	0.00%
24	594	15.04	0.00%	594	12.92	0.00%	594	14.19	0.00%
25	1197	15.91	4.85%	1183	26.54	3.72%	1161	15.17	1.89%
26	1200	13.37	3.25%	1221	17.96	4.91%	1168	30.81	0.60%
27	763	27.26	5.50%	763	29.78	5.50%	771	15.64	6.49%
28	761	18.44	8.94%	761	16.67	8.94%	761	28.29	8.94%
29	712	18.53	0.00%	712	20.57	0.00%	712	20.05	0.00%
30	739	17.93	0.00%	739	18.19	0.00%	739	19.00	0.00%
31	1422	22.98	7.10%	1422	26.45	7.10%	1393	44.98	5.17%
32	1509	16.17	3.38%	1506	15.78	3.19%	1562	20.10	6.66%
33	976	32.26	10.76%	914	27.19	4.70%	949	45.06	8.22%
34	770	35.28	7.14%	750	26.31	4.67%	803	26.27	10.96%
35	890	20.99	0.00%	890	20.43	0.00%	890	21.32	0.00%
36	893	20.26	0.00%	893	25.25	0.00%	893	20.93	0.00%
max		35.28	10.76%		29.78	10.19%		45.06	10.96%
avg		16.00	3.41%		15.85	3.21%		17.83	3.31%
min		3.60	0.00%		3.60	0.00%		3.28	0.00%

Appendix 4: The computational results for data set C used to determine the number of iterations for the stopping condition

No.	stopping 200	Secs	Gap %	stopping 500	Secs	Gap %	stopping 1000	Secs	Gap %
1	440	1.61	0.23	440	3.60	0.23	440	3.60	0.23
2	444	1.69	0.68	444	4.33	0.68	444	4.33	0.68
3	246	3.99	2.50	244	6.36	1.67	252	6.36	5.00
4	219	2.32	6.83	213	5.23	3.90	211	5.23	2.93
5	285	2.26	0.00	285	5.24	0.00	285	5.24	0.00
6	273	2.20	0.00	273	5.32	0.00	273	5.32	0.00
7	594	3.07	3.85	596	4.86	4.20	580	4.86	1.40
8	506	2.04	1.61	502	4.80	0.80	507	4.80	1.81
9	353	4.83	6.97	367	5.62	11.21	354	5.62	7.27
10	363	2.50	7.40	356	8.71	5.33	340	8.71	0.59
11	375	2.94	0.00	375	6.97	0.00	375	6.97	0.00
12	366	3.17	0.00	366	6.96	0.00	366	6.96	0.00
13	633	2.92	2.43	632	6.38	2.27	634	6.38	2.59
14	720	3.85	9.76	704	7.45	7.32	689	7.45	5.03
15	484	2.93	9.01	457	9.44	2.93	475	9.44	6.98
16	397	3.46	8.17	388	7.41	5.72	406	7.41	10.63
17	431	3.93	0.00	431	9.28	0.00	431	9.28	0.00
18	459	4.13	0.00	459	8.80	0.00	459	8.80	0.00
19	1067	6.46	2.99	1051	16.00	1.45	1066	16.00	2.90
20	947	3.99	4.30	940	17.87	3.52	1011	17.87	11.34
21	624	8.99	12.84	581	14.51	5.06	609	14.51	10.13
22	644	4.79	10.09	628	11.44	7.35	636	11.44	8.72
23	609	5.48	0.00	609	12.79	0.00	609	12.79	0.00
24	594	5.96	0.00	594	12.92	0.00	594	12.92	0.00
25	1163	13.32	2.11	1183	26.54	3.86	1183	26.54	3.86
26	1196	5.71	3.01	1168	17.96	0.60	1221	17.96	5.17
27	827	7.67	14.70	774	29.78	7.35	763	29.78	5.83
28	735	15.44	6.06	768	16.67	10.82	761	16.67	9.81
29	712	8.08	0.00	712	20.57	0.00	712	20.57	0.00
30	739	8.48	0.00	739	18.19	0.00	739	18.19	0.00
31	1399	10.35	5.90	1334	26.45	0.98	1422	26.45	7.65
32	1502	12.64	3.02	1537	15.78	5.42	1506	15.78	3.29
33	945	12.34	8.50	956	27.19	9.76	914	27.19	4.94
34	785	9.64	9.79	761	26.31	6.43	750	66.31	4.90
35	890	8.47	0.00	890	20.43	0.00	890	20.43	0.00
36	893	10.40	0.00	893	25.25	0.00	893	25.25	0.00
max		15.44	14.70		29.78	11.21		66.31	11.34
avg		5.89	3.96		13.15	3.02		15.67	3.43
min		1.61	0.00		3.60	0.00		3.60	0.00