COLUMN GENERATION APPROACH FOR DYNAMIC BERTH ALLOCATION PROBLEM

by

Özge Narin

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

Koç University

August, 2009

Koç University

Graduate School of Sciences and Engineering

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

Özge Narin

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:

_____

Assoc. Prof. Ceyda Oğuz

_____

Asst. Prof. Onur Kaya

_____

Asst. Prof. Sibel Salman

Date: _____

To my family

ABSTRACT

Berth allocation problem (BAP) is to find the best allocation of berths (i.e., sections of the quayside) to the incoming ships at a container terminal and the definition of the problem is usually dictated by the objective function used and the constraints imposed. In this thesis we addressed the well known berth allocation problem called Dynamic Berth Allocation Problem (DBAP). The DBAP involves a set of ships that may not be ready for handling before the berths become available as opposed to the Static Berth Allocation Problem (SBAP). This means that the ships will arrive during the planning horizon which makes the problem much harder than the SBAP because requirements of the problem formulation increase in terms of introducing new variables and constraints to cover this relaxation. The DBAP gets more difficult to solve exactly as the problem size increases (i.e., number of berths and ships increases). In order to tackle this difficulty we proposed a Column Generation (CG) Algorithm for DBAP. CG is a common method to solve large scale integer programming (IP) problems. We first implement CG procedure similar to its application in the literature where the relaxation of the master problem is solved at each iteration and the subproblems are solved exactly to find the schedule (column) that has the minimum reduced cost for the associated berth. The drawback of this approach is the high computation time because the subproblems are Mixed Integer Programming (MIP) models and are solved exactly at each iteration for a number that is equal to the number of berths in the problem instance. In order to decrease the computational burden of the proposed CG algorithm, a heuristic approach is developed for the subproblems. First, we relaxed the subproblem by ignoring the constraints and the variables that complicates the problem. Consequently, subproblem is turned into a simple assignment problem. This assignment problem is solved exactly and the ships that are included in the optimal solution of the assignment problem are used to find a better solution for DBAP's subproblem by including the ignored variables and constraints. This solution is found by a local search algorithm. If this heuristic cannot find a schedule that has a negative reduced cost, the exact procedure is employed to ensure

that there exist no more schedules that has a potential to improve the objective function of the master problem for the berth related to the current subproblem.

The proposed CG algorithm is tested on 84 large scale DBAP instances which are taken from the literature. Results of the small instances are not tested since the computation times needed to find the optimal solution with the exact algorithm are almost negligible in these instances. When the proposed decomposition based CG algorithm compared with the other solution procedures in the literature, it does not perform the best for the entire 84 problem instances. The performance of the proposed algorithm is better than other approaches on the instances that are more difficult than others in terms of the structure of the parameter setting. Solution quality of the proposed algorithm is the same with the exact solution for the instances that are solvable exactly. Moreover, CG algorithm gives better solutions than the Variable Neighborhood Search (VNS) algorithm provided in the literature which gives the best results in terms of computational time for the instances that are not solvable by the exact procedure because of the *out of memory error*.

We conclude that the proposed CG algorithm provides the optimum solution for large size instances which cannot be solved by the exact algorithms. Even though the computational time of the algorithm is large for these instances, as we can obtain exact solutions compared to the heuristic methods, the CG algorithm provides an alternative for the managers of the container terminals with respect to the quality of the solutions.

ÖZETÇE

Bu tezde literatürde yer alan Dinamik Rıhtım Tahsis Etme Problemi (DRTEP) üzerinde çalışılmıştır. Rıhtım Tahsis Etme Problemleri rıhtıma yanaşan gemilerin belirlenen bir amaca göre en iyi şekilde rıhtımda servis edilecekleri noktalara dağıtılmasıdır. Dinamik Rıhtım Tahsis Etme Problemi (DRTEP), Statik Rıhtım Tahsis Etme Problemi'nin (SRTEP) aksine rıhtı mdaki noktalar gemilere servis vermek için hazır olmadan önce gelebilecek gemileri de içermektedir. Bu, gerçek hayatta olduğu gibi, gemilerin planlama ufuğu süresince rıhtıma yanaşabilmelerine olanak sağlamaktadır. Bu esneklik probleme dinamiklik katarken, problemi daha da zorlaştıran gereksinimlere yol açmaktadır. Dolayısıyla, problem örnekleri büyüdükçe DRTEP'ın çözülmesi de daha zor hale gelmektedir. Bu sorunu aşabilmek için büyük ölçekli tam sayı problemlerini çözmekte kullanılan bir kolon üretme algoritması sunuyoruz. Çözüm yönteminde ilk olarak kolon üretme algoritmasını literatürde bilindiği şekilde gevşetilmiş ana problemi indirgenmiş maliyeti eksi olan kolon kalmayana kadar çözdük. Fakat bu yöntem alt problem her rıhtım noktası için ayrı ayrı tam sayı problemi olarak çözüldüğü için çok fazla zaman istemekteydi. İstenilen zamanı azaltabilmek için algoritmanın en çok zaman harcayan kısmına, yani alt probleme, sezgisel bir yöntem uyguladık. Alt problemin bazı değişken ve kısıtlarını yok sayarak problemi basit bir atama problemine dönüştürdük. Bu problemin çözümüne de basit bir yerel araştırma algoritması uyguladık. Bu sezgisel yö ntem indirilmiş maliyeti eksi olan kolon üretemediğinde, alt problem kesin çözümlü olarak tekrar çözüldü. Ayrıca algoritma, indirilmiş maliyeti eksi olan kolon bulamayana kadar devam ettirildiğinde çözüm zamanı çok uzadığından algoritmanın durma kriteri, ana problemin amaç fonksiyon değeri önceden belirlenmiş bir iterasyon sayısı kadar değişmezse algoritmayı durdurmak olarak değiştirildi.

Önerilen kolon üretme yöntemi literatürden alınan 84 büyük problem örneği ile test edildi. Küçük problem örnekleri test edilmedi. Bu problem örneklerinin test edilmeme nedeni ise gerçek çözüm sürelerinin göz ardı edilebilecek kadar kısa olmasıdır. Literatürdeki diğer yöntemlerle karşılaştırıldığında önerilen kolon üretme yönteminin 84 problem örneğinin

hepsi için en iyi çözüm yöntemi olduğu söylenemez. Problem örnekleri büyüdükçe ve parametre yapısı zorlaştıkça önerdiğimiz kolon üretme yönteminin performansı diğerlerinden daha iyi hale gelmeye başlıyor. Önerilen yöntemin çözüm kalitesi çözülebilen problem örnekleri için gerçek çözümle aynıdır. Ayrıca kolon üretme yöntemi literatürde bellek sorunu yüzünden çözülemeyen problemler için en iyi çözüm yöntemi olarak gösterilen Degişken Komşu Arama yönteminden daha iyi sonuç vermektedir.

Sonuç olarak, önerilen kolon üretme yöntemi literatürdeki gerçek çözüm yöntemleriyle çözülemeyen büyük problem örnekleri için en iyi sonucu vermektedir. Kolon üretme yöntemi diğer sezgisel yöntemlerle karşılaştırıldığında çözüm sürelerinin uzunluğuna rağmen, çözüm kalitesindeki farklılıkla konteynır terminali yöneticilerine farklı bir alternatif sunmaktadır.

ACKNOWLEDGMENTS

I would like to thank my supervisor Assoc. Prof. Ceyda Oğuz who has been a great source of inspiration and provided the right balance of suggestions, criticism, and freedom. I acknowledge the computational support provided by the Koç-IBM Supply Chain Laboratory for the computational experiments part of my thesis study.

I am grateful to members of my thesis committee for critical reading of this thesis and for their valuable comments.

Moreover I thank my family and my friends for providing me a morale support that helps me in hard days of my research. Finally I'd like to acknowledge TUBITAK for financial support during my M.S. studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Abbreviations used in text

BAP	Berth Allocation Problem
DBAP	Dynamic Berth Allocation Problem
MIP	Mixed Integer Programming
CG	Column Generation
TEU	Twenty Feet Equivalent Unit
BCP	Branch-and-cut-and-price
LP	Linear Programming
IP	Integer Programming
TST	Total Service Time
VNS	Variable Neighborhood Search Algorithm

Parameters used in models

S_i	Availability time of the berth i
A_j	Arrival time of ship j
c_{ij}	Handling time of ship j at berth i
r_{id}	Cost of column d for berth i
α_i	Dual variable of berth i
β_j	Dual variable of ship j

Variables used in models

x_{ijk}	Binary variable indicates whether ship j is scheduled at position k at berth i, or not.
y_{ijk}	The length of the idle period at berth i before the arrival of ship j that will be scheduled at the k^{th} last position.
z_i^d	Binary variable indicates whether column d of berth is chosen, or not.
a_{jk}	Binary variable indicates whether ship j is scheduled at position k, or not.
w_{jk}	The length of the idle period before the arrival of ship j that will be scheduled at the k^{th} last position.

Abbreviations used in algorithms

Algorithm 1

$kmax$	Maximum number of swaps between 2 random berths and 2 random ships in one iteration.
$kmin$	Minimum number of swaps between 2 random berths and 2 random ships in one iteration.
$maxIter$	Maximum number of iterations.
$NoOfShips$	Number of ships in the problem instance.
$NoOfBerths$	Number of berths in the problem instance.
$fbest$	Best objective function found during the algorithm.
$CostOfTheInitialSchedule$	Cost of the initial schedule found by the ordereded assignment of the ships.
fs	Objective function of the temporary schedule found during the iterations.
$CostOfTheNewSchedule$	Cost of the schedule found from swap of the ships.

Algorithm 2

$NearZero$	A parameter which is very close to zero and also negative.
$NumOfIterWithoutImp$	Counts the number of iterations that have same objective function value consequtively.
$NoOfShips$	Number of ships in the problem instance.
$NoOfBerths$	Number of berths in the problem instance.
$MaxNumOfIterWithoutImp$	Maximum number of iterations that have same objective function value consequtively.
$Schedule$	Schedule found from subproblem i
$ObjFunSub$	Objective function of the schedule found from the subproblem.
$ObjFunMas$	Objective function of the master problem that is solved with the columns on hand.
$FinalSchedule$	Final schedule found by algorithm.
$IntegerSolution$	Schedule found by solving the RMP as an integer programming problem at the end of the algorithm.

Algorithm 3

$NearZero$	A parameter which is very close to zero and also negative.
$NumOfIterWithoutImp$	Counts the number of iterations that have same objective function value consequtively.
$NoOfShips$	Number of ships in the problem instance.
$NoOfBerths$	Number of berths in the problem instance.
$MaxNumOfIterWithoutImp$	Maximum number of iterations that have same objective function value consequtively.
$Schedule$	Schedule found from subproblem i
$NumOfExactSub$	Algorithm changes the subproblem solution procedure to exact solution for at most $NumOfExactSub$
$NumShips$	Number of ships scheduled in $Schedule$.
$MaxNoTrial$	Maximum Number of trials for swapping the positions of 2 ships of $Schedule$ to find a new schedule ($NewSchedule$).
$ReducedCost$	Reduced cost of the schedules.
$BestSchedule$	Best schedule found tduring the algorithm.
$FinalSchedule$	Final schedule found by algorithm.
$IntegerSolution$	Schedule found by solving the RMP as an integer programming problem at the end of the algorithm.

Chapter 1

INTRODUCTION

Containers are large boxes that are used to transport goods from one destination to another. Using containers has several superiorities over using conventional methods, such as, strength and durability (i.e., they are designed to carry heavy loads when they are stacked in higher columns), modularity (i.e., all shipping containers are made to the same standard measurements and as such they provide modular elements that can be combined into larger structures), transformation (i.e., pre-fabricated modules can also be easily transported by ship, truck or rail, because they already conform to standard shipping sizes), availability (i.e., used shipping containers are available across the globe), and cost. In this context, the term twenty-feet-equivalent-unit (TEU) is used to refer to one container with a length of twenty feet. A container of 40 feet is expressed by 2 TEU. This way of unit representation simplifies the tracking of goods during the transportation.

Several transportation systems can be used to transport containers. Sea-borne transportation is carried out by ships and trucks or trains can be used to transport containers over land. Terminals or ports are used to transmit containers from one mode of transportation to another. For instance, a container can be taken off a ship and placed on a train at a container terminal or the other way around. Containers were first used in mid-fifties [2]. Proportion of cargo handled by containers has steadily increased through the years. Growth of container traffic over the world can be seen in Table 1 [4]. A further continuous increase is expected in the upcoming years, especially between Asia and Europe [5]. Hence, the importance of the container terminals has grown accordingly. Efficiency and productivity improvements in terminal operations result in cost and time effective handling of the ships and containers as expected. Therefore, efficient management of container terminals has gained significant attention in recent years.

This thesis concentrates on efficient berth allocation that finds out an assignment of

Table 1.1: Growth of container traffic over the world

Worlwide	2005	2006	2007
1 Singapore	23,190,000	24,800,000 (+6.94%)	27,932,000 (+12.63%)
2 Shanghai	18,084,000	21,700,000 (+20.00%)	26,150,000 (+20.51%)
3 Hong Kong	22,602,000	23,230,000 (+2.78%)	23,881,000 (+2.80%)
Europe	2005	2006	2007
1 Rotterdam	9,287,000	9,690,000 (+4.34%)	10,790,000 (+11.35%)
2 Hamburg	8,087,550	8,861,804 (+9.57%)	9,900,000 (+11.72%)
3 Antwerp	6,482,030	7,018,799 (+8.28%)	8,176,614 (+16.50%)

ships that are going to be handled at berths in a container terminal for cargo handling. Quick turnaround of the ships is the main goal of the container terminal management. This aim is also related to the customer satisfaction. Fast transmit time from an origin to a destination is the primary issue for the shippers (i.e., customers). Any delay in a transshipment port results in a significant delay for the overall schedule of the ship and containers may loose their planned connection if they are planned to be transmitted to other ships or other modes of transportation. Therefore, punctual schedule is the main goal of both the shipping companies and the container terminals. This objective is achieved by minimizing the total completion time of the ships (i.e., waiting time and the handling time of the ships). Any delay is also considered by this objective implicitly since this objective procures quick handling of the ships generally (this claim will be elaborated more in Chapter 3 by Theorem 1).

In most of the situations, planning horizon of the BAP is one week but the berthing plan is updated every day. Some berths may not be available in some time when the problem is reoptimized because of the rolling horizon. Moreover, there can be another availability restriction on the quay due to the maintanence operations. Availability situations of the berths are integrated to the BAP by assigning an availability parameter for each berth. Moreover, every ship has its own time window starting from its arrival time, which is known in advance, to its departure time. Container terminals can be managed efficiently when both the user costs and the port costs are minimized that are associated with the service time. That is why the objective of BAP is determined most of the time as minimizing

the total service time of the incoming ships. Different weights can be given to each ship if the importance of the incoming ships is not the same. In that case, sum of the weighted service times must be reflected in the objective function. These weights can be determined to constitute a pricing scheme or to define the number of containers to be moved. In some situations, a penalty term can be included in the objective function of BAP to satisfy the contracted departure time of the ship if it exists. If this is the case, implicit consideration of the delay of the ships cannot be accomplished (explanation is given by Theorem 1 in Chapter 3).

In this thesis, we contribute to the solution of the well known Dynamic Berth Allocation Problem (DBAP) by implementing a column generation (CG) algorithm. Although the technique of CG has been around since the early sixties, it has not been applied to a BAP before. However, there are a certain number of CG applications on shortest path problems (SPP) [6], vehicle routing problems with time windows (VRPTW) [7], cutting stock problems (CSP) [8], maritime transportation applications such as, large scale models in airline industry [9], inventory ship routing [10], ship scheduling with recurring visits and visit separation requirements [11] and production environments such as, machine scheduling problems [12]. These applications result in different levels of success. Since DBAP has some similarities with these problems, CG algorithm seems to be a promising method for it.

We worked on a compact Mixed Integer Programming (MIP) formulation of DBAP proposed in [13] and in [14]. We split this compact MIP formulation into two formulations, such as master problem and subproblems (pricing) where the master problem formulation constitutes a set partitioning problem. These problems are interrelated in a way that subproblem uses the dual variables that are derived from the master problem. Besides, optimum solution that comes from the subproblem is added to the master problem as a new column for the berth which the subproblem is solved for. This procedure continues until optimum objective function value of the subproblem is no more negative, which means that there exists no new column that has potential to improve the objective function of the master problem (i.e., reduced cost of the new variable (column) is not negative). Since the berths of the DBAP do not have the same characteristics (i.e., not identical), the optimum solution of the subproblem is different for each berth. Subproblem must be solved individually for each berth and solutions of the subproblems must be added to the columns of the associated

berth. Hence, we need to be sure that no columns with negative reduced cost exists for all berths to terminate the CG algorithm.

The CG approach described in the above paragraph can be taken as an exact procedure for DBAP because it continues until no more potential schedule exists for all berths. This property is desirable in terms of the solution quality but it requires long computation times, which is a common drawback of CG algorithms. Subproblems are the major contributors to this large computation time because they are MIP models and they are solved exactly. When the size of the problem instances has grown, the computational effort needed to find an optimal solution for a single berth increases. Thus, we focus on the subproblem to decrease the computation time of the CG algorithm. We propose a heuristic, which first solves the corresponding assignment problem for the subproblem. Afterwards, optimal solution of the assignment problem is improved by a local search algorithm, which is a modified version of the VNS algorithm proposed in [13]. This heuristic first finds the objective function value of the optimum solution found by the assignment problem corresponding to DBAP. After that, it continues by swapping the positions of the randomly selected ships scheduled at the current berth. Then, it deletes the randomly selected ships from the schedule of the current berth and swaps the positions of the rest. When the algorithm finds a solution better than the solution taken from the assignment problem in terms of the objective function value, it keeps the solution as the final solution. If this heuristic fails to find a schedule with negative reduced cost, heuristic approach is altered with the exact solution to ensure that there exists no schedule with negative reduced cost. If exact solution finds any schedule that has negative reduced cost, solution procedure is turned into heuristic approach again.

In this thesis, we will develop a solution approach for DBAP that will be a good tool to find better solutions in terms of the solution quality. In doing this, we will address the larger problem instances because they cannot be solved by exact procedures. In the remaining part of this chapter, we will give a brief summary of our contribution and the flow of the thesis. This study will contribute to the literature by being the first study of CG algorithm on BAPs. Accordingly, it can be a good tool for the new studies of BAPs and other problems that are similar to BAPs. Moreover, we will develop a heuristic model for the subproblem to decrease the computation time of the CG algorithm. It can also be used as a solution instrument for the problems that have a similar structure with the subproblem

(i.e., machine scheduling problems). CG approach used in this thesis is a successful solution approach for DBAP in the way that it provides better solutions than the best solution approach given in the literature so far especially for the instances that are not solvable by exact procedures. The DBAP instances larger than 10 berths - 45 ships cannot be solved exactly because of the *out of memory* error. CG algorithm that will be proposed in this thesis gives the solutions that has the best solution quality for the instances larger than 10 berths - 45 ships compared to the literature.

The outline of this thesis can be summarized as follows. Literature review of BAPs and CG algorithms is given in Chapter 2. Technical background of the DBAP and mathematical formulations is provided in Chapter 3. CG approach developed for the DBAP is explained in Chapter 4. Details of the master problem and subproblem are presented in Section 4.1 and Section 4.2, respectively. CG algorithm provided for DBAP is given in Section 4.4 as a whole. Details of the CG implementation is given in Section 4.5. Finally, we conclude this thesis by Chapter 5.

Chapter 2

LITERATURE REVIEW

2.1 Berth Allocation Problems

Berth allocation problem (BAP) tries to find the best allocation of berths (i.e., sections of the quayside) to the incoming ships at a container terminal. A typical container terminal can be seen in Figure 2.1. When the ships arrive at a port, they enter in the harbor waiting to moor at the quay. The main elements in the layout of a container terminals are given in Figure 2.2. The quay is a platform that projects into the water to alleviate the loading and unloading of cargo. The locations where mooring can take place are called berths. Berths are equipped with giant cranes, called quay cranes which can be seen in Figure 2.3. Quay cranes facilitate loading and unloading of containers. The containers are transferred to and from the yard by a fleet of vehicles. If container terminal is a transshipment terminal, the yard allows temporary storage before containers are transferred to another ship or another mode of transporatation (i.e., railway or road). The process of loading and unloading operations can be observed in Figure 2.4.

Two interrelated decisions are faced while assigning incoming ships to berth positions which are *where* and *when* the ships should be moored. *Where* corresponds to the berth that is chosen for the ship to be moored and *when* corresponds to the order of the ships that are to be scheduled at the berth chosen. Berthing point and the distance between the berth and the pick-up delivery area of a container stored in the yard affect the handling time of a ship, which in turn determines the performance of a container terminal. Handling time also depends on another associated decision which is the number of assigned quay cranes to the ships. This decision is treated by the Quay Crane Assignment Problem (QCAP). Number of cranes assigned to a ship not only affects the handling time, but also creates an impact on the solution of BAP. In a complex system, like a transshipment port, the decision making process is often hierarchical and the QCAP is solved before the BAP. In fact, decisions about the QCAP are subject to less flexibility since the terminal must achieve

Figure 2.1: Picture of a Container Terminal

a contractual performance level. Moreover, the number of quay cranes assigned to each ship is chosen according to practical rules that consider the ship length and its priority. As there are several criteria that influences the handling time of a ship, a more detailed analysis is required in real world applications.

The BAP can be modeled as a discrete problem if the quay is regarded as a finite set of berths and as a continuous problem if quay is represented as a continuous line. Here the objective is the minimization of the total service time for all ships which is defined as the sum of the elapsed times between the arrival of each ship to the harbour and its completion of handling. For each ship, time elapsed in the harbor is also referred as the turnaround time of the ship. Therefore, assigning berths to incoming ships for their cargo handling plays an important role in minimizing the turnaround time. The main reason for this is the nonidentical berths, so that the handling time for a specific ship being not necessarily the same at every berth, that is berths have different characteristics that change handling

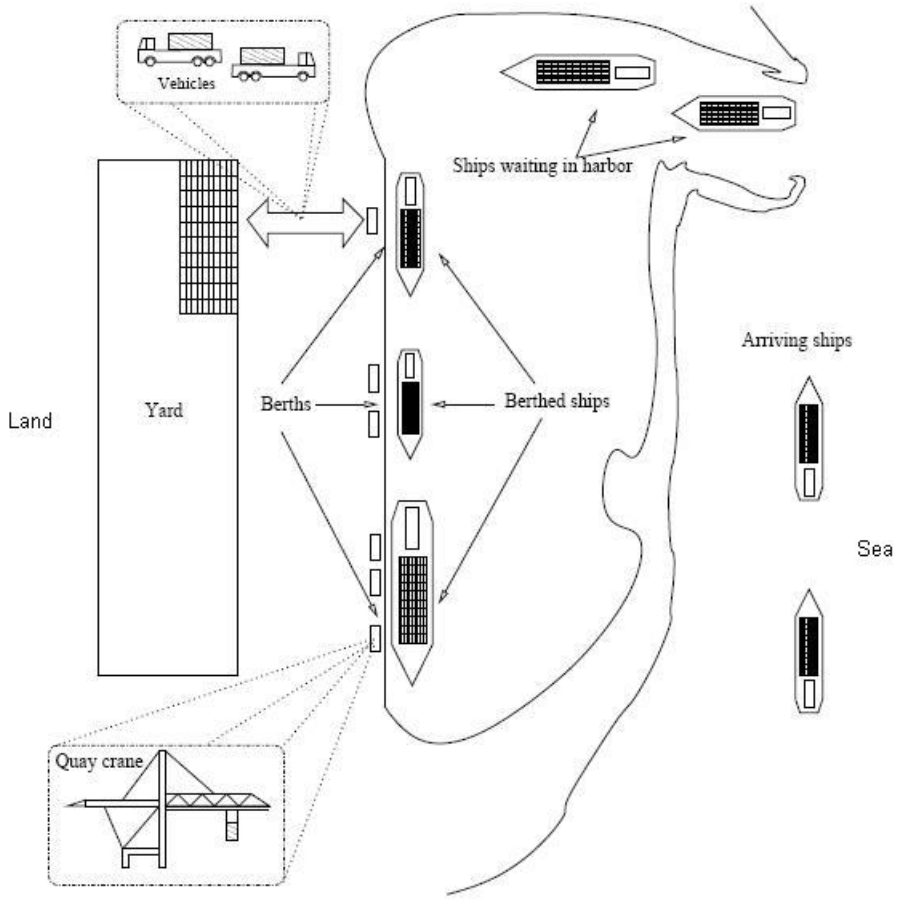

Figure 2.2: Container Terminal Layout [1]

time of a ship as a consequence. Therefore, the BAP can be modeled as a discrete or a continuous problem.

For the discrete case, berths are described as fixed length segments because when ships have different lengths, dividing the quay by a set of berths, which have different lengths is difficult to handle due to the dynamic variations among requirements of ships. By this way, spatial dimension of the berths can be discarded and berths are treated as points. This problem can be represented in two-dimensional space where one dimension is berths and other dimension is the time. In the discrete case, BAP can be modeled as unrelated parallel machine scheduling problem where ships are treated as jobs, berths are machines and, arrival times of the ships are the release times of the jobs [15]. NP-hardness of this

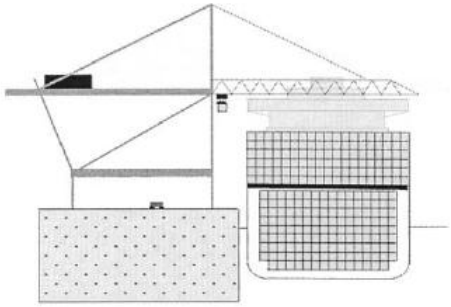

Figure 2.3: Quay crane [2]

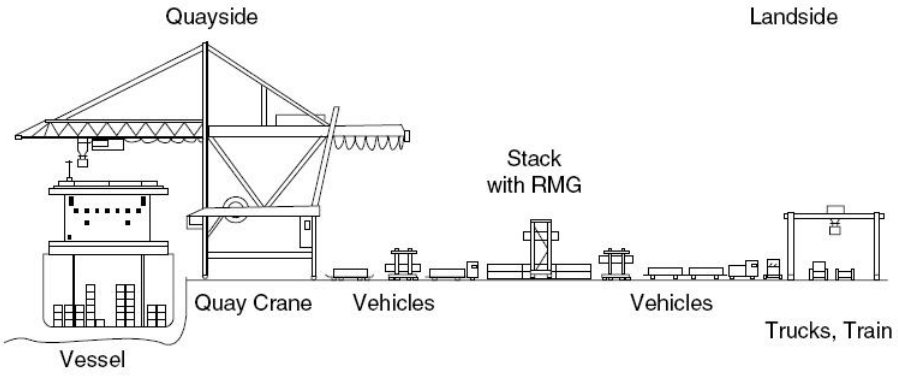

Figure 2.4: Process of loading and unloading operations of a ship [3]

problem is proven in [16] and consequently BAP becomes NP-hard as well. In addition, there can be other constraints relative to the water depth or the length of the ship.

Continuous BAP is used in cases where ships can be moored across the berth boundary because of the ship length. In these cases, BAP should be solved without using the berth. Furthermore, continuous BAP has a significant flexibility for the berth allocation but this advantage is offset by the difficulty in solving the problem due to its complexity. In the literature, this problem is addressed as static BAP in most of the cases in order to deal with a simplified problem. Dynamic case of BAP is studied in [17] where authors utilize the similarity between dynamic continuous space BAP and the two dimensional cutting stock problem.

BAP can also be modeled as static or dynamic as mentioned in the previous paragraph.

In Static Berth Allocation Problem (SBAP), a set of ships are given at the beginning of the period that are ready to be handled before the berths become available [18]. If this assumption is relaxed, SBAP turns into Dynamic Berth Allocation problem (DBAP) [19]. In DBAP, the ships may arrive before or after berths become available and this property makes BAP dynamic. As a result, BAP becomes harder since finding the optimum schedule results in a larger solution space. As a matter of fact, SBAP is solvable in polynomial time with the Hungarian method [20] since it is reducible to the assignment problem. Similarly, an appropriate Lagrangean relaxation procedure for the DBAP is also proposed in [20] where the subproblem is an assignment problem. Computational results of this approach points out that DBAP becomes easy to solve when the instances are similar to the SBAP which means most of the ships arrive before berths become available. On the other hand, in [21] a DBAP is considered where the quay is taken as a collection of berths with the assumption that up to two ships can be handled at the same berth if their lengths are compatible with the length of the berth segment. They modeled this problem as a non-linear integer program as they incorporated additional constraints related with the water depth and proposed a genetic algorithm for its solution due to the intractibility of the model.

An extended formulation for DBAP is presented in [19] that takes service priorities of the ships into account by introducing a term to the objective function, which is related to the service times of the ships. In this work, an appropriate Lagrangean relaxation is proposed for the resulting non-linear formulation which has a subproblem that becomes a quadratic assignment problem. Quadratic assignment problem is not solved well with exact solution methods so they have developed a generic heuristic. In [22], the multiple ship treatment in the discrete BAP, like [21], is applied to the indented terminal with a simple procedure to deal with small ships at indented berths. Also, comparisons in terminal performances between conventional and indented terminals are presented in this work. Allocating some ships to other container terminals because of capacity limit of an extremely busy container terminal in a developing country is considered in [23].

On the other hand, corrections to SBAP and DBAP models given in [19] and [24] were introduced in [13]. The idea of reverse order for the position of the ships at a berth is used in [13] in contrast to [19] where direct order is used. In addition, constraints given in [19] are not sufficient to ensure that a ship is not going to be handled at a given berth in the k^{th}

first position while no ship is handled in the $(k-1)^{st}$ position at that berth. Therefore, [13] mentioned that in order to solve this inaccuracy, additional constraint set should be added to the models proposed in [19] or reverse order of ships can be used. An extended case of [13] is proposed in [14] as Minimum Cost Berth Allocation Problem (MCBAP). Instead of reflecting handling times of the ships to the objective function, they reflect handling costs, which depends on the berth used (number of cranes used) and the number of containers that is transmitted. Earliness and tardiness of the ships are included as premiums and penalties. These premiums and penalties are weighted to express the fact that delays are more or less important according to both the size of the ship and the commitments of the ship owners' company, particularly, the scheduled arrival date at the next visited terminal.

Following these studies, continuous case of the BAP is studied in [25] by representing quay as a continuous line. In this paper, handling times are assumed to be constant and a heuristic procedure is developed to decide the berthing points given the berthing time of the ships. However, this approach cannot solve the general problem in which the berthing time is a decision variable and the handling times vary from berth to berth. A non-linear IP model that also considers the quay crane assignment problem (QCAP) is introduced in [26]. Main asssumption that integrates BAP and QCAP is that the handling times vary linearly with the number of cranes assigned to a ship. Authors assumed that the optimal berthing point is known and if different berthing point is chosen, they apply a penalty over that ship. Their objective function is to minimize the total penalty over all ships. Thus, lagrangean relaxation and a subgradient optimization method are used as a solution procedure in this study.

In recent years, metaheuristics are used as a solution method for most of the studies. Some of these studies can be summarized as follows. Variable Neighborhood Search (VNS) algorithm is used in [13] to solve Minimum Cost Berth Allocation Problem (MCBAP) and DBAP. This study has already mentioned above in this section. Genetic algorithm is used in [27] to develop an efficient heuristic. They addressed the problem of determining the berthing position and time of each ship as well as the number of quay cranes assigned to each ship and the objective function is determined as minimizing total service time, waiting time and delay time for every ship. In [28], GA is used to develop an appropriate heuristic for simultaneous berth and crane allocation problem. This objective is almost the same

with the objective used in [27]. Another method to consider guay crane assignment with berth allocation is solving the Quay Crane Assignment Problem (QCAP) before BAP. This method is used in [4] with additional depth of water and time windows on completion time constraints. The common point of these recent studies is to integrate QCAP and BAP in order to make the solution more realistic.

2.2 Column Generation

A wide range of real-life optimization problems can be formulated as maximizing or minimizing a linear function of variables subject to equality or inequality constraints and some variables are restricted to integers. Such problems are modeled as mixed integer programs (MIP). In order to find a successful solution to large-scale MIP problems, their linear programming (LP) relaxations should give a better approximation to the convex hull of feasible solutions of MIP. These approximations are used as bounds to carry out a technique that uses an implicit enumeration of all feasible solutions called branch-and-bound. The quality of the approximations (bounds) is critical for the efficiency of the algorithm. Bounds are provided by LP relaxations of the MIP formulation in conventional branch-and-bound algorithm. Solving MIP formulation by relaxing the integrality constraints is easy but the quality of the bound is often poor. In order to provide better bounds, other types of approximation algorithms are used. Lagrangean relaxation relaxes some of the constraints of a given formulation of the problem as opposed to all integrality constraints. Polyhedral approach improves LP relaxation by adding inequalities to strengthen the LP formulation.

An alternative way to branch-and-bound algorithms is decomposing the constraint set of the problem formulation. Consider a simple integer problem given as

$$\text{Min } cx \tag{2.1}$$

s. t.

$$Ax = b \tag{2.2}$$

$$Dx \leq d \tag{2.3}$$

$$x \text{ integer} \tag{2.4}$$

which has constraints that can be partitioned into a class of global constraints (2.2) and a class of specific constraints referred as a subsystem (2.3 and 2.4). All the solutions of the subsystem of constraints can be enumerated and problem can be reformulated in terms of subsystem solutions by performing a variable transformation. First situation for the natural usage of this decomposition approach is when the optimization problem is easy to solve over subsystem of constraints (i.e., constraint set (2.3) constitutes a well-known polyhedron) but global constraints complicate the problem. Second situation is reached when $Dx \leq d$ have a block diagonal structure in which the subsystem can be decomposed into K subsystems.Therefore, K subsystems become independent from each other without global constraints (2.2) regarded as linking constraints in this case. Whether the integer program with a large number of variables results from decomposition or not, it is referred to as *master problem* by reference to Dantzig-Wolfe decomposition algorithm in LP.

Branch-and-cut-and-price (BCP) algorithms are actually branch-and-bound algorithms where dual bounds are obtained by solving a linear program with huge number of rows and columns. Therefore, cut generation (seperation) and column generation (pricing) must be performed during the execution of the algorithm. It is mentioned in [29] that such algorithm is *robust* when structures of the separation and pricing subproblems continue to be unchanged during the execution of the algorithm. First BCP algorithm is applied on edge-coloring problem by [30] but this algorithm is not robust. The new columns added to the master problem complicate the pricing subproblem so it becomes costly to be solved. In late 90's, researchers argued that structure of the subproblem will remain unchanged by separating generated cuts from the original formulation. As a consequence, cuts are generated first and translated to columns. Afterwards, these columns are added to the master problem. This idea allowed the construction of robust BCP algorithms in the subsequent studies (i.e., [31], [32], [33], [34], [35]) .

There are several reasons for formulating the problems with huge number of variables as stated in [36]:

- Compact formulation of MIP may have a weak LP relaxation. Relaxation can be reformulated with a huge number of variables to be tightened.

- Compact MIP may have a symmetric structure that causes branch-and-bound algo-

rithm to perform poorly. Reformulation with huge number of variables can handle this problem.

- Column generation decomposes problem into master and subproblems. This decomposition may have an interpretation that allows for incorporation of additional important constraints.

- Formulation with huge number of variables can be the only choice.

There are fundamental difficulties in applying LP column generation techniques in IP solution methods as stated in [36]:

- Conventional IP branching on variables may not be effective because fixing variables can destroy the structure of the pricing problem.

- Solving LPs and these subproblems to optimality may not be efficient in which case different rules will apply for managing the branch-and-price tree.

Master reformulation of integer program is beneficial because its LP relaxation provides a tight bound on the integer solution value and it also takes away the difficulties that come from the inherent symmetry of the problem considered. Master reformulation proposes a two level decomposition with an aggregated level which is the master problem, and a disaggregated level which is the subproblem that determined with the subsystem of the constraints (2.3) of the compact MIP formulation.

In general, decomposition principle is presented by Dantzig and Wolfe in 1960 [37]. This decomposition algorithm increases the number of applications of CG to solve large scale linear programs and become a standard LP solution methodology. This decomposition principle is also mentioned in various textbook and publications such as, [38], [39], [40] and [41]. Actually, Dantzig-Wolfe decomposition is introduced to solve large scale LP problems in computers with limited core storage capacity. Its aim is to complement simplex method by extending the applicability range of LP. When Dantzig-Wolfe decomposition is applied to linear programs, it performs by generating an equal *master* problem having only a few more rows than the linking constraints (2.2) but having many more columns.

Even though the Dantzig-Wolfe decomposition principle was introduced for LP, it is also used in IP to obtain good approximations. In this approach, an LP reformulation (master problem) was offered because it results in a generally tighter relaxation than the standard linear relaxation. When decomposition is applied to an integer program, an integer master problem is obtained with a large number of columns. The linear relaxation of the master problem is solved by CG algortihm. Subproblem is also an integer program in all IP applications of CG algorithm. The master IP reformulation that includes all possible columns and the initial compact IP formulation is equivalent so that solving the master IP resulting from the decomposition is equivalent to solving the original IP formulation in terms of finding the optimal solution.

Let p_q, $q \in Q$, be the set of the extreme points of the polyhedron $x \in \Re_+^n : Dx \leq d$ which is constructed by Constraint set (2.3) where Q is the set of possible columns. Then the equivalent master linear program of the integer program formed with equations (2.1),(2.2),(2.3) and (2.4) is given as

$$\text{Min} \sum_{q \in Q} c_q \lambda_q \tag{2.5}$$

s. t.

$$\sum_{q \in Q} a_q \lambda_q = b \tag{2.6}$$

$$\sum_{q \in Q} \lambda_q = 1 \tag{2.7}$$

$$\lambda_q \geq 0 \qquad \forall q \in Q \tag{2.8}$$

where $c_q = cp_q$, $a_q = Ap_q$ and λ_q's are in between 0 and 1 that denoted the selection percentage of column q.

Briefly, the idea of CG algorithm is to deal with a subset of variables (columns) and generate missing variables if and when it is necessary. At every iteration of CG algorithm, restricted master problem (RMP) is solved with a subset $\tilde{Q} \subset Q$ of columns (i.e., the basic feasible solution plus eventually a few other columns). New column, q, is introduced to the RMP as λ_q by augmenting $\tilde{Q}$. The purpose of the RMP is to provide dual variables to be used in the subproblem to price out new non-basic variables to enter the basis. Algorithm

gathers a primal feasible solution for the compact formulation of the problem (MIP) in the end.

The IP CG algorithm is an exact optimization procedure that integrates CG with branch-and-bound technique. The bounding plan which is used in pruning the branch-and-bound tree is based on the linear relaxation of the master problem. The CG algorithm is used in each node of the branch-and-bound tree to solve the linear relaxation of the master problem because of the huge number of columns in the master problem. Subproblem is adjusted appropriately to make branching plan compatible with the CG algorithm.

The CG algorithm is a primal method that ensures the primal feasibility and works towards the dual feasibility. Dual point of view gives most valuable insight into the functioning of the algorithm because the dual solution of the RMP directly affects the selection of the new columns as stated in [42]. The CG algorithm starts with a known basic feasible solution of the linear relaxation of the master problem. This initial solution can be generated by formulating the problem with appropriate artificial basic variables and using the CG to solve this augmented problem with an artificial objective function which penalizes the presence of the artificial variables in the basis. An alternative way to find an initial feasible solution is to use heuristics. A feasible solution is found for the compact MIP formulation and the solution is transformed into new variables (columns) for the linear relaxation of the RMP. This feasible solution constitutes the begining primal basis in order to take the first dual variables needed to find new columns for the primal basis.

The algorithm continues by pricing out the variables (columns) that do not exist in the subset $\tilde{Q}$. After solving the linear relaxation of the RMP, values of the dual variables are gathered and passed to the subproblem. Subproblem uses the dual variables to find a new column that has a potential to improve the objective function value of the master problem by minimizing the reduced cost of the new column. If the reduced cost of the column found from the subproblem has a negative reduced cost, it has a potential to improve the objective function of the master problem when the MIP is a minimization problem. The new column found from the subproblem is not added to the master problem if it has nonnegative reduced cost. The new RMP can be solved with any LP solver and the entire procedure can be repeated until no more columns found by the subproblem with negative reduced costs. Satisfaction of this condition ensures the optimality because this means no more solution

exists that has a potential to improve the objective function value of the master problem (i.e., objective function of the DBAP). Afterwards, the RMP is solved as an integer program by turning the column variables into integer variables as opposed to the linear relaxation case. This last solution gives the optimal integer solution to the problem.

Chapter 3

BERTH ALLOCATION PROBLEMS

BAP determines an efficient assignment of incoming ships to the berths in the container terminal before loading and unloading operations (cargo handling). The most realistic objective might be indicated as the customer's satisfaction (i.e., the incoming ship's satisfaction). Transit time of the ships from origin port to the destination port is the first concern for the ship owners and the companies that wait for the consignees in container shipping. In order to increase the customer's satisfaction, ship departure needs to be kept on schedule. Any delay in departure results in a late arrival for the next port and late departure from that port consecutively. If these delays continue, ships suffer from a substantial delay at the final port but the most important point is that those containers that planned to be transhipped to other vessels might loose their scheduled connections. Thus, punctual departure schedule is the most intended goal in shipping lines for the shipping companies. Significance of the BAPs for efficient management of the container terminals is stated in [43] and, [44]. In order to be consistent, we defined our objective in this study as to minimize the total service time of the ships which is the sum of total cargo handling and waiting times. This objective also handles the minimization of the total delay times of the incoming ships and this is explained by Theorem 1 in Section 3.1.

3.1 Problem Description and Mathematical Models

In this section, we will present problem description and the mathematical model for SBAP and DBAP following the notation in [13]. As mentioned in Section 2.1, SBAP is the simplest form of the berth allocation problems. Assumptions made through SBAP are as follows [13]:
(a) Set of berths is B and indexed by i=1,2,...,I and berths are available after time S_i.
(b) Set of ships is denoted by V and indexed by j=1,2,...,T and ships have arrival times A_j.
(c) Each berth can handle one ship at a time or can remain idle for some time.
(d) Any ship j can be handled at any berth i, with a handling time c_{ij}.

(e) Total completion time, i.e, waiting and handling time for all ships is to be minimized.
(f) All ships arrive before any berths becomes available, i.e.,

$$\max_j A_j \leq \min_i S_i \tag{3.1}$$

Due to assumption (f), all ships are ready to be handled by all berths. This situation provokes a problem environment in which ships can be scheduled to berths consecutively without any idle times between ships. If this assumption does not hold, possible idle time between consecutively scheduled ships should be taken into account (as in DBAP). Since ships have to arrive before berths become available, SBAP model can only be applied to short-term models. Moreover, SBAP is deterministic and fractional assignments such as changing a ship's berth during loading or unloading operations is forbidden.

Since the direct order of the ships requires extra constraints to be sure that no ship is assigned to order k when there is no ships assigned to position $k-1$ (as mentioned in Section 2.1), reverse order of ships is used in the model. Binary variables x_{ijk} is introduced to show which ship j is handled at berth i.

$$x_{ijk} = \begin{cases} 1 & \text{ship } j \text{ is scheduled at position } k \text{ at berth } i \\ 0 & \text{otherwise} \end{cases}$$

An index k is used for assigning the reverse order at which ships will be handled at each berth such that $k \in O$, where O is the set that contains the information of the position of the ships (i.e., order of the ships). One should notice that this reverse order is not known. Now it is easier to see that the total completion time of all ships handled at berth i will be

$$\sum_{j \in V} \sum_{k \in O} (S_i - A_j) x_{ijk} + \sum_{j \in V} \sum_{k \in O} k c_{ij} x_{ijk}, \tag{3.2}$$

where the first term is for calculating the waiting times of all ships handled at that berth before the berth becomes available and the second term is the sum of the handling time for all such ships and the waiting time of the $k-1$ ships still to be handled while the k^{th} ship is being attended to. This situation is illustrated in Figure 3.1, where W_j represents the waiting time, H_j represents the handling time of the ship j, $j = 1, 2, ..., T$, and TCT denotes the total completion time. The SBAP, which is formulated as a more general model than the one given in [18], is presented in [13] as follows:

(SBAP)

$$\text{Minimize} \sum_{i \in V} \sum_{j \in V} \sum_{k \in O} (kc_{ij} + S_i - A_j) x_{ijk} \tag{3.3}$$

subject to

$$\sum_{i \in B} \sum_{k \in O} x_{ijk} = 1 \qquad \forall j \in V, \tag{3.4}$$

$$\sum_{j \in V} x_{ijk} \leq 1 \qquad \forall i \in B, k \in O, \tag{3.5}$$

$$x_{ijk} \in 0, 1 \qquad \forall i \in B, j \in V, k \in O. \tag{3.6}$$

The ships will be assigned to consecutive positions at each berth since the coefficients of the variables x_{ijk} in the objective function (3.3) decrease as k increases. It is easy to see that ships will be handled at each berth in increasing order of their handling time, c_{ij} in the optimum solution. This problem can be turned into a two-dimensional assignment problem and easily solved by the polynomial algorithm presented in [45] or the "auction" algorithm stated in [46]. It is also noted in [18] and [13] that this problem is a particular case of a scheduling problem, i.e., minimizing the total completion time for independent tasks on unrelated parallel machines. A solution to this problem is also given in [47].

In the model SBAP, constraint set 3.4 ensures that every ship is scheduled just once, constraint set 3.5 ensures that at most one ship must be scheduled at any position k and constraint set 3.6 indicates that x_{ijk} is a binary variable.

Dynamic Berth Allocation Problem (DBAP) is arisen when the assumption (f) of SBAP is relaxed, i.e., some ships may arrive before some or all berths become available. If this problem is considered with a single berth, it reduces to minimizing the total completion time with release dates on a single machine and it is proven to be NP-hard in [48]. The first come, first served rule seems to be adopted for this problem, which prevents overpassing between ships. This rule does not give the optimal solution for the objective of total completion time. This situation can be explained on a simple example that is represented in Figure 3.1. In this example, there are 2 ships with A_1=0, A_2=1, c_{11}=10, c_{12}=1 and S_1=0. Start times of the ships are denoted by s_{ik} where index i is used for berth and k is used for *order*. If ship 1 is scheduled first, its start time will be s_{11}=0, and then ship 2 is scheduled on time such

that s_{12}=10, and the total completion time is 0+10+10+1=21. If ship 2 is scheduled first then, s_{11}=3, s_{12}=1, and total completion time is 1+1+2+10=14. This example indicates that in order to minimize the total completion time, berths can be kept idle even if there exists a ship that has already arrived at port.

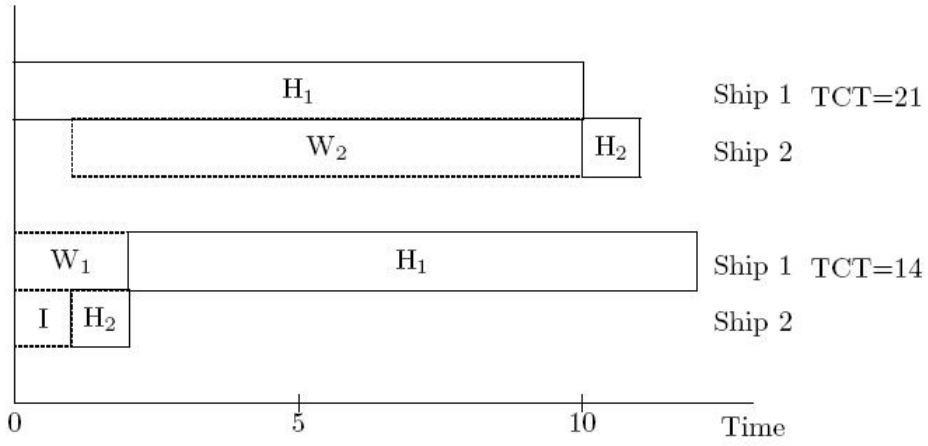

Figure 3.1: A DBAP example where overpassing between ships is beneficial and the berth may be kept idle even if a ship is waiting (idle time is represented by I).

If the case $A_j > S_i$ holds for some ships and berths, possibility of idle periods for berths should be taken into account. This situation is included in [19] by introducing a continuous variable y_{ijk} representing the idle period at berth i before the arrival of ship j, which will be handled at position k. Additional constraints are also included to the model for giving the right values to these variables. After the corrections mentioned in Section 2.1 applied to the DBAP model proposed in [19], continuous variable y_{ijk} is defined as:

y_{ijk} = the length of the idle period at berth i before the arrival of ship j that will be scheduled at the k^{th} last position.

So the mathematical model for DBAP is given as follows:

(DBAP)

$$\text{Minimize} \sum_{i\in B}\sum_{j\in V}\sum_{k\in O}(kc_{ij} + S_i - A_j)x_{ijk} + \sum_{i\in B}\sum_{j\in V}\sum_{k\in O} k y_{ijk} \tag{3.7}$$

s. to

$$\sum_{i\in B}\sum_{k\in O} x_{ijk} = 1 \qquad \forall j \in V, \tag{3.8}$$

$$\sum_{i\in B}\sum_{k\in O} x_{ijk} \leq 1 \qquad \forall i \in B, k \in O, \tag{3.9}$$

$$\sum_{l\in V}\sum_{m\in P_k} (c_{il}x_{ilm} + y_{ilm}) + y_{ijk} - (A_j - S_i)x_{ijk} \geq 0 \quad \forall i \in B, j \in W_i, k \in O, \tag{3.10}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i \in B, j \in V, k \in O, \tag{3.11}$$

$$y_{ijk} \geq 0 \qquad \forall i \in B, j \in V, k \in O. \tag{3.12}$$

where $P_k = \{m \in O : m > k\}$ (P_k is the set that includes the indexes m which are bigger than index k) and $W_i = \{j \in V : A_j > S_i\}$ (W_i is the set that includes the ships which have arriving time greater than the starting time of berth i).

An example for the problem instance can be seen in Table 3.1.

Total completion time of all the ships that are scheduled at berth i can be found from the equation below (3.13):

$$\sum_{j\in V}\sum_{k\in O} [(kc_{ij} + S_i - A_j)x_{ijk} + ky_{ijk}]. \tag{3.13}$$

An arriving ship may have to wait when $A_j < S_i$ (for j=1 in Figure 3.2) or the berth is occupied (for j=2 in Figure 3.2), which is denoted by W_j in Figure 3.2 or because a ship with a short handling time will be arriving soon. If the berth is unoccupied, an arriving ship may be handled immediately. An idle period (denoted by I in Figure 3.2 for ship 3 and may be equal to 0) for the berth occured which immediately precedes the handling of the ship. The total completion time of the ships handled at berth i is given by 3.13 and shown by the shaded area in Figure 3.2. This equation can be explained as follows:

Let

$$j_i^*(k) = j \in V | x_{ijk} = 1, \tag{3.14}$$

which denotes the index of the ships handled in the k^{th} last position at berth i for k=1, 2,..., T_i. Furthermore, let O_i = 1, 2,..., T_i, where T_i is the number of ships scheduled at

Table 3.1: An Example for 10 Berths - 20 Ships DBAP Instances

	Berths	1	2	3	4	5	6	7	8	9	10
	S_i	203	203	203	203	203	203	203	203	203	203
Ships	A_j	Handling Times (c_{ij})									
1	143	56	56	94	114	105	115	116	110	109	99
2	129	81	64	116	99	115	72	71	58	92	108
3	14	65	67	90	103	114	88	55	81	50	99
4	212	51	98	76	101	66	82	52	110	95	59
5	231	91	88	88	83	101	99	52	69	70	109
6	14	87	55	68	72	109	87	73	100	70	65
7	116	53	97	93	63	65	88	112	94	64	88
8	42	108	66	74	104	92	77	56	73	85	67
9	170	94	77	56	111	85	63	105	68	62	94
10	102	55	72	63	84	81	51	81	5	60	58
11	152	95	84	84	103	111	62	55	4	114	80
12	111	91	84	63	89	83	52	104	112	56	86
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...
19	110	86	70	65	80	85	76	112	86	112	89
20	70	60	115	68	101	115	89	78	63	81	114

S_i is starting time of berth i and, A_j is arrival time of ship j.

berth i, for all $i \in B$. Sum of the waiting times of the ships that arrive before S_i until berth i become available, where $W_i = \{j \in V : A_j > S_i\}$ is given in equation (3.15):

$$\sum_{k \in O_i} \sum_{j_i^*(k) \in V/W_i} (S_i - A_{j_i^*(k)}). \tag{3.15}$$

This sum is shown by the shaded rectangles to the left of the vertical line passing through $S_i = 0$ in Figure 3.2. The sum of the difference between departure time and S_i for all ships handled at berth i, which involves waiting, handling, idle and those periods before a ship arrives, is given by the equation (3.16):

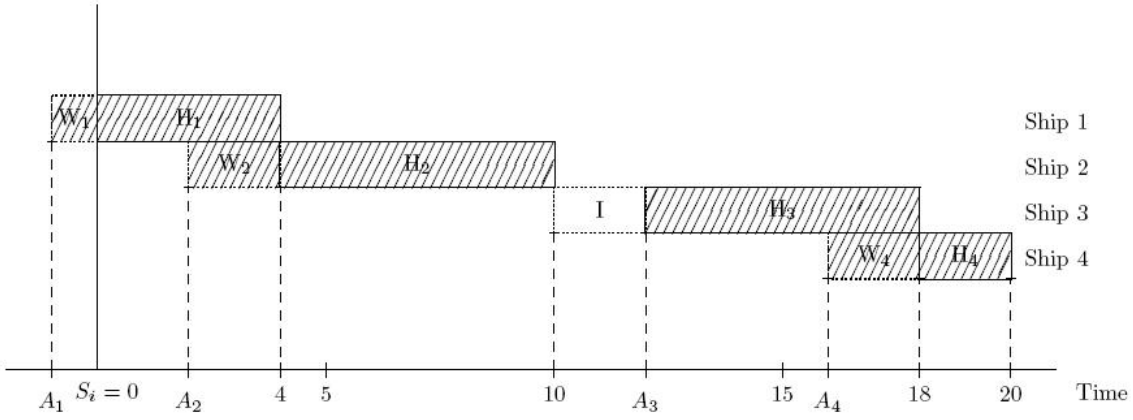

Figure 3.2: Example of handling ship at berth i for DBAP: A_1=-1, A_2=2, A_3=12, A_4=16, W_1=1, H_1=4, W_2=2, H_2=6, W_3=0, H_3=6, W_4=2, H_4=2, TCT=23.

$$\sum_{k \in O_i} \sum_{j_i^*(k) \in V} (d_{j_i^*(k)} - S_i), \tag{3.16}$$

where $d_{j_i^*(k)}$ stands for the departure time of ship j which is scheduled at position k of the berth i. This departure time is the end of the each H_j period that defines the handling time of ship j which are presented in Figure 3.2. Equation 3.16 corresponds to the sum of the areas between the end of H_j's (i.e., $d_{j_i^*(k)}$) and the right of $S_i = 0$ for each ship j. If these areas are partitioned into rectangles, we see that Equation (3.16) is equal to Equation (3.17) below:

$$\sum_{k \in O_i} k(y_{ij_i^*(k)k} + c_{ij_i^*(k)}). \tag{3.17}$$

An estimation of the total completion time is found by summing (3.15) and (3.17) when ships with $j \in \overline{W}_i$ arrive after S_i. A correction amount which is given by

$$\sum_{k \in O_i} (A_{j_i^*(k)} - S_i) \tag{3.18}$$

must be subtracted from this sum. This gives the area bounded by the vertical line through S_i, the abscissae axis and a staircase line immediately below and to the right of the shaded

rectangles corresponds to the handling a ship at berth i in Figure 3.2. Therefore, the sum will give

$$\sum_{k \in O_i} \sum_{j \in V} (S_i - A_{j_i^*(k)}) + k(y_{ij_i^*(k)k} + c_{ij_i^*(k)}), \tag{3.19}$$

which is the same as (3.13) when the decision variables are known (fixed).

Minimizing the total service time of the ship also refers to the minimization of the total delay in ships' departure time when the weight is not taken into account. Corresponding theorem and its proof is given below [22]:

Theorem 1. *The minimization of the total service time results in minimization of the total delay time for DBAP.*

Proof. Delay of ship j is given by d_j which is defined by equation below:

$$d_j = f_j - F_j \tag{3.20}$$

where F_j is the planned departure time and f_j is the completion time (actual departure time) of handling ship j.

Therefore, if we sum these delays over all ships, total delay (TD) can be interpreted as:

$$TD = \sum_{j \in V} d_j = \sum_{j \in V} f_j - F_j = \sum_{j \in V} f_j - \sum_{j \in V} F_j. \tag{3.21}$$

When the ship is handled at the best berth (i.e., the berth that has the minimum handling time among others) as soon as it arrives at the container terminal, F_j can be found by the equation below (3.22):

$$F_j = A_j + \min_i c_{ij}. \tag{3.22}$$

Since the total service time (TST) is determined as the sum of the service times of all ships, it is found by summing the difference of the completion time and the arrival time of the ship which can be seen in the folllowing equation (3.23):

$$TST = \sum_{j \in V} (f_j - A_j) = \sum_{j \in V} f_j - \sum_{j \in V} A_j. \tag{3.23}$$

If we consider equations (3.22) and (3.23) together, we have

$$TST = \sum_{j \in V} f_j - \sum_{j \in V} A_j = TD + \sum_{j \in V} F_j - \sum_{j \in V} A_j \tag{3.24}$$

As $\sum_{j \in V} F_j - \sum_{j \in V} A_j = \sum_{j \in V} \min_i c_{ij}$ is constant for ships in set V, TST is proportional to TD. ■

If different weights are assigned to the ships, minimization of the TST does not lead to minimization of TD time. This claim can be confirmed by a simple example given in Figure 3.3. Service times (ST) indicates the sum of waiting time and handling time, D-1 is the name of the case where all ships have the same weight that equals to 1, D-2 is the name of the case where ship A and ship B have different delay of departure weights. Weights of departure delay for ship A is 3, and 1 for ship B for the case D-2 for two different schedules on one berth as given in Table 3.2. Ship A and B have arriving times as 1 p.m. and 2 p.m, and handling times as 10 hours and 4 hours, respectively. There are two possible schedules for one berth and two ships case. In schedule 1, ship A is scheduled first and ship B is next and in schedule 2, ship B is scheduled first and ship A is next. These schedules have total service times 23 and 19 hours, respectively. If all the weights are 1, the weighted delay for both schedules are 9 and 5 hours, respectively (see Delay of departure D-1 column in Table 3.2). However, if these ships have different weights (i.e., 3 for ship A and 1 for ship B), the weighted delays turn out to be 9 and 15 hours, respectively (see Delay of departure D-2 column in Table 3.2). Therefore, this example proves that if the ships have different weights for departure delays, there will be no proportional relation between TST and TD any more.

A new constraint set (3.10) is added to SBAP to identify idle periods at berth i in DBAP. When k^{th} ship to be handled arrives, the time during which berth i has been available is

$$A_{j_i^*(k)} - S_i \tag{3.25}$$

and it has been used or idle until the departure of the ship handled at the $(k+1)^{st}$ last position, for a time of 0 if $k = T_i$ and

$$\sum_{m \in O_i, m > k} (y_{ij_i^*(m)m} + c_{ij_i^*(m)}) \tag{3.26}$$

otherwise.

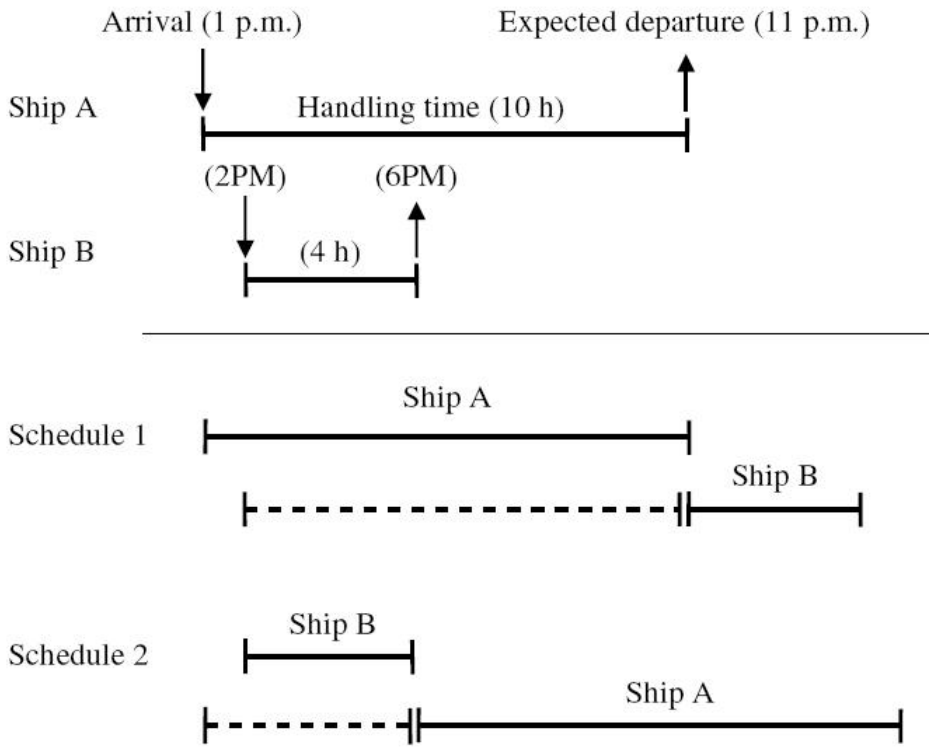

Figure 3.3: Service time and delay of departure for ships A and B

After the decision variables have been fixed, subtracting 3.26 from 3.25 corresponds to constraint set 3.10 on the length y_{ijk} of the idle period before handling the k^{th} last ship.

The size of the DBAP rapidly increases with the number of berths and ships considered. Size of a problem can be identified with the number of the existing variables and constraints. There are $IJT = |B|\,|V|\,|O|$ binary variables x_{ijk} and as many continuous variables y_{ijk}, J of constraint set (3.8), IT of constraint set (3.9) and IJT of constraint set (3.10). $IJT = |B|\,|V|\,|O|$ gives an upper bound for the number of constraints in Constraint set (3.10) because some $j \in V/W_i$ for some i and then no idle time y_{ijk} occurs. For example, for a realistic case used in [19], there are 25,000 0-1 binary variables, 25,000 continuous variables, 50 constraints of type (3.8), 500 constraints of type (3.9) and 25,000 constraints of type (3.10). Such numbers are very large for a MIP, even if constraints (3.8) and (3.9) are multiple-choice ones.

Table 3.2: Example to show TD and TST are not proportional if ships have different weights for departure delays. Service Time (ST), delay of departure (D-1) (all weights=1), delay of departure (D-2) (ship A's weight=3, ship B's weight=1)

		ST	Delay of departure (D-1)	Delay of departure (D-2)
Schedule 1	Ship A	10	0	0
	Ship B	13	9	9
	Total	**23**	**9**	**9**
Schedule 2	Ship A	4	0	0
	Ship B	15	5	15
	Total	**23**	**5**	**15**

In [1], DBAP is modeled as a variant of Multiple Depot Vehicle Routing Problem with Time Windows (MDVRPTW). In this model the ships are formulated as the customers, and the berths as the depots where one vehicle is located. They authors construct a tabu search heuristic on MDVRPTW formulation of DBAP. This heuristic always yields optimal solutions for small instances and it outperforms CPLEX solution for larger instances however the problem with 10 berths and 35 ships is the largest instance they have tried. It can be observed that the efficiency of tabu search decreases as problem instance size increases. Although tabu search algorithm is a good solution tool compared to CPLEX, it is not enough for larger problem instances in terms of solution quality. In [21], a genetic algorithm (GA) is developed for DBAP with simultaneous service in the public berth system for a container terminal. They first solved DBAP without simultaneous service to compare with the Lagrangian relaxation-based heuristic (LR) given in [19]. Although run time is good for small size problem instances, there is still a 10% optimality gap and this gap increases to 20% for larger instances and becomes closer to the LR. As a result of our calculations and the results of the studies given above, size of the DBAP gets more problematic when the problem instance is larger than 10 berths-35 ships. In fact, exact solution for some of the large instances of DBAP become unsolvable due to the memory errors related with the huge size of the branch-and-bound tree. In this thesis, CG algorithm is proposed for DBAP in order to overcome this memory problem without sacrificing from the solution quality by decomposing it into master and subproblems. Application of column CG to DBAP is

explained in Chapter 4.

Chapter 4

COLUMN GENERATION APPROACH FOR DYNAMIC BERTH ALLOCATION PROBLEM

CG is a good approach for those problems where the size of the MIP model grows immensely as a result of dealing with all the variables explicitly. CG generates only those variables which has the potential to improve the objective function. These variables, which have negative reduced costs, are determined by the subproblem (pricing problem). If the generated variable has nonnegative reduced cost, it is not considered as a new variable for the rest of the algorithm.

Structure of BAPs are similar to parallel machine scheduling problems in such a manner that jobs are scheduled to machines in parallel machine scheduling problems as ships are scheduled to berths in BAP. In fact, DBAP is the same with $R|r_j|\sum C_j$ which is unrelated parallel machine scheduling to minimize the total completion times of the jobs with arbitrary release dates (i.e., constraints and variables have almost the same role in problem formulation). In $R|r_j|\sum C_j$, jobs have different processing times for each machine as ships have different handling times for each berth in DBAP. Moreover, $R|r_j|\sum C_j$ deals with the jobs that have release dates similar to the arrival times of the ships considered in DBAP. Finally, the objective function of $R|r_j|\sum C_j$ minimizes the total completion time of the jobs and the objective function of DBAP minimizes the total service times of the ships.

CG algorithm is applied to $P||\sum C_j$ problem which is identical parallel machine scheduling to minimize the total completion times of the jobs without considering release dates ([49] and [50]). This problem needs less computational effort than $R|r_j|\sum C_j$ because $P||\sum C_j$ deals with the jobs that do not have release dates. In addition to this, jobs have the same processing times among the machines because all the machines have the same properties that affect the processing time of the jobs (i.e., identical machines). In this thesis, we implemented CG algorithm to solve DBAP which is described in Section 2.1, and which is computationally more difficult than $R|r_j|\sum C_j$. CG algorithm is a decomposition algo-

rithm as mentioned in Section 2.2. In order to use this decomposition algorithm, we first formulate DBAP as an integer program, and then reformulate this integer program as a set partitioning problem. CG algorithm needs initial *partial schedules* formed by a subset of ships for each berth. These partial schedules must satisfy the constraints of the master problem so, the initial solution given to the CG algorithm must be a feasible solution. This initial feasible solution is found by a simple heuristic in a negligible time. Details of the initial feasible solution procedure will be given in Section 4.4. We split the DBAP into two problems: the master problem and the subproblem (pricing problem). The master problem is the original problem with only a subset of variables being considered and the subproblem is a new problem created to identify a new variable. The objective function of the subproblem is the reduced cost of the new variable (column) with respect to the current dual variables, and the new column should satisfy the constraints of DBAP. More detailed explanation of the master problem and the subproblem are provided in Section 4.1 and Section 4.2, respectively.

4.1 The Master Problem

All assumptions made in DBAP model in Section 3.1 remain the same and the new assumptions for the master problem are introduced as follows:
(g) All berths have a feasible column set D_i indexed by D_i =1, 2,..., d_i.
(h) Variable z_i^d denotes d^{th} column of berth i and if this variable is 1, cost of r_{id} is incurred for column d=1, 2,...,D_i of berth i=1,2,...,I.
(i) Columns have T number of components for the ships denoted by $z_i^d = \{z_{i1}^d, z_{i2}^d, ..., z_{iT}^d\}$. If it is decided to schedule ship t at berth i in column d, z_{it}^d component of column d is 1, otherwise it is 0.

Only variable introduced to the master problem decides whether the associated column (schedule) is selected for the berth that is considered, or not. Description of this decision variable can be seen below:

$$z_i^d = \begin{cases} 1 & \text{if column } d \in D_i \text{ is chosen for berth } i, \\ 0 & \text{otherwise.} \end{cases}$$

Formulation of the master problem is given as follows:

(Master Problem Model)

$$\text{Minimize} \sum_{i \in B} \sum_{d \in D_i} z_i^d r_{id} \tag{4.1}$$

s.t.

$$\sum_{d \in D_i} z_i^d = 1 \qquad \forall i \in B, \tag{4.2}$$

$$\sum_{i \in B} \sum_{d \in D_i} z_{it}^d z_i^d = 1 \qquad \forall t \in V, \tag{4.3}$$

$$z_i^d \in \{0, 1\} \qquad \forall i \in B, d \in D_i. \tag{4.4}$$

Constraint sets (4.2) and (4.3) correspond to the original constraint sets (3.8) and (3.9) respectively. These constraints dictates that each ship is covered by exactly one feasible schedule and each berth is occupied by exactly one feasible schedule, respectively. In order to get the corresponding dual variables of the constraints of the Restricted Master Problem (RMP), the linear relaxation of the RMP (i.e., by defining z_i^d as continuous variables) is solved in each iteration with the existing columns. After this solution, dual variables of the constraint sets (4.2) and (4.3) are passed to the pricing problem which are given as follows:
α_i = Dual variable that is associated with the constraint i of constraint set (4.2).
λ_j = Dual variable that is associated with the constraint j of constraint set (4.3).
Every berth and every ship has its associated dual variable (i.e., α_i's are used for the berths and λ_j's are used for the ships). These dual variables will help to find new partial schedules by solving the subproblems (pricing) for each berth one by one.

4.2 The Pricing Problem (Subproblem)

The goal of the subproblem is to find the column (schedule) with the minimum reduced cost for the associated berth to be added to the RMP. Since DBAP has berths that have different starting times (S_i) and different handling times for the ships (c_{ij}). A schedule which is

optimal for one berth may not be optimal for any other berth, hence the subproblem should be solved for each berth separately.

Only difference between the variables defined in the DBAP and in the subproblem is that one index of the variables of the subproblem are dropped off compared to the ones in the DBAP. Since we solve the subproblem for each berth, the index used for berths in variables x_{ijk} and y_{ijk} of DBAP are not necessary any more for the subproblem formulation. Number of the constraints defined for berths in DBAP is also decreased (3.9 and 3.10). These differences lessen the computational requirements of the DBAP formulation enormously so the resulting subproblem formulation is much easier in terms of finding the optimal solution.

New variable associated with the variable x_{ijk} of DBAP is introduced as follows:

$$a_{jk} = \begin{cases} 1 & \text{if ship } j \text{ is scheduled in position } k, \\ 0 & \text{otherwise.} \end{cases}$$

This variable is necessray to find whether ship j is scheduled at the associated berth in position k or not. It can be easily realized that this variable corresponds to an element of the columns generated for the RMP. Any element of variable z_i^d of master problem indicates whether the corresponding ship t is included in schedule d of berth i or not by z_{it}^d. Thus, the value of the variable a_{jk} is turned into z_{it}^d by changing index j into t and, ignoring the k index. Index d is set automatically by the program to denote a schedule by calculating the number of columns added for each berth individually. If the last column added for berth i has column number n for index d, the next column generated for this berth should have column number $n + 1$ for index d.

Next, the continuous variable w_{jk} is defined to find the idle period before the arrival of ship j that will be scheduled at position k.

w_{jk} = the length of the idle period before the arrival of ship j that will be scheduled at position k.

This variable is not valid if there is no such ship scheduled at position k. Then, this variable is either 0 or positive if there is an idle period before the arrival of the ship that will be scheduled at position k. Reduced cost of schedule d for berth i can be found by the equation given below:

$$r'_{id} = r_{id} - \sum_{j \in V} \sum_{k \in O} \lambda_j a_{jk} - \alpha_i, \tag{4.5}$$

where α_i=α_1,...α_B are the given value of the dual variable corresponding to constraint set (4.2) and λ_j=λ_1,..., λ_T are the given values of the dual variables corresponding to constraint set (4.3). To test whether the current solution is optimal, we determine whether there exists a berth schedule $d \in D_i$ with negative reduced cost for all of the berths independently or not. Thus, *pricing problem* is solved for finding the berth schedule in D_i with minimum reduced cost. Therefore, we essentially have to minimize

$$r_{id} - \sum_{j \in V} \sum_{k \in O} \lambda_j a_{jk} - \alpha_i = \sum_{j \in V} \sum_{k \in O} (kc_{ij} + S_i - A_j - \lambda_j) a_{jk} + \sum_{j \in V} \sum_{k \in O} k w_{jk} - \alpha_i \tag{4.6}$$

subject to the adopted versions of constraint sets 3.9 and 3.10. As a result, subproblem is given by equations 4.7 - 4.11 as follows:

(Subproblem Model for bert i)

$$\text{Minimize} \sum_{j \in V} \sum_{k \in O} (kc_{ij} + S_i - A_j - \lambda_j) a_{jk} + \sum_{j \in V} \sum_{k \in O} k w_{jk} - \alpha_i \tag{4.7}$$

s.t.

$$\sum_{k \in O} a_{jk} \leq 1 \qquad \forall j \in V, \tag{4.8}$$

$$\sum_{l \in V} \sum_{m \in P_k} (c_{il} a_{lm} + w_{lm}) + w_{jk} - (A_j - S_i) a_{jk} \geq 0 \quad \forall j \in W_i, k \in O, \tag{4.9}$$

$$a_{jk} \in \{0, 1\} \qquad \forall j \in V, k \in O, \tag{4.10}$$

$$w_{jk} \geq 0 \qquad \forall j \in V, k \in O. \tag{4.11}$$

The subproblem model needs to be solved I (number of berths) times, one time for each berth separately at each iteration. If this problem has berths that are identical, the

CG algorithm can be terminated when the reduced cost obtained from the subproblem is nonnegative (i.e., there is no candidate schedule to improve the objective function of the master problem). When this condition holds, the RMP is turned into the master problem that includes all necessary schedules as there exist no more candidate columns left to be added to column set D. This master problem is solved as a MIP to find an integer feasible solution. Since DBAP has nonidentical berths, subproblems associated with each berth give different optimal schedules with different reduced costs at each iteration. Therefore, iterations cannot be terminated by the common stopping criterion that looks whether there exists a schedule with negative reduced cost or not. Details of the CG algorithm will be explained in Section 4.4.

4.3 Relation of the Master Problem and the Pricing Problem

Relation of the master problem and the pricing problem is explained in Sections 4.1 and 4.2. In this section, economical interpretation of the dual variables that come from constraints of the master problem and their contribution to find a potentially improving column will be given to clarify the relationship between the master problem and the pricing problem. In order to understand the relation of the master and the subproblem, one should start from the calculation of the reduced cost of the new schedule that is found from the solution of the subproblem. This reduced cost calculation which is also given in Section 4.2 can be seen below.

$$r_{id}^{'} = r_{id} - \sum_{j \in V} \sum_{k \in O} \lambda_j a_{jk} - \alpha_i, \tag{4.12}$$

The equation given above gives the reduced cost of schedule d for berth i where r_{id} is the cost of schedule d for berth i. Aim of minimizing the reduced cost of the new schedule is to find a new schedule that has a negative reduced cost so as to improve the objective function of the master problem. Correspondingly, improving the objective function of the master problem means improving the objective functon of the original DBAP.

The question here is how the reduced cost of the new schedule is found. Solving the linear relaxation of the master problem gives dual variables for each berth and each ship which are λ_j and α_i correspondingly. Economical interpretatons of these variables can be explained as follows;

λ_j = the cost or benefit of adding one unit of ship j to the schedule considered.

α_i = the cost or benefit of adding one unit of berth i to the schedule considered.

These are the general interpretations of the dual variables. Interpretations of these variables should be given by considering the properties of the DBAP to make them more understandable. Therefore, interpretation of the λ_j is given below as;

λ_j = the cost or benefit of scheduling ship j at the corresponding berth.

For example, if the pricing problem of berth i is considered and ship j is scheduled at that berth i, one of the binary variables a_{jk} become 1 for one order k. This means corresponding λ_j value of the ship j is subtracted from the cost of the schedule by the $\sum_{j \in V} \sum_{k \in O} \lambda_j a_{jk}$ part of the reduced cost equation. Cost of scheduling ship j at berth i is calculated in r_{id} part of the reduced cost by considering order of the ship in new schedule and subtracting $\sum_{j \in V} \sum_{k \in O} \lambda_j a_{jk}$ from r_{id} gives the reduced cost of scheduling ship j at berth i.

In addition to λ_j, interpretation of α_i is given below as;

α_i = the cost or benefit of using berth i for the new schedule.

Since new schedules are found berth by berth, α_i is just subtracted from the objective function of the subproblem of berth i. This means that it just affects the objective function value of berth i. Since α_i is not multiplied with any other decision variable of the pricing problem, we are sure to subtract this value from the cost of the new schedule to find the reduced cost of the new schedule. Since α_i is subtracted from all of the possible schedules, value of this dual variable does not affect the optimal solution of the pricing problem by means of finding the new schedule. Therefore, α_i is just used to find the real reduced cost of the new schedule.

As a result, when a new schedule with negative reduced cost is found, it is added to the master problem as a new column and master problem is solved after solving the pricing problem for all berths. After the solution of the master problem, new dual variables are taken from the constraints and they are passed to the new pricing problems of the berths. This process goes on like this until no more schedules with negative reduced cost exists.

4.4 The Algorithm

As explained in Chapter 4, the CG algorithm needs an initial feasible solution to obtain dual variables of the linear relaxation of RMP. A simple heuristic algorithm is used to find

this initial feasible solution in a negligible time. This algorithm sorts the ships and berths in increasing order of their arriving times and starting times. After that, it schedules first ship to the first berth, second ship to the second berth and when it comes to the end of the berths, it starts from the first berth to schedule the remaining ships. Cost of this schedule is calculated and kept as best schedule on hand. Then, the algorithm chooses 2 random ships among 2 random berths and swaps them for a predetermined iteration number. If the schedule obtained from this swap has smaller objective function value than that of the best schedule, the algorithm changes best schedule to the schedule obtained from the swap operation. Outline of this simple algorithm is given by Algorithm 1.

After the execution of this heuristic algorithm, we get a feasible solution for the DBAP. This feasible solution should be turned into the column structure to be added to the RMP. The solution taken from Algorithm 1 gives the schedule of all berths and the costs of these schedules. An example for solution of a 10 berths - 20 ships instance can be seen in Table 4.1. The numbers in Table 4.1 indicate the order of the corresponding ship in schedule of the corresponding berth. For example, entry 3 given at the first column for ship 4 means that ship 4 is scheduled at berth 1 in order (position) 3 (i.e., ship 4 is the 3^{rd} ship to be handled at berth 1).

Actually, the column structure is an array of size equal to the number of ships in the problem instance so column size is 20 for this example problem instance. Since berths are not identical, there exist different optimal schedules (i.e., equal to the number of berths in the problem instance) for different berths in a feasible solution. These columns consist of just 0's and 1's such that, 0 indicates that the corresponding ship is not scheduled at the associated berth and 1 indicates that the corresponding ship is scheduled at the associated berth which the schedule is belonging to. Each optimal schedule is a new column for the RMP. The column representation of the initial feasible solution given in Table 4.1 can be seen in Table 4.2.

Information given in Table 4.2 is the only information provided to the RMP for the initiation of the CG algorithm. Order of the ships scheduled at one berth cannot be realized from the columns passed to the RMP but costs of the columns are calculated according to the orders of the ships. Influence of the order of the ships is reflected to the objective function of the RMP by adding the column with the cost of that schedule. Moreover,

Algorithm 1 Heuristic Approach for Initial Solution of DBAP

Sort ships in increasing order of their arrival times and store it in $OrderShip$

Sort berths in increasing order of their starting times and store it in $OrderBerth$

$kmax \leftarrow 15$

$kmin \leftarrow 1$

$maxIter \leftarrow 10$

$iter \leftarrow 0$

$i \leftarrow 0$

$k \leftarrow 1$

for $j = 1$ to $NoOfShips$ **do**

  if $i \geq NoOfBerths$ **then**

    $k \leftarrow k + 1$

    $i \leftarrow 1$

  else

    $i \leftarrow i + 1$

  end if

  $ii \leftarrow OrderBerth[i]$

  $x[ii][k] \leftarrow OrderShip[j]$

end for

$fbest \leftarrow CostOfTheInitialSchedule$

while $iter < maxIter$ **do**

  $iter \leftarrow iter + 1$

  $kcur \leftarrow kmin$

  while $kcur \leq kmax$ **do**

    Choose 2 random berths and 2 random ships, then swap the positions of the ships

    $fs \leftarrow CostOfTheNewSchedule$

    if $fs < fbest$ **then**

      $fbest \leftarrow fs$

      $kcur \leftarrow kmin$

    else

      $kcur \leftarrow kcur + 1$

    end if

  end while

end while

order information of the ships can be obtained from the variables a_{jk} of the corresponding subproblem of the column which are introduced in Section 4.2. Therefore, there is no need to complicate the RMP by introducing a new variable or a new index that contains the order of the ships.

An example for the columns generated can be provided by using the information given in Table 4.2. First columns of the berths added to the master problem can be represented as below (only the z_{ij}^{d} elements of the columns that have value 1 is represented, other elements of the columns have value 0):

$z_1^1 = 1(z_{11}^1), ..., 1(z_{14}^1), ..., 1(z_{1,20}^1)$; with the objective function coefficient $r_{11} = 523$
$z_2^1 = ..., 1(z_{22}^1), ..., 1(z_{26}^1), ...$; with the objective function coefficient $r_{21} = 437$
$z_3^1 = ..., 1(z_{38}^1), ...$; with the objective function coefficient $r_{31} = 235$
$z_4^1 = ..., 1(z_{47}^1), ..., 1(z_{4,19}^1), ...$; with the objective function coefficient $r_{41} = 386$
$z_5^1 = ..., 1(z_{5,15}^1), ..., 1(z_{5,18}^1), ...$; with the objective function coefficient $r_{51} = 448$
$z_6^1 = ..., 1(z_{6,13}^1), ..., 1(z_{6,17}^1), ...$; with the objective function coefficient $r_{61} = 341$
$z_7^1 = ..., 1(z_{73}^1), ..., 1(z_{7,11}^1), ...$; with the objective function coefficient $r_{71} = 405$
$z_8^1 = ..., 1(z_{85}^1), ..., 1(z_{8,14}^1), ...$; with the objective function coefficient $r_{81} = 253$
$z_9^1 = ..., 1(z_{99}^1), ..., 1(z_{9,12}^1), ...$; with the objective function coefficient $r_{91} = 299$
$z_{10}^1 = ..., 1(z_{10,10}^1), ..., 1(z_{10,16}^1), ...$; with the objective function coefficient $r_{10,1} = 425$

These columns are added to the master problem and its linear relaxation is solved for only those columns at the first iteration. New dual variables are obtained from the solution of the RMP at every iteration that is solved with the updated column set D. That is why the master problem is called the restricted master problem. In most of the cases, CG algorithms terminate if there exists no column with negative reduced cost which means that the objective function of the subproblem is nonnegative as indicated at the beginning of the Chapter 4. We have different subproblems at the end of each iteration because of the parameters that depend on the dual variables of the previous RMP (i.e., λ_j and α_i).

The CG algorithm is presented in Algorithm 2 and CG can be explained as follows. We define two decision criteria before the execution of the algorithm which are given as *NearZero* and *NumOfIterWithoutImp*. *NearZero* is a small negative number which is determined to decide whether the column obtained from the subproblem should be added to the RMP or not. If the column has a reduced cost smaller than *NearZero*, it is decided to be added to

the RMP. *NumOfIterWithoutImp* is determined to decide whether to stop the algorithm or not. If the objective function value of the RMP does not change for *NumOfIterWithoutImp* sequential iterations, algorithm is terminated. This control criterion is used to avoid the *tailing-off effect*. In this algorithm, the relaxation of the RMP is solved for the columns obtained from the initial feasible solution. Dual variables for each berth and each ship are passed from the RMP to the subproblem. Subproblem is solved exactly for each berth and a column (i.e., schedule) is obtained for the corresponding berth. If the objective function of the subproblem (i.e., reduced cost of the new column) is smaller than *NearZero*, it is added to the RMP, otherwise it is not. After the execution of the subproblems, the relaxation of the RMP is solved for the columns on hand. This process goes on like this and *NumOfIterWithoutImp* is updated at each iteration if the objective function value of the RMP does not improve. If an improvement is observed in this value, *NumOfIterWithoutImp* is set to zero. When *NumOfIterWithoutImp* is reached to a predetermined number (i.e., maximum number of iterations without improvement (*MaxNumOfIterWithoutImp*)), the algorithm is terminated and RMP is solved as a MIP and an integer solution is found as a final solution.

Termination criterion of CG can be modified as *if there exists no columns with negative reduced cost for all subproblems (berths), terminate the CG*. We tried this stopping criterion for our CG algorithm but we have faced the most common drawback of the CG algorithm which is the *tailing-off effect*. Simplex-based CG is known for its poor convergence as stated in [42]. While usually a near optimal solution is attained considerably fast, only a little progress per iteration is made when algorithm gets close to the optimal solution. There can be another case where the CG algorithm does not improve the solution for prominent amount of iterations since the optimal solution is already found in previous iterations. The CG algorithm does not terminate since it continues to find columns with negative reduced cost for some or all berths. The solution obtained may also be a degenerate solution and it is time consuming to prove optimality of a degenerate optimal solution. When the behaviour of the CG algorithm displayed figuratively, a long tail is observed for the values of the best solution of each iteration at hand by [51]. Moreover, performing the CG algorithm until no more column with negative reduced cost can be found by all of the subproblems may result in tailing-off effect due to the slow convergence of the CG algorithm. Subproblems need long

iterations to reach nonnegative objective function values (i.e., reduced costs for generated columns) however, the CG algorithm may achieve the optimal solution long before the subproblems reach nonnegative objective function values. This situation is shown in Figure 4.1 for a 5 berths - 40 ships instance which has an optimal objective function value as 24,875.

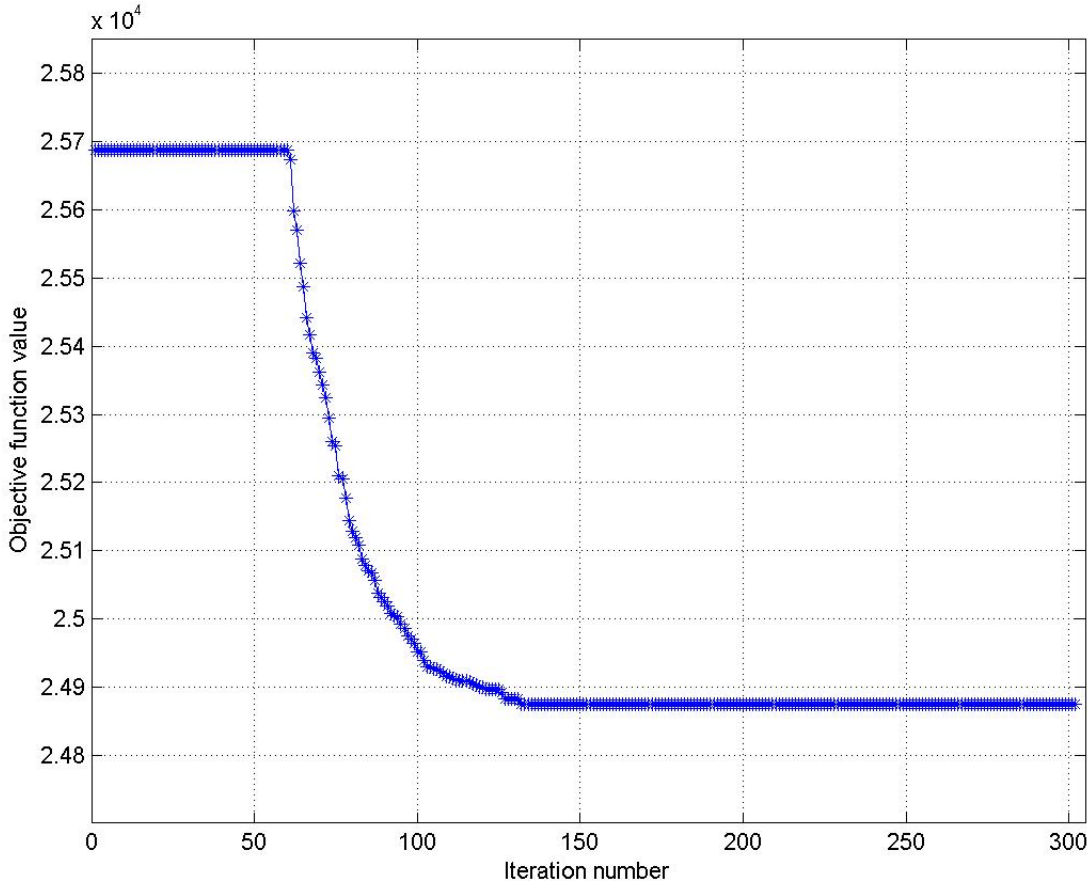

Figure 4.1: Slow convergence of CG algorithm (tailing-off effect) for a 5 berths - 40 ships instance that has an optimal objective function value as 24875.

The so-called *long tail* is observed in Figure 4.1 as a result of repeating objective function value at hand between iterations 130 and 310, which is the optimal objective function value. Although the optimal objective function value is obtained at iteration 130, CG algorithm did not terminate since it did not ensure the stopping criterion which is described as finding no more columns with negative objective function values. Following similar observations, we

come up with a decision about changing the stopping criterion to improve the computation times of the CG algorithm without sacrificing the solution quality. Thus, we used *number of iterations without improvement in objective function value of the master problem* as a stopping criterion. Different values for this stopping criterion are used for different problem instances according to the difficulty in finding the optimal solution of the problem instance.

This algorithm needs to solve subproblem for each berth (*number of berths* times) exactly at each iteration. If algorithm needs $NumIter$ iterations to terminate, subproblem must be solved $NumIter \times BerthNo$ exactly until the CG algorithm terminates. When the problem instances get larger and harder, it gets difficult to find the optimal solution for the subproblems. This means that the CG algorithm requires more computational effort for the larger and harder problem instances. Computational results for the CG algorithm are given in Section 4.5.3. When we analyzed the results of the CG algorithm, we still observe the long computation times, especially for the harder instances.

The data used have different difficulty levels for same group instances (i.e., they have same number of berths and ships as described in Section 4.5.1). This difficulty levels are labeled by taking 1/2, 3/5, 5/8 and 7/8 of the time between the arrivals of the first and the last ships for the availability times of the berths (starting times S_i). Instances that have the ratio 1/2 for the starting times of the berths are the most difficult instances and we have 3 different instances for the same difficulty level. Thus, there are 12 different instances for same number of berths and ships.

The effect of this different difficulty levels can be seen in subproblems because the starting times of the berths and the arrival times of the ships are employed only in the subproblems. Moreover, since the linear relaxation of the RMP is solved at each iteration, most of the elapsed time is spent for solving the subproblems. Therefore, we decided to apply a heuristic algorithm to the subproblems for decreasing the computation time of the CG algorithm. This heuristic algorithm is a local search heuristic and we used the solution that come from the assignment problem as the initial solution for the local search heuristic. Subproblem of the CG algorithm is developed as an assignment problem by ignoring the constraint set (4.9) and changing the objective function as minimizing;

$$\sum_{j \in V} \sum_{k \in O} (kc_{ij} + S_i - A_j) a_{jk} - \sum_{j \in V} \sum_{k \in O} \lambda_j a_{jk} - \alpha_i. \tag{4.13}$$

Eventually, the assignment problem with objective function 4.13 and constraint sets (4.8), (4.10) and (4.11) is solved and a local search algorithm is performed for its solution. The CG approach with the heuristic applied for the subproblem is given in Algorithm 3. Moreover, a flow chart of the Algorithm 3 can be seen in Figure 4.2. Since the CG algorithm cannot guarantee the optimality with a heuristic approach, subproblems are solved exactly when heuristic cannot find a schedule with negative reduced cost. Hence, the CG algorithm should be terminated when none of the subproblems find a column with negative reduced cost at any iteration.

The reason why we use both heuristic and exact solution for subproblem is to prevent stucking in a local optimal solution. This algorithm ensures that if the columns generated by heuristic algorithm cannot improve objective function value of the RMP *(Maximum Number of Iterations Without Improvement - Max Number of Exact Subproblem Iterations)* iterations consecutively, the algorithm changes subproblem solution procedure to exact solution for at most *Max Number of Exact Subproblem Iterations* exact iterations as can be seen in Algorithm 3. If the columns generated by the exact procedure can improve the objective function value of the RMP, subproblems continue to be solved by the heuristic procedure until *Maximum Number of Iterations Without Improvement* consecutive iterations are recorded with the same objective function value. If an improvement cannot be achieved through *Max Number of Exact Subproblem Iterations* iterations, the algorithm terminates with the current best integer solution by solving the RMP as a MIP. Parameters such as *Maximum Number of Iterations Without Improvement* and *Max Number of Exact Subproblem Iterations* can be tuned according to the behaviour of the instance that is considered to be solved. For example if the problem has a tendency to stuck in local optimal solution for long iterations, *Max Number of Exact Subproblem Iterations* should be increased to find a solution with good quality.

We achieved better computation times for DBAP instances by Algorithm 3 without sacrificing the solution quality. Results and comparisons of this algorithm with the Algorithm 2 and other algorithms provided for DBAP in the literature are given in Section 4.5.3.

4.5 Implementation Details

4.5.1 Test Problems

We tested the proposed CG approach for the DBAP by using the data from the literature. We report the computational results for medium and large size instances of DBAP which are given in [14]. These test data are generated according to the procedure given in [19]. [14] also presented a Variable Neighborhood Search (VNS) algorithm and compared the performance of VNS with a Genetic Algorithm proposed in [21], Multi-Start Variable Neighborhood Descent (MVDN) and Memetic Search Algorithm (MA) which is the extension of the GA.

Problem instances used in this study can be summarized as follows:

- Number of berths, $|B|$= 5, 10.
- Number of ships, $|V|$ = 35, 40, 45, 50.
- Arrival times of the ships, A_j, are from a uniform distribution in the range of $[1, (7000/60)(|V| \,/\, |B|)]$.
- Handling times, c_{ij} =(2*u_{ij} + 1.5)* 2000/60, where u_{ij} is a random number from the uniform distribution between 0 and 1.
- Availability times of the berths, S_i = 1/2, 3/5, 5/8, 7/8, of the arrival time interval between the arrivals of the first and the last ships.

Three problem instances for each combination of the data above are generated. Therefore, there are twelve problem instances for each berth-ship combination. $S_i = 1/2, 3/5, 5/8, 7/8$ gives the difficulty level of the problem sets where 1/2 indicates the most difficult one and 7/8 indicates the easiest level. The reason why the small number of ships, such as 10, 15, 20, and 25 are not used in this study is that the VNS, the CG and the exact algorithms all result in short computational times. In addition to this, they all give the optimal solution for these instances so that a comparison is not meaningful for these instances.

4.5.2 Computational Platform

Computational experiments are performed on an Intel(R) computer with Xeon(R) CPU 3.00 GHz, 4.00 GB of RAM. We report the total elapsed times, not the CPU time for

each instance. We have used ILOG CPLEX version 11.0 [52]. Mixed integer and primal simplex optimizers of CPLEX are used to solve MIP and LP relaxations respectively. The CG heuristic algorithm and the exact method are implemented with C++ API of ILOG CPLEX Concert Technology with Microsoft Visual Studio 2005.

4.5.3 Computational Results

Performance of the CG heuristic algorithm and the heuristic CG heuristic algorithm have been tested on 84 problem instances which can be assumed as medium and large size problem instances of DBAP. These problem instances can be named by the number of berths and the number of ships in the problem. Since there are four difficulty levels for each berth-ship combination and three problem instances for each difficulty level, difficulty level and order of the problem instance in that difficulty level should also be included in the name of the problem instances. As an example, DBAP-10B-50S-1/2-1 refers to the DBAP problem instances comprised of 10 berths, 50 ships, first difficulty level (i.e., 1/2). Since a difficulty level includes three different problem instances of the same difficulty (i.e., availability (starting) times of the berths are computed with the same ratio), the number 1 at the end of the problem instance name corresponds to the first problem instance of the difficulty level 1/2.

In our computational experiment, we compare a heuristic method (VNS) from the literature, a hybrid approach (the proposed CG heuristic algorithm) and an exact method (CPLEX solution). Even though we have the same data with [14] where the results of VNS and the exact algorithm were given, it is not fair to compare the run times due to different computational platforms. For this reason, we implemented an exact method on the same computational platform with our CG heuristic algorithm. VNS computation times are the results that are taken from [14]. Since they provided the average of the computation times, we use the same run times for the instances that have the same number of berths, number of ships and difficulty level.

The smallest size test instances have 5 berths-35 ships and 10 berths-40 ships because for smaller size test instances all three methods give the optimal solution with almost negligible run times.

It can be seen from Table 4.3 that the CG heuristic algorithm has the same solution quality with the exact method. Thus, the CG heuristic algorithm reaches optimality but

run times of the exact solution is lower than the CG heuristic algorithm for all instances given in Table 4.3. If we compare the solution quality of the CG heuristic algorithm with VNS, we see that the CG heuristic algorithm has better objective function values than those provided by VNS in 5 of the 24 instances. Since CG is a hybrid algorithm, it needs exact solutions for the subproblems. As a consequence, the CG heuristic algorithm needs longer computation times than VNS heuristic.

Table 4.4 can be interpreted in a similar way as Table 4.3. In this case, the CG heuristic algorithm has reached optimality in all of the instances but there is one instance that we cannot find the optimal solution with the exact method because of the *out of memory error* (DBAP-5B-50S-1/2-2). Run times of VNS for the instances in Table 4.4 are smaller than the run times of the CG heuristic algorithm similar to the instances presented in Table 4.3 but the CG heuristic algorithm outperforms VNS in 2 of the 24 instances in terms of the objective function value.

When the berth number is increased to 10, the problem becomes more difficult which can be observed from the increase in the run times given in Table 4.5. The CG heuristic algorithm has better objective function values than the VNS heuristic in 17 of the 36 problem instances given in Table 4.5. However the computation times of the CG heuristic algorithm are extremely longer than the VNS algorithm. For the exact solution, comparison of the CG heuristic algorithm becomes more difficult than the 5 berths problem instances case because of the *out of memory error* taken from 14 problem instances out of 36. The CG heuristic algorithm reaches optimality for the instances that are solvable by the exact method (i.e., for the problem instances that we know the optimal objective function value) and the exact method has smaller run times than the CG heuristic algorithm for the solvable instances.

Overall, we can observe from the results that the CG heuristic algorithm is successful in finding the optimal solution where the exact algorithm fails to do so. Even though the computation time of the CG heuristic algorithm is larger compared to the computation time of the exact algorithm, the CG heuristic algorithm can be used as a practical algorithm since these decisions are taken on a weekly basis in real life.

On the other hand, since CG heuristic algorithm provides better quality solutions than VNS algorithm, it proves to be a usable approach to obtain optimal solutions.

These outcomes are promising for the future of the CG heuristic algorithm application

for DBAP. The CG heuristic algorithm may become more compatible with the heuristic algorithms provided in the literature by some improvements such as, a heuristic algorithm that gives better solutions for the subproblem, addition of dual-optimal inequalities that have potential to accelerate the CG heuristic algorithm, or other stabilization methods.

Table 4.1: An example of the initial feasible solution for 10 berths - 20 ships instance

	Berths									
Ships	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0
4	3	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	2	0	0
6	0	1	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	2	0
10	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	2	0	0	0
12	0	0	0	0	0	0	0	0	1	0
13	0	0	0	0	0	2	0	0	0	0
14	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	2	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	2
17	0	0	0	0	0	1	0	0	0	0
18	0	0	0	0	1	0	0	0	0	0
19	0	0	0	2	0	0	0	0	0	0
20	2	0	0	0	0	0	0	0	0	0
Cost	523	437	235	386	448	341	405	253	299	425

Table 4.2: Column representation of the initial feasible solution given in Table 4.1

	Berths									
Ships	1	2	3	4	5	6	7	8	9	10
1	1	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0
4	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0
6	0	1	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0
10	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	0	0	1	0	0	0
12	0	0	0	0	0	0	0	0	1	0
13	0	0	0	0	0	1	0	0	0	0
14	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	1	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	1	0	0	0	0
18	0	0	0	0	1	0	0	0	0	0
19	0	0	0	1	0	0	0	0	0	0
20	1	0	0	0	0	0	0	0	0	0
Cost	523	437	235	386	448	341	405	253	299	425

Algorithm 2 Algorithm of CG Approach

$NearZero \leftarrow -10^{-3}$ (A parameter defined for deciding to add a column).

$NumOfIterWithoutImp \leftarrow 0$ (For counting the iterations that have the same objective function value consecutively).

Solve the LP relaxation of the RMP with the columns derived from the initial solution.

Pass the dual variables (λ and α) to the subproblem.

for $i = 1$ to *NoOfBerths* **do**

  Solve the subproblem for berth i with the set of current dual variables.

  Schedule is the schedule of ships obtained from subproblem i.

  if $ObjFunSub < NearZero$ **then**

    Add *Schedule* to the RMP as a new column for berth i.

  end if

end for

while $NumOfIterWithoutImp < MaxNumOfIterWithoutImp$ **do**

  Solve the LP relaxation of RMP with the curent columns.

  if New $ObjFunMas$ = Old $ObjFunMas$ **then**

    $NumOfIterWithoutImp \leftarrow NumOfIterWithoutImp + 1$

  else

    $NumOfIterWithoutImp \leftarrow 0$

  end if

  for $i = 1$ to *NoOfBerths* **do**

    Solve the subproblem for berth i with the set of current dual variables.

    Schedule is the schedule of ships obtained from subproblem i.

    if $ObjFunSub < NearZero$ **then**

      Add *Schedule* to the RMP as a new column for berth i.

    end if

  end for

end while

solve the RMP as an integer problem to get an $IntegerSolution$

$FinalSchedule \leftarrow IntegerSolution$

Algorithm 3 Algorithm of CG Approach with the Heuristic Applied to the Subproblem

First 11 lines of Algorithm 2.
while $NumOfIterWithoutImp < MaxNumOfIterWithoutImp$ **do**
  Solve the LP relaxation of *RMP* with the current columns.
  if New $ObjFunMas$ = Old $ObjFunMas$ **then**
    $NumOfIterWithoutImp \leftarrow NumOfIterWithoutImp + 1$
  else
    $NumOfIterWithoutImp \leftarrow 0$
  end if
  for $i = 1$ to *NoOfBerths* **do**
    if $NumOfIterWithoutImp > MaxNumOfIterWithoutImp - NumOfExactSub$ **then**
      Solve the subproblem for berth i exactly with the set of current dual variables.
      Schedule is the schedule of ships that come from subproblem i.
      if $ObjFunSub < NearZero$ **then**
        Add *Schedule* to the *RMP* as a new column for berth i.
      end if
    else
      Solve the assignment problem version of the subproblem for berth i.
      Schedule is the schedule of ships obtained from subproblem i.
      Set $NumShips$ to the number of ships scheduled in *Schedule*.
      $BestSchedule \leftarrow NewSchedule$
      for $k = 0$ to *NumShips* **do**
        remove k random ships from *Schedule*
        for $t = 0$ to *MaxNoTrial* **do**
          define 2 random ships from *Schedule* and change their position to get a $NewSchedule$
          if *ReducedCost* of $NewSchedule <$ *ReducedCost* of $BestSchedule$ **then**
            $BestSchedule \leftarrow NewSchedule$
          end if
        end for
      end for
      add $BestSchedule$ to the RMP as a new column for berth i.
    end if
  end for
end while
solve *RMP* as an integer problem to get an $IntegerSolution$
$FinalSchedule \leftarrow IntegerSolution$

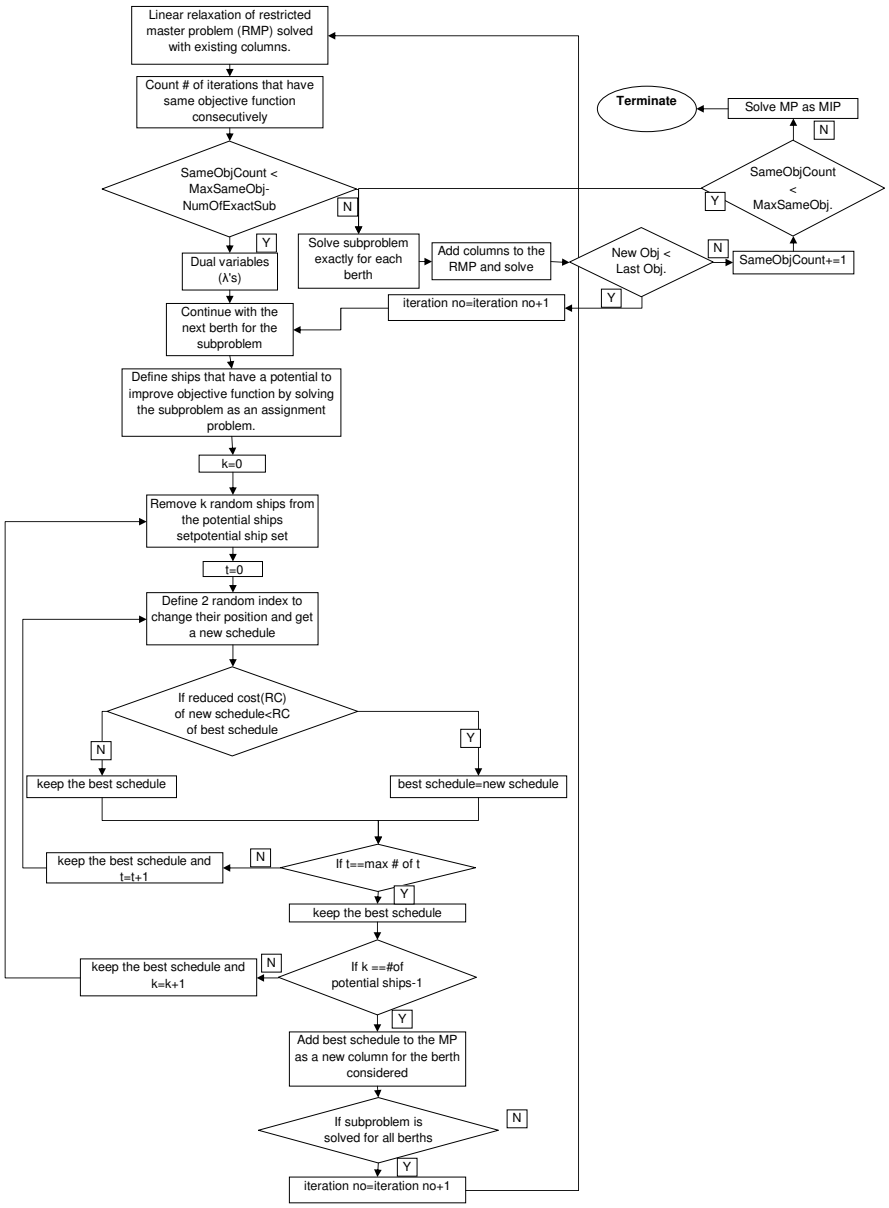

Figure 4.2: Flow Chart of the Heuristic CG Algorithm

Table 4.3: Results of DBAP instances for 5 berths-35 ships and 5 berths-40 ships with heuristic CG

	Objective Function Values			Computation time (Seconds)			
Problem Instance	VNS	CG	Exact	VNS	CG	CG Heuristic	Exact
DBAP-5B-35S-1/2-1	8289	**8282**	**8282**	1.08	1803.92	1269.00	162.27
DBAP-5B-35S-1/2-2	8348	8348	8348	1.08	1710.00	1754.36	288.28
DBAP-5B-35S-1/2-3	8102	8102	8102	1.08	6164.33	3994.48	3005.41
DBAP-5B-35S-3/5-1	11393	11393	11393	0.60	1258.83	616.28	19.71
DBAP-5B-35S-3/5-2	11657	11657	11657	0.60	1218.61	1038.59	28.50
DBAP-5B-35S-3/5-3	11206	11206	11206	0.60	2220.77	1137.86	64.73
DBAP-5B-35S-5/8-1	10744	**10740**	**10740**	0.60	1388.393	807.61	22.06
DBAP-5B-35S-5/8-2	10990	10990	10990	0.60	1707.30	851.84	58.93
DBAP-5B-35S-5/8-3	10602	10602	10602	0.60	2036.23	1431.20	123.59
DBAP-5B-35S-7/8-1	17761	17761	17761	0.87	147.07	162.51	1.27
DBAP-5B-35S-7/8-2	18579	18579	18579	0.87	201.70	202.67	2.17
DBAP-5B-35S-7/8-3	17910	17910	17910	0.87	219.13	161.92	2.50
DBAP-5B-40S-1/2-1	9893	**9868**	**9868**	3.71	9414.15	7017.59	4725.28
DBAP-5B-40S-1/2-2	9463	9463	9463	3.71	18865.80	18744.00	8119.95
DBAP-5B-40S-1/2-3	11778	11778	11778	3.71	7850.98	4359.14	1554.27
DBAP-5B-40S-3/5-1	14168	14168	14168	0.98	4249.30	2229.36	109.25
DBAP-5B-40S-3/5-2	13601	13601	13601	0.98	6589.31	3481.00	574.53
DBAP-5B-40S-3/5-3	16043	16043	16043	0.98	2841.72	1938.50	184.30
DBAP-5B-40S-5/8-1	13274	**13263**	**13263**	2.29	2818.75	2109.00	243.20
DBAP-5B-40S-5/8-2	12742	**12737**	**12737**	2.29	7298.20	5236.94	970.86
DBAP-5B-40S-5/8-3	15201	15201	15201	2.29	3354.25	1814.06	252.81
DBAP-5B-40S-7/8-1	23341	23341	23341	0.07	343.86	388.344	3.00
DBAP-5B-40S-7/8-2	22511	22511	22511	0.07	851.78	758.56	7.32
DBAP-5B-40S-7/8-3	24875	24875	24875	0.07	271.11	350.84	2.24

Table 4.4: Results of DBAP instances for 5 berths-45 ships and 5 berths-50 ships with heuristic CG

	Objective Function Values			Computation time (Seconds)			
Problem Instance	VNS	CG	Exact	VNS	CG	CG Heuristic	Exact
DBAP-5B-45S-1/2-1	13843	13843	13843	3.54	20641.60	13625.10	13843.00
DBAP-5B-45S-1/2-2	12674	12674	12674	3.54	36457.50	29271.00	44686.10
DBAP-5B-45S-1/2-3	15205	15205	15205	3.54	9931.91	7401.14	1020.44
DBAP-5B-45S-3/5-1	19254	19254	19254	2.74	9822.77	4920.19	602.48
DBAP-5B-45S-3/5-2	17866	17866	17866	2.74	13080.90	7895.16	436.06
DBAP-5B-45S-3/5-3	20363	20363	20363	2.74	7974.00	3509.25	105.58
DBAP-5B-45S-5/8-1	18145	18145	18145	3.68	11029.10	6654.13	520.02
DBAP-5B-45S-5/8-2	16797	16797	16797	3.68	15773.50	9056.59	1372.97
DBAP-5B-45S-5/8-3	19305	19305	19305	3.68	8436.45	4567.41	135.25
DBAP-5B-45S-7/8-1	30697	30697	30697	0.36	1181.54	851.05	6.89
DBAP-5B-45S-7/8-2	29186	29186	29186	0.36	2824.97	1953.44	13.02
DBAP-5B-45S-7/8-2	31085	31085	31085	0.36	517.08	345.67	3.18
DBAP-5B-50S-1/2-1	17738	**17733**	**17733**	3.71	47700.00	18093.30	6771.73
DBAP-5B-50S-1/2-2	17380	17380	(**)	3.71	41564.70	20976.00	(**)
DBAP-5B-50S-1/2-3	18205	18205	18205	3.71	27059.90	13641.50	2491.28
DBAP-5B-50S-3/5-1	24514	24514	24514	2.80	22202.20	11194.30	1194.64
DBAP-5B-50S-3/5-2	24045	**24036**	**24036**	2.80	20518.20	9662.84	425.09
DBAP-5B-50S-3/5-3	24955	24955	24955	2.80	12314.50	5200.11	306.56
DBAP-5B-50S-5/8-1	23088	23088	23088	3.89	20502.90	11321.60	716.92
DBAP-5B-50S-5/8-2	22715	22715	22715	3.89	21184.70	8281.11	471.92
DBAP-5B-50S-5/8-3	23590	23590	23590	3.89	12308.30	6812.20	386.55
DBAP-5B-50S-7/8-1	38416	38416	38416	1.63	1534.42	1522.30	8.66
DBAP-5B-50S-7/8-2	37898	37898	37898	1.63	1584.78	1453.10	8.00
DBAP-5B-50S-7/8-3	38813	38813	38813	1.63	2514.06	1627.33	13.92

(**)Solution cannot be found because of "out of memory" error.

Table 4.5: Results of DBAP instances for 10 berths-40 ships, 10 berths-45 ships and 10 berths-50 ships with heuristic CG

	Objective Function Values			Computation time (Seconds)			
Problem Instance	VNS	CG	Exact	VNS	CG	CG Heuristic	Exact
DBAP-10B-40S-1/2-1	6373	6373	6373	1.33	4054.36	2484.95	1095.14
DBAP-10B-40S-1/2-2	6586	**6585**	**6585**	1.33	5661.38	2893.97	890.66
DBAP-10B-40S-1/2-3	6058	**6052**	**6052**	1.33	5348.92	2634.22	2838.56
DBAP-10B-40S-3/5-1	8432	8432	8432	1.79	2892.11	976.87	47.83
DBAP-10B-40S-3/5-2	8586	8586	8586	1.79	2393.42	906.03	58.10
DBAP-10B-40S-3/5-3	7925	**7924**	**7924**	1.79	2446.34	996.75	83.73
DBAP-10B-40S-5/8-1	7997	7997	7997	1.25	2597.78	1063.69	64.67
DBAP-10B-40S-5/8-2	8171	8171	8171	1.25	2448.36	1029.58	51.06
DBAP-10B-40S-5/8-3	7536	7536	7536	1.25	3009.14	1143.11	86.67
DBAP-10B-40S-7/8-1	12818	12818	12818	1.20	438.70	131.09	4.06
DBAP-10B-40S-7/8-2	12949	**12947**	**12947**	1.20	685.28	142.34	5.55
DBAP-10B-40S-7/8-3	12042	12042	12042	1.20	760.52	152.23	7.16
DBAP-10B-45S-1/2-1	7886	**7876**	(**)	2.99	10901.00	4961.33	(**)
DBAP-10B-45S-1/2-2	6446	6446	(**)	2.99	24615.30	3476.32	(**)
DBAP-10B-45S-1/2-3	7143	**7135**	(**)	2.99	5546.34	13241.50	(**)
DBAP-10B-45S-3/5-1	10527	**10516**	**10516**	2.13	4603.72	1019.64	6356.25
DBAP-10B-45S-3/5-2	8841	8841	(**)	2.13	10898.30	2093.45	(**)
DBAP-10B-45S-3/5-3	9689	9689	9689	2.13	7402.84	1547.78	144.84
DBAP-10B-45S-5/8-1	9938	9938	9938	3.19	7577.30	1556.22	97.22
DBAP-10B-45S-5/8-2	8298	**8297**	(**)	3.19	9397.80	2159.26	(**)
DBAP-10B-45S-5/8-3	9152	**9143**	**9143**	3.19	7634.28	1647.37	242.42
DBAP-10B-45S-7/8-1	16081	**16080**	**16080**	1.13	1616.03	273.81	15.92
DBAP-10B-45S-7/8-2	14471	14471	14471	1.13	3259.05	568.79	27.64
DBAP-10B-45S-7/8-3	15472	15472	15472	1.13	1686.53	399.53	13.61
DBAP-10B-50S-1/2-1	7060	7060	(**)	3.10	60528.50	30980.30	(**)
DBAP-10B-50S-1/2-2	9370	9370	(**)	3.10	27952.00	8836.73	(**)
DBAP-10B-50S-1/2-3	9020	9020	(**)	3.10	33527.20	13948.60	(**)
DBAP-10B-50S-3/5-1	9966	**9961**	(**)	3.53	30238.80	7728.98	(**)
DBAP-10B-50S-3/5-2	12631	**12628**	(**)	3.53	12257.80	3065.83	(**)
DBAP-10B-50S-3/5-3	12155	**12153**	(**)	3.53	21270.60	5060.73	(**)
DBAP-10B-50S-5/8-1	8177	**8172**	(**)	3.33	32360.30	8644.39	(**)
DBAP-10B-50S-5/8-2	10740	10740	(**)	3.33	11345.20	6607.11	(**)
DBAP-10B-50S-5/8-3	11076	**11068**	(**)	3.33	11198.80	5123.20	(**)
DBAP-10B-50S-7/8-1	15753	**15749**	**15749**	2.04	5703.70	1809.47	45.57
DBAP-10B-50S-7/8-2	18544	18544	18544	2.04	2259.80	511.03	14.42
DBAP-10B-50S-7/8-3	18931	**18930**	**18930**	2.04	1800.38	340.23	12.86

(**)Solution cannot be found because of "out of memory" error.

Chapter 5

CONCLUSIONS & FUTURE WORK

5.1 Conclusions

In this thesis, we studied a well-known optimization problem which is Dynamic Berth Allocation Problem (DBAP). For the DBAP, we applied a hybrid Column Generation (CG) algorithm where subproblem is solved heuristically or exactly if necessary. This research is the first study that proposes a CG for the solution of Berth Allocation Problems (BAP). Since DBAP can be thought as a large scale MIP problem, we were motivated by the promising results of the application of CG heuristic algorithm to the large scale MIP problems such as, cutting stock, bin-packing, crew scheduling, multi-comodity network flow and vehicle routing. The decomposition technique which is successfully applied to several optimization problems has been utilized. The DBAP has been decomposed into two subproblems. The master problem is a set partitioning problem with assignment constraints and the subproblem is a Mixed Integer Programming (MIP) problem with the complicating constraints of the DBAP. The CG technique has been applied to solve the relaxed master problem (RMP) and a hybrid algorithm has been designed in order to generate the columns in the subproblem. It is observed that the linear master problem (i.e, relaxation of RMP) is solved easily. Although the computation times for the master problem increase as the number of columns increase, the total solution times spent in solving the relaxation of the master problem is always negligible compared to the solution times required for the subproblems. Hence, solving the subproblem efficiently is very crucial for the performance of the CG heuristic algorithm. For this reason, we introduced a heuristic algorithm for the subproblem which is the most time-consuming part of the CG heuristic algorithm. We reduced the subproblem to an assignment problem by ignoring the complicating constraints and related variables. Afterwards, we applied a local search heuristic to the solution obtained from the assignment problem. When the result of the master problem is repeated for a predetermined number of iterations, heuristic algorithm is changed with the exact procedure for finding the necessary

columns to prevent stucking in a local optima. If this modification does not result in a change in the objective function value of the master problem, algorithm terminates with the current best integer solution by solving the master problem as an integer problem.

Previously, a Variable Neighborhood Search (VNS) heuristic was proposed in the literature with good solution quality and computation times [14]. Hence, we compared our method with this VNS algorithm. We observe that our proposed CG heuristic algorithm also gives good quality solutions. When we analyzed the results, it is observed that the CG heuristic algorithm performs well especially on the harder and larger problem instances. Compared to the VNS, computation times are not better for the proposed CG heuristic algorithm. However, CG heuristic algorithm outperforms VNS algorithm in 24 of the 84 instances in terms of the objective function value. DBAPis not solvable exactly in 15 of 84 instances because the size of the branch-and-bound tree causes memory error and we are not sure that proposed CG heuristic algorithm reaches the optimal solution for these problem instances. However, it provides better solutions than VNS algorithm in 8 of these 15 problem instances in terms of the solution quality.

We can conclude that the CG heuristic algorithm requires longer computation times for all of the tested problem instances but it always gives good quality solutions even for large problems. The CG heuristic algorithm can be a good tool to find a good quality solution for larger and harder instances since it decomposes the problems and deals with the smaller parts of the problems subsequently. It is superior over VNS algorithm for larger and harder instances as this can be seen in computational results. It is also observed that the solution quality gap between CG heuristic algorithm and VNS algorithm widens as the size of the problem instances grow. If the algorithm can be improved so that its computation time will decrease, it will become the most prefarable tool for the DBAP even for the small problem instances. In order to do so, a well-developed heuristic for the subproblem can be developed since the structure of the subproblem of the DBAP is not suitable for fast exact solution methods. Therefore, the CG heuristic algorithm is a promising technique especially for the larger and harder problem insantances of the DBAP.

5.2 Future Work

As our proposed solution procedure uses the assignment problem to find an initial schedule, harder but still easily solvable problems can be found and applied to initiate the heuristic algorithm. Doing so improves the quality of the initial solution and subsequently decreases the overall run time of the problem. Another extension can be made by constructing a metaheuristic algorithm, such as tabu search or genetic algorithm, instead of local search algorithm used in the thesis. Developing such metaheuristics may improve both the solution quality and the run time of the problem if one can build those heuristics by considering the specific characteristics of the DBAP. However, it is difficult to construct such smarter metaheuristics as they are more complex than other search algorithms.

BIBLIOGRAPHY

[1] J.F. Cordeau, G. Laporte, P. Legato, and L. Moccia, "Models and Tabu Search Heuristics for the Berth Allocation Problem," *Transportation Science*, vol. 39, no. 4, pp. 526–538, 2005.

[2] I.F.A. Vis and R. de Koster, "Transshipment of Containers at a Container Terminal: An Overview," *European Journal of Operational Research*, vol. 147, no. 1, pp. 1–16, 2003.

[3] D. Steenken, S. Voß, and R. Stahlbock, "Container Terminal Operation and Operations Research-a Classification and Literature Review," *OR Spectrum*, vol. 26, no. 1, pp. 3–49, 2004.

[4] G. Giallombardo, L. Moccia, M. Salani, and I. Vacca, "The tactical Berth Allocation Problem with Quay Crane Assignment and Transshipment-related Quadratic Yard Costs," in *Proceedings of the European Transport Conference (ETC)*, 2008.

[5] H.O. Günther and K.H. Kim, "Container Terminals and Terminal Operations," *OR Spectrum*, vol. 28, no. 4, pp. 437–445, 2006.

[6] S. Irnich and G. Desaulniers, *Shortest Path Problems With Resource Constraints*, Springer, 2004.

[7] B. Kallehauge, J. Larsen, O.B.G. Madsen, and M.M. Solomon, "Vehicle Routing Problem with Time Windows," *Column Generation*, pp. 61–98, 2005.

[8] H. Ben Amor and J.M. Valerio de Carvalho, "Cutting Stock Problems," *Column Generation*, pp. 131–161.

[9] D. Klabjan, *Large Scale Models in the Airline Industry*, Springer, 2005.

[10] M. Christiansen and B. Nygreen, "Robust Inventory Ship Routing by Column Generation," *Column Generation*, pp. 197–224, 2005.

[11] M.M. Sigurd, N.L. Ulstein, B. Nygreen, and D.M. Ryan, "Ship Scheduling with Recurring Visits and Visit Separation Requirements.," *Column Generation*, pp. 225–245, 2005.

[12] M. van den Akker, H. Hoogeveen, and S. van de Velde, "Applying Column Generation to Machine Scheduling," *Column Generation*, pp. 303–330, 2005.

[13] P. Hansen and C. Oğuz, *A Note on Formulations of the Static and Dynamic Berth Allocation Problems*, Groupe d'études et de recherche en analyse des décisions, 2003.

[14] P. Hansen, C. Oğuz, and N. Mladenović, "Variable Neighborhood Search for Minimum Cost Berth Allocation," *European Journal of Operational Research*, vol. 191, no. 3, pp. 636–649, 2008.

[15] M.L. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Springer, 2008.

[16] M.R. Garey, D.S. Johnson, et al., *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman San Francisco, 1979.

[17] A. Imai, X. Sun, E. Nishimura, and S. Papadimitriou, "Berth Allocation in a Container port: Using a Continuous Location Space Approach," *Transportation Research Part B*, vol. 39, no. 3, pp. 199–221, 2005.

[18] A. Imai, K.I. Nagaiwa, and C.W. Tat, "Efficient Planning of Berth Allocation for Container Terminals in Asia," *Journal of Advanced Transportation*, vol. 31, no. 1, pp. 75–94, 1997.

[19] A. Imai, E. Nishimura, and S. Papadimitriou, "The Dynamic Berth Allocation Problem for a Container Port," *Transportation research part B*, vol. 35, no. 4, pp. 401–417, 2001.

[20] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, 1998.

[21] E. Nishimura, A. Imai, and S. Papadimitriou, "Berth Allocation Planning in the Public Berth System by Algorithms," *European Journal of Operational Research*, vol. 131, no. 2, pp. 282–292, 2001.

[22] A. Imai, E. Nishimura, M. Hattori, and S. Papadimitriou, "Berth Allocation at Indented Berths for Mega-containerships," *European Journal of Operational Research*, vol. 179, no. 2, pp. 579–593, 2007.

[23] A. Imai, E. Nishimura, and S. Papadimitriou, "Berthing Ships at a Multi-user Container Terminal with a Limited Quay Capacity," *Transportation Research Part E*, vol. 44, no. 1, pp. 136–151, 2008.

[24] A. Imai, E. Nishimura, and S. Papadimitriou, "Berth Allocation with Service Priority," *Transportation Research Part B*, vol. 37, no. 5, pp. 437–457, 2003.

[25] A. Lim, "The Berth Planning Problem," *Operations Research Letters*, vol. 22, no. 2-3, pp. 105–110, 1998.

[26] Y.M. Park and K.H. Kim, "A Scheduling Method for Berth and Quay Cranes," *OR Spectrum*, vol. 25, no. 1, pp. 1–23, 2003.

[27] C. Liang, Y. Huang, and Y. Yang, "A Quay Crane Dynamic Scheduling Problem by Hybrid Evolutionary Algorithm for Berth Allocation Planning," *Computers & Industrial Engineering*, vol. 56, no. 3, pp. 1021–1028, 2009.

[28] A. Imai, H.C. Chen, E. Nishimura, and S. Papadimitriou, "The Simultaneous Berth and Quay Crane Allocation Problem," *Transportation Research Part E*, vol. 44, no. 5, pp. 900–920, 2008.

[29] M.P. de Aragão and E. Uchoa, "Integer Program rRformulation for Robust Branch-and-Cut-and-Price Algorithms," in *Proceedings of the Conference Mathematical Program in Rio: A Conference in Honour of Nelson Maculan*, 2003, pp. 56–61.

[30] G.L. Nemhauser and S. Park, "A Polyhedral Approach to Edge Coloring.," *Operations Research Letters*, vol. 10, no. 6, pp. 315–322, 1991.

[31] F. Vanderbeck, "Lot-Sizing with Start-up Times," *Management Science*, vol. 44, no. 10, pp. 1409–1425, 1998.

[32] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon, and F. Soumis, "2-path Cuts for the Vehicle Routing Problem with Time Wndows," *Transportation Science*, vol. 33, no. 1, pp. 101–116, 1999.

[33] J.M. Van den Akker, C.A.J. Hurkens, and M.W.P. Savelsbergh, "Time-Indexed Formulations for Machine Scheduling Problems: Column Generation," *INFORMS Journal on Computing*, vol. 12, no. 2, pp. 111–124, 2000.

[34] C. Barnhart, C.A. Hane, and P.H. Vance, "Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems," *Operations Research*, vol. 48, no. 2, pp. 318–326, 2000.

[35] D. Kim, C. Barnhart, K. Ware, and G. Reinhardt, "Multimodal Express Package Delivery: A Service Network Design Application," *Transportation Science*, vol. 33, no. 4, pp. 391–407, 1999.

[36] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance, "Branch-and-Price: Column Generation for Solving Huge Integer Programs," *Operations Research*, vol. 46, no. 3, pp. 316–329, 1998.

[37] G.B. Dantzig and P. Wolfe, "Decomposition Principle for Linear Programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[38] A.M. Geoffrion, "Lagrangean Relaxation For Integer Programming," *Mathematical programming studies*, vol. 2, pp. 82–114, 1974.

[39] T.L. Magnanti, J.F. Shapiro, and M.H. Wagner, "Generalized Linear Programming Solves the Dual," *Management Science*, vol. 22, no. 11, pp. 1195–1203, 1976.

[40] M. Savelsbergh, "A Branch-and-Price Algorithm for the Generalized Assignment Problem," *Operations Research*, pp. 831–841, 1997.

[41] J.F. Shapiro, *Mathematical Programming: Structures and Algorithms*, Wiley New York, 1979.

[42] M.E. Lubbecke and J. Desrosiers, "Selected Topics in Column Generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[43] P.J.M. Meersmans and R. Dekker, *Operations Research Supports Container Handling*, Econometric Institute, Erasmus University Rotterdam]; Erasmus University [Host], 2001.

[44] R.I. Peterkofsky and C.F. Daganzo, "A Branch and Bound Solution Method for the Crane Scheduling Problem," *Transportation Research B*, vol. 24, no. 3, pp. 159–172, 1990.

[45] R. Jonker and A. Volgenant, "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[46] D.P. Bertsekas, "Auction Algorithms for Network Flow Problems: A Tutorial Introduction," *Computational Optimization and Applications*, vol. 1, no. 1, pp. 7–66, 1992.

[47] J.L. Bruno, E.G. Coffman, and R. Sethi, "Scheduling Independent Tasks to Reduce Mean Finishing Time," *Communications of the ACM*, vol. 17, no. 7, pp. 382–387, 1974.

[48] J.K. Lenstra, A.H.G. Kan, and P. Brucker, "Complexity of Machine Scheduling Problems," in *Studies in Integer Programming: Proceedings of the Institute of Operations Research Workshop, Sponsored by IBM, University of Bonn, Germany, Sept. 8-12, 1975*. North Holland, 1977, vol. 1, pp. 343–362.

[49] Z.L. Chen and W.B. Powell, "Solving Parallel Machine Scheduling Problems by Column Generation," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 78–94, 1999.

[50] J.M. Van Den Akker, J.A. Hoogeveen, and S.L. Van De Velde, "Parallel machine scheduling by column generation," *Operations Research*, vol. 47, no. 6, pp. 862–872, 1999.

[51] P.C. Gilmore and R.E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem-Part II," *Operations Research*, vol. 11, no. 6, pp. 863–888, 1963.

[52] CPLEX I., "11.0 Users Manual," *ILOG SA, Gentilly, France*, 2008.