SOLVING INTEGRATED BERTH ALLOCATION AND CRANE ASSIGNMENT PROBLEM USING A TABU SEARCH METAHEURISTIC

by

Zeynep Şuvak

B.S., Industrial Engineering, Boğaziçi University, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University
2013

ACKNOWLEDGEMENTS

I would like to express my special thanks to my supervisor Prof. Necati Aras for his valuable guidance and encouragement throughout my undergradutae and graduate studies. I feel really fortunate to work with him and this work is not completed without his precious support and inspiring suggestions.

I am very grateful to Assist. Prof. Yavuz Boğaç Türkoğulları for his help during my research and Prof. İ. Kuban Altınel for his time and effort to take part in my thesis committee. Moreover, I would like to acknowledge to Düzce University for the concerned and understanding attitude during my thesis study.

I am also very thankful to all my friends and work fellows for their endless support, encouragement and the great times we had together.

No words can be found to express my gratitude for my family. My mother always give me peace with his affectionate presence and I can be just thankful for her being *my* mother. Also, I would like to thank my father who has the lion's share for all successes I had, throughout my life. I could not be the person who I am if I do not have them. Murat and Şevval you are the best brother and sister ever. I am extremely lucky to have you.

Finally, my heart is too full to tell my feelings about my husband, Vedat. You have always been by my side with your patient, considerate and encouraging attitude. You made it possible, thank you.

ABSTRACT

SOLVING INTEGRATED BERTH ALLOCATION AND CRANE ASSIGNMENT PROBLEM USING A TABU SEARCH METAHEURISTIC

The volume of container transportation and the throughput rate at terminals has been increased vastly in recent years. Container terminal operators have to utilize their limited resources efficiently in order to cope with the rising demand while satisfying the expectations of vessel operators. The management of integrated quayside operations is the primary concern of this study. Hence, two models one of which integrates berth allocation and quay crane (number) assignment problems (BACAP), and the other unifies berth allocation and specific quay crane assignment problems (BACASP) are provided. A tabu search algorithm and local improvement procedures are implemented to solve BACAP and the method is tested on 3 different instance groups which are generated where vessels arrive with changing frequencies. A quay crane schedule is obtained by solving the shortest path problem on the graph representation of a BACAP solution. As a consequence, TS algorithm given for BACAP is modified to a solution method for BACASP by incorporating the shortest path problem into the metaheuristic. The TS algorithm for BACASP is justified on the test instances which are previously generated for BACAP.

ÖZET

BÜTÜNLEŞİK RIHTIM ATAMA VE VİNÇ ATAMA PROBLEMİNİN BİR TABU ARAMA METASEZGİSELİ İLE ÇÖZÜLMESİ

Konteyner taşımacılığı ve limanlardaki konteyner trafiği son yıllarda büyük artış göstermiştir. Artan talebi ve deniz taşımacılığı yapan şirketlerin beklentilerini karşılayabilmek amacıyla, liman işletmeleri ellerindeki kısıtlı kaynakları etkili bir şekilde kullanmak zorundadırlar. Bu çalışmanın asıl konusu bütünleşik kıyı operasyonlarının yönetilmesidir. Sonuç olarak, bu raporda bütünleşik rıhtım ve vinç sayısı atama problemi (RAVAP) ile bütünleşik rıhtım ve özellikli vinç atama problemi (RAÖVAP) olmak üzere iki model verilmiştir. RAVAP için önerilen tabu arama (TA) algoritması ve yerel iyileştirme yöntemleri gemilerin geliş sıklığını değiştirerek elde edilen 3 farklı veri grubuyla denenmiştir. Bir RAVAP çözümünün çizge gösterimi üzerinde en kısa yol problemi çözülerek bir vinç çizelgesi elde edilebilir. Bundan hareketle, RAVAP için önerilen TA algoritması en kısa yol problemini de içine dahil ederek RAÖVAP için bir çözüm yöntemine dönüştürülmüştür. RAÖVAP için önerilen TA algoritması daha önce bahsedilen veri grupları üzerinde denenmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES ix
LIST OF SYMBOLS xi
LIST OF ACRONYMS/ABBREVIATIONS xiii
1. INTRODUCTION 1
2. LITERATURE SURVEY 4
3. INTEGRATED BERTH ALLOCATION AND QUAY CRANE ASSIGNMENT PROBLEM 14
3.1. Model Assumptions 14
3.2. Cost Structure 18
3.3. Solution Procedure: A Tabu Search Metaheuristic 20
3.3.1. Solution Representation and Generation 22
3.3.2. Initial Solution 25
3.3.3. Outer Search 26
3.3.4. Inner Search 27
3.4. Local Improvement Algorithms 28
3.4.1. Smoothing Quay Crane Utilization 28
3.4.2. Spatial Improvement Procedure 31
3.4.3. Temporal Improvement Procedure 34
3.4.3.1. Checking Berthing Times 34
3.4.3.2. Checking Departure Times 36
3.4.4. Integration of Local Improvement Procedures to Tabu Search . 37
3.5. Experimental Results for BACAP 38
3.5.1. Determination of Tabu Search Parameters 38
3.5.2. Instance Generation 38

4. INTEGRATED BERTH ALLOCATION AND SPECIFIC QUAY CRANE ASSIGNMENT PROBLEM 50
4.1. Additional Assumptions 50
4.1.1. Objective Function of BACASP 52
4.1.2. Solution Method 54
4.1.3. Integration of Shortest Path Problem to Tabu Search Algorithm 57
4.2. Experimental Results for BACASP 59
5. CONCLUSION 67
REFERENCES 69

LIST OF FIGURES

Figure 3.1. A Feasible Solution. 17

Figure 3.2. Solution Obtained from $P = \{1, 2, 3, 4\}$ and $R = \{0, 0, 0, 0\}$. 23

Figure 3.3. Solution Obtained from $P_1 = \{2, 1, 3, 4\}$ and $R_1 = \{0, 0, 0, 0\}$ and $P_2 = \{2, 1, 3, 4\}$ and $R_2 = \{0, 0, 1, 1\}$. 24

Figure 3.4. Neighboring Solution Generation of Outer Search. 27

Figure 3.5. Quay Crane Smoothing without Changing Completion Times. . . 29

Figure 3.6. Quay Crane Smoothing Changing Completion Times. 30

Figure 3.7. Different Solutions for Different (P, R) Pairs. 33

Figure 3.8. Optimal Solution After Spatial Improvement. 34

Figure 3.9. Checking and Improving Berthing Times. 35

Figure 3.10. Checking and Improving Departure Times. 36

Figure 4.1. A Feasible Solution for BACASP. 52

Figure 4.2. A Feasible Solution Before Finding Specific Quay Cranes. 56

Figure 4.3. The Directed Graph to Obtain a Quay Crane Schedule. 57

Figure 4.4. The BACASP Solution. 58

LIST OF TABLES

Table 3.1. Parameters of the model. 16

Table 3.2. Decision variables of the model. 16

Table 3.3. Cost coefficients in the objective function. 19

Table 3.4. A sample. 20

Table 3.5. A sample instance with three vessels. 32

Table 3.6. TS parameters. 38

Table 3.7. The instance given in (Zhang *et al.*, 2010) with $c_{i4} = 0$. 40

Table 3.8. Test results for data given in (Zhang *et al.*, 2010) with $c_{i4} = 10$. . . 40

Table 3.9. Test results for loose arrival times with $c_{i4} = 0$. 43

Table 3.10. Test results for moderate arrival times with $c_{i4} = 0$. 44

Table 3.11. Test results for tight arrival times with $c_{i4} = 0$. 45

Table 3.12. Test results for loose arrival times with $c_{i4} = 10$. 46

Table 3.13. Test results for moderate arrival times with $c_{i4} = 10$. 47

Table 3.14. Test results for tight arrival times with $c_{i4} = 10$. 48

Table 3.15. Test results for some instances with $c_{i4} = 50$. 49

Table 3.16. Test results for some instances with $c_{i4} = 100$. 49

Table 4.1. Decision variables of the BACASP. 51

Table 4.2. Cost coefficients of the objective function in BACASP. 53

Table 4.3. The instance given in (Zhang *et al.*, 2010) with $c_4 = 0$. 61

Table 4.4. Test results for loose arrival times with $c_4 = 10$. 62

Table 4.5. Test results for moderate arrival times with $c_4 = 10$. 63

Table 4.6. Test results for tight arrival times with $c_4 = 10$. 64

Table 4.7. Test results for some instances with $c_4 = 50$. 65

Table 4.8. Test results for some instances with $c_4 = 100$. 66

LIST OF SYMBOLS

AC_j	Number of idle cranes in period j
B	Number of berth sections
c_{i1}	Cost of one unit deviation from the desired berthing section for vessel i
c_{i2}	Cost of berthing one period later than the arrival time for vessel i
c_{i3}	Cost of departing one period later than the due time for vessel i
c_{i4}	Cost of changing the number of assigned cranes by one unit for vessel i
c_4	Cost of moving a specific quay crane
C_1	Maximum number of successive iterations in the outer search without improvement
C_2	Maximum number of successive iterations in the inner search without improvement
d_i	Due time of vessel i
e_i	Arrival time of vessel i
K	Number of vessel clusters to gernerate random priority list
$\underline{k}^i$	Minimum number of cranes that can be assigned to vessel i
$\overline{k}^i$	Maximum number of cranes that can be assigned to vessel i
l_i	Length of vessel i
lm_i	Leftmost crane assigned to vessel i
n_1	Number of generated neighboring solutions in one iteration of the outer search
n_2	Number of generated neighboring solutions in one iteration of the inner search
N	Number of total quay cranes
p_i	Berthing position of vessel i
P_{arr}	Priority list ordered according to increasing arrival time
P_{due}	Priority list ordered according to increasing due times

P_{rand}	Priority list ordered randomly
P_x	Priority list corresponding to solution x
R_x	Rule list corresponding to solution x
R_0	Rule list corresponding to zero vector
rm_i	Rightmost crane assigned to vessel i
q_i	Departure time of vessel i
S_s^i	The set of vessels which are simultaneously served with at least one vessel from V_s^i
s_i	Desired berthing position of vessel i
T	Number of time periods
t_i	Berthing time of vessel i
t^temp_i	The earliest time period that vessel i can berth at its desired berthing position
TL_1	Tabu list of outer search
TL_2	Tabu list of inner search
TT_1	Tabu duration of outer search
TT_2	Tabu duration of inner search
V	Number of vessels
V_s^i	The set of vessels which prevent vessel i to prevent earlier
v^max_i	The maximum number of allocated quay cranes to vessel i in any solution
w_i	Handling time of vessel i in QC-hours
y_{ij}	Number of cranes assigned to vessel i in period j
z	Objective value
θ_{ijkl}	1 if crane k is moved from i to j between periods l and $l+1$

LIST OF ACRONYMS/ABBREVIATIONS

BACAP	Berth Allocation and Quay Crane Assignment Problem
BACASP	Berth Allocation and Quay Crane (Specific) Assignment Problem
BAP	Berth Allocation Problem
CPU	Central Processing Unit
EDD	Earliest Due Date
FCFS	First Come First Served
GA	Genetic Algorithm
MA	Memetic Algorithm
MILP	Mixed Integer Linear Program
NP	Nondeterministic Polynomial Time
QC	Quay Crane
QCAP	Quay Crane Allocation Problem
QCAP-number	Quay Crane Assignment (Number) Problem
QCAP-specific	Quay Crane Assignment (Specific) Problem
SA	Simulated Annealing
SPP	Shortest Path Problem
TEU	Twenty-foot Equivalent Unit
TS	Tabu Search
VNS	Variable Neighborhood Search

1. INTRODUCTION

Beginning from the year 1951 when the earliest version of container ship is introduced, the share of containerized transportation is increased eventually. The container volume is expressed as TEU's. A TEU is equal to the capacity of a standard container of 20-ft (length)× 8-ft (width) × 8-ft 6-in (height). The container volume in global trade has reached 13.5 million TEU in 1980, 28.7 million in 1990, 68.7 million in 2000, 140.9 million 2010 and 151 million in 2011 (UNCTAD, 2012). In parallel with the development of containerization, the capacity of container ships is also increased. The first prototype which is called *converted cargo vessel* is 135m long with 500 TEU capacity. Today, the largest container ship, *New Panamax*, has reached to the length of 397m with the capacity of 14,500 TEU. This increase in the share of container transportation gives rise to throughput rate of container terminals. Since investing to resources in a container terminal such as berths, cranes, transportation vehicles and handling equipment is very costly, the efficient utilization of the available instruments gains importance in order to meet the growing demand while optimizing a performance measure.

The operations handled in a container terminal can be divided into two basic categories: Quayside and yardside operations. Although there exist a number of different problems both in the quayside and in the yard, this work focuses on the quayside problems. These problems arise from due to the allocation of limited resources. Berths, quay spaces where vessels are loaded and unloaded, are critical resources and optimal assignment of berths and service time intervals to vessels is an important issue at quayside, which is known as Berth Assignment Problem (BAP). Containers are unloaded from vessels by quay cranes and placed onto trucks, while containers are loaded to vessels following the reverse process. Quay cranes can move along the quayside usually on a rail system and trucks transfer the containers from the quayside to the storage area in the yard. Determining the number of quay cranes allocated to vessels for loading/unloading is called Quay Crane (Number) Assignment Problem (QCAP-number)

and the allocation of specific cranes to vessels is named as Quay Crane (Specific) Assignment Problem (QCAP-specific). If both number and set of cranes are determined in a single problem, then it is called QCAP.

Although the initial works about container terminals tackles with these problems one by one in a sequential manner, these problems are noticed to be strongly related. The recent papers mostly consider the integrated solution approaches for two or more problems in ports. Our main goal is to develop a procedure which produces near optimal results in a reasonable amount of time for the integrated problem of BAP and QCAP-number (BACAP) and the integrated problem of BAP and QCAP-specific (BACASP).

Container terminal related problems are mostly characterized by the model assumptions and these assumptions are the main factors which determine the difficulty of the problem. For instance BAP with static arrival times can be solved at optimality easily whereas the variant with dynamic arrival times are proved to be NP-hard in the literature. Similarly,the version with discrete berth layout is known to be less complicated than the version with continuous layout. Our problems assume continuous berth layout with dynamic arrival times. Quay crane allocation is variable in time and the handling times are dependent on the assignment of quay cranes. Each vessel is defined with a due time for departure and a desired berth segment. The exact methods presented in (Türkoğulları *et al.*, 2012a) and (Türkoğulları *et al.*, 2012b) are capable to find optimal values up to 60 vessels for BACAP and BACASP respectively. Due to the fact that required computational time and the allocated working memory are quite high for exact methods, a heuristic approach is applied here.

In this study, a tabu search algorithm is developed and a few local improvement procedures are provided to solve BACAP. In addition to tabu search and local procedures, a shortest path problem is solved to gain the quay crane schedule in BACASP. The experiments are carried out with different instances. The results are compared with the optimum or the best integer solution found by the method of (Türkoğulları

et al., 2012a) and (Türkoğulları *et al.*, 2012b).

The remaining of the thesis is organized as follows: A literature review about quayside operations is given in Chapter 2. The problem assumptions of BACAP and the proposed tabu search algorithm are elaborated and the experimental results are given in Chapter 3. In Chapter 4, BACASP model is introduced and the solution approach employing tabu search and Dijkstra's algorithm is presented. The numerical results are tabulated at the end of the section. Finally, the concluding remarks are stated in Chapter 5.

2. LITERATURE SURVEY

The problems encountered in container terminals have been studied in a significant amount of research papers. In the beginning, these problems are treated separately and solution approaches are proposed ignoring the effect of other problems. However, the interaction between the problems was taken into consideration and integrated solution approaches for two or more problems have been developed throughout the last decade.

In the BAP, a berth layout of the container terminal and the number of vessels to be served are given and the aim is to assign a berthing position and berthing time to each vessel within a planning horizon. The berth layout may consist of discrete berths where only one vessel can moor to one berth. Alternatively, the layout can be a continuous berth where each vessel can be positioned at any position of the quay. In general, the problems with discrete berth assumption are regarded as simpler than the ones with continuous berth layout. However, the former may be more suitable and realistic for some container terminals. Apart from these, hybrid berth layout where one berth can be allocated to more than one smaller vessel or more than one berth is allocated to one large vessel is also studied in the literature. Regardless of the layout type, no two vessels can occupy the same quay space simultaneously. All vessels must be served within the boundaries of the quay respecting the planning horizon to optimize an objective function. The objective can be minimizing waiting times, service times, tardiness, completion times, deviation from desired berthing positions etc. or a weighted sum of these measures. An extensive survey of container terminal related works can be found in (Bierwirth and Meisel, 2010) with a detailed classification scheme.

(Edmond and Maggs, 1978) and (Schonfeld and Frank, 1984) can be stated as the initial works about container terminal operations. These papers differ from the works following them in terms of the methods they use and the objective they pursue.

The main goal in (Edmond and Maggs, 1978) is to decide how to invest in berth construction and container handling equipment with the aid of simple queue models. The overall costs related to berths, cranes, storage yard, labor, and containers are minimized by providing an analytical model in (Schonfeld and Frank, 1984). Besides, (Lai and Shih, 1992), which is another early work, proposes three berth allocation policies and provides a measurement strategy using heuristics and computer simulation. Also, a mathematical model is developed in (Brown *et al.*, 1994) to obtain an optimal berth plan minimizing berth shifts of vessels during service.

The authors in (Imai *et al.*, 2001) use discrete berth layout and the handling time of a vessel depends on the berth it moors. They provide two models minimizing the total waiting and handling times of vessels. In the first model, vessels are ready at the port and can berth at any time, i.e., arrival times of vessels are static. This problem is reduced to classical assignment problem which can be solved to optimality. The second model assumes dynamic arrival times, where vessels cannot berth earlier than their arrival times. Since the second problem, which is formulated as a mixed integer linear program (MILP), cannot be solved at a reasonable amount of time, Lagrangean relaxation based heuristic is proposed.

The formulation with dynamic arrival times of (Imai *et al.*, 2001) is upgraded to more compact MILP model in (Hansen and Oğuz, 2006) so that the new model can find the optimal solution for larger instances. Additionally, the dynamic BAP with a single berth is proved to be NP-hard by reducing it to the problem of minimizing the total completion time of jobs on a single machine with release dates.

The vessels are directed to an external port in (Imai *et al.*, 2008b) after a pre-determined due time is reached. Assumptions are almost the same with those of (Imai *et al.*, 2001), except that the objective is to minimize the number of rejected/directed vessels, given maximum acceptable waiting times for each vessel. A genetic algorithm is provided to solve BAP with dynamic arrival times.

Instead of minimizing the loss time due to waiting or tardiness, (Hansen *et al.*, 2008) proposes to minimize the total weighted costs for the same problem in (Hansen and Oğuz, 2006). Dynamic arrival times, discrete berth layout and berth dependent operation times are the assumptions. Each vessel is defined with a desired berthing position and deviation from that berthing position causes transportation costs within the terminal. The terminal operator wins premium for early completion times and pay fine for overdue. Moreover, the cost of dissatisfaction because of waiting times is included. The authors provide several solution approaches: Variable Neighborhood Search (VNS), Genetic Algorithm (GA) and Memetic Algortihm (MA). They conclude that VNS outperforms the other two heuristics.

The dynamic version of BAP is also studied in (Cordeau *et al.*, 2005), to minimize the weighted sum of waiting and handling times by assuming berth dependent operation times. MILP formulations are given for both discrete and continuous berth cases. Although the models can be solved optimally for small instances, a tabu search heuristic is developed to solve instances with realistic size. For discrete case, tabu search algorithm gives better solutions than CPLEX within a time window. For continuous case, tabu search is compared with a first-come-first-served (FCFS) based constructive algorithm. (Mauri *et al.*, 2008) applies a column generation approach for the same problem with discrete layout and reports that this new method produces better results in shorter times than the tabu search algorithm applied in (Cordeau *et al.*, 2005).

Each discrete berth is defined with a length and depth in (Han *et al.*, 2006) in addition to berth dependent operation times. Vessels can moor to a berth if and only if the berth is in appropriate physical condition for that vessel in terms of length and depth. A nonlinear program is developed for dynamic BAP. Two heuristics, one of which is GA, are offered. The other method is the hybrid of GA and Simulated Annealing (SA). The hybrid method is shown to be superior. The authors in (Zhou *et al.*, 2006) also use similar arguments for berths. However they assumed that arrival times and operation times of vessels are stochastic variables since they might be affected by weather, crane productivities, container distribution in the vessel etc. in real life.

Continuous berth layout is introduced in (Li *et al.*, 1998) by generalizing the BAP to "multiple-jobs-on-one-processor" pattern where jobs are vessels and processors are berths. According to this pattern, multiple jobs can be processed on a processor as long as they do not exceed the capacity of the processor. First-Fit-Decreasing heuristic is provided for static BAP with fixed handling times to minimize the schedule makespan. (Guan *et al.*, 2002) solves the same problem to minimize the total weighted completion times. In the solution approach, vessels are ordered according to a priority rule and groups of vessels are allocated to berth segments greedily.

The weighted sum of waiting and operation times is minimized in (Guan and Cheung, 2004), assuming dynamic arrival times and continuous berth layout. Handling times of vessels are fixed. A hybrid heuristic approach is adopted where a tree-search procedure and pair-wise exchange heuristic are combined. The same problem with a different objective can be found in (Wang and Lim , 2007). The aim is to minimize the weighted sum of rejected vessels, deviation from desired berthing position and waiting time of vessels before berthing. Stochastic beam search is shown to be more accurate than the deterministic beam search in solving BAP. Also the authors use real-life data taken from Singapore Port to test their proposed solution.

In (Park and Kim, 2002), BAP with fixed handling times, continuous berth layout, and dynamic arrival times is formulated as MILP and Lagrangean relaxation of the model is solved by subgradient optimization technique. The objective is to minimize the total weighted tardiness and the distance from the favorite berthing position of vessels. The same problem is solved using SA in (Kim and Moon, 2005). A different method which combines mathematical modeling and discrete event simulation is proposed in (Briano *et al.*, 2005).

The problem mentioned in (Park and Kim, 2002), is introduced with a peculiar objective function in (Lim, 1998). The vessels' berthing times are determined as their arrival times. Given the arrival and handling times, the minimum quay crane is searched satisfying the schedule within the planning horizon allowing no overlapping

of vessels. This problem is proved to be NP-complete by reducing it to set partitioning problem. In the graph representation of the problem, nodes stand for the vessels and there is an arc between two nodes if the corresponding vessels stay together at the port in any time period. The problem is extended by adding the constraint which forces vessels to respect the berth depths in (Lim, 1999) and solved by applying a greedy construction heuristic and a post optimal algorithm.

Total handling and waiting times of vessels are minimized in (Imai *et al.*, 2005). The berth layout is continuous, arrival times are dynamic and the handling times changes according to the berthing position of vessels. This problem is also proved to be NP-hard by reducing it to two dimensional cutting stock problem. This problem tries to obtain rectangular pieces from a sheet of stock material minimizing the trim loss. A vessel can be represented as a rectangle: one side stands for the length of the vessel and the other side represents the duration it spends at the berth.

BAP variants with dynamic arrival times mentioned above are studied also for hybrid berth layouts in the literature. Fixed handling times assumption is used to minimize the weighted sum of waiting times and deviation from desired berth position in (Moorthy and Teo, 2006). The case, where handling times are affected by berthing positions, is explored in (Imai *et al.*, 2007) minimizing the total waiting and operation times. Here, BAP is formulated for indented berths that are constructed perpendicular to the quay. Vessels can be served from both sides in this special berth type. Lastly, the version with berth depths can be found in (Nishimura *et al.*, 2001).

In QCAP, the quay cranes (QCs) are assumed to be identical and move along a rail track. They are allocated to vessels knowing where and when the vessel berths. By construction of rail system, QCs are prohibited to pass over each other. The aim is to fulfill the loading/unloading operations of all vessels in the schedule. If the berth layout is discrete, QCAP may not be a challenge since each berth has its own distinct cranes. Otherwise, the handling times of vessels are strongly correlated with the number of QCs allocated. This interdependence is considered in many research papers and the

workload of a vessel is expressed in terms of QC-hours. This problem targets to find either the number or the specific set of QCs. For simplicity, it can be assumed that the number or set of QCs is never changed within the service duration of vessel. This is called "time invariant" assignment in (Bierwirth and Meisel, 2010). Alternatively, "variable-in-time" assignment allowing changes can be used to increase the flexibility of operational decisions. Moreover, the number of allocated QCs may be bounded below by contracts between the vessel and terminal operators as well as bounded above by the length of the vessel. The objective is mostly to minimize the number of crane setups. QCAP is not regarded as a difficult problem so it is usually studied in the literature together with either BAP or Quay Crane Scheduling Problem (QCSP) that tries to schedule detailed operational tasks of loading/unloading.

As mentioned before, the integration of two or more problems has gained popularity in recent years. BAP and QCAP are integrated in (Lokuge and Alahakoon, 1999) so that the output of one problem is input to the other one in a feedback loop. Hybrid berth layout is used and handling times of vessels vary with the number of allocated QCs and berthing position. Each vessel is defined with an arrival time and a multi-agent system is proposed to minimize total waiting time and total tardiness separately.

BAP and QCAP-number are unified in a MILP formulation in (Meisel and Bierwirth, 2009). The berth layout is continuous and arrival times are dynamic. Operation times depend on the number of cranes utilized but QC productivity does not increase linearly. Each added QC has a decreasing marginal effect, since interference among QCs slows down the work performance. The crane assignment is "variable-in-time" and the distance between berthing position and storage area affects the handling time of vessels due-to internal truck traffic. The pursued objective is to minimize the weighted sum of tardiness, distance from desired berthing position, QC productivity loss and the speed-up cost to berth earlier than the expected arrival time. The authors present several procedures: a construction heuristic with local refinements, squeaky wheel optimization meta-heuristic and a tabu search meta-heuristic. Finally, the results are

compared with each other and with the method of (Park and Kim, 2003).

A MILP formulation, which aims to minimize the maximum crane capacity needed for each time window when vessels arrive periodically, is given in (Hendriks *et al.*, 2008). Each vessel has an acceptable waiting time before berthing and the handling times are related to the output of QC allocation. (Giallombardo *et al.*, 1978) also gives two mathematical formulations where the discrete berth layout is used. However it is reported that even small instances cannot be solved to optimality and further procedures are needed.

In (Park and Kim, 2003), BAP, QCAP-number and QCAP-specific are solved in two stages. In the first stage, Lagrangean relaxation of MILP model, which integrates BAP and QCAP-number, is solved utilizing subgradient optimization. The goal is to minimize the weighted sum of tardiness, speed-up cost, deviation from desired berthing position and waiting times. The output of the first stage, berth plan and number of assigned cranes, is used to specify the set of cranes in the second stage. The QCAP-specific problem tries to minimize the number of crane setups and a dynamic program is developed for this stage .

The approach in (Imai *et al.*, 2008a) is quite similar to the one in (Park and Kim, 2003). Here, BAP and QCAP-number is solved by a GA. The obtained berth plan and QCAP-number solution is used to check whether a feasible QCAP-specific solution exists. Otherwise, the infeasible solution is modified to make it feasible. These two problems work in a feedback loop until they give the minimum service time with a feasible crane schedule.

The container terminal management issues are decomposed into two parts in a case study of (Meier and Schumann, 2007). BAP is solved by berth planners minimizing total service times, QCAP and QCSP are solved by crane planners maximizing crane productivity. Since the service times are directly related to crane schedule, an iterative information flow is built between these two planning modules. The feedback loop is

ended after a pre-determined number of iterations.

Another integration approach is given in (Ak, 2008) with the goal of minimizing the sum of the total waiting times, handling times and tardiness. A monolithic MILP formulation is provided integrating these three problems. Since the model is not solvable in polynomial time, a nested tabu search algorithm is given which decides on the berth plan and specific crane assignment to the holds of vessels, where QCs are assigned to holds and leaves only when it finishes the work of that hold. The results are compared with a rule-of-thumb solution which can be generated by a terminal operator. These kind of monolithic solution procedures are accepted as a deeper way of integration than feedback loops.

In the literature, in some works BAP is excluded and only QCAP and QCSP are integrated. This approach originates in (Daganzo, 1989). The vessels arrive at different times and wait in a queue until a berth becomes available. QCs are allocated to holds of vessels and a vessel cannot depart before all its holds are processed. The paper presents an exact method which can solve instances with a few ships to minimize aggregate costs of vessel delay. For larger instances, an approximation method is also provided. A branch-and-bound method is given for the same problem in a later work of the author in (Peterkofsky and Daganzo, 1989). QC schedule, for a given set of parallel vessels, is obtained through applying a GA in (Tavakkoli-Moghaddam *et al.*, 2009) to minimize the sum of weighted finishing times of cranes and sum of weighted vessel tardiness.

A recent work deeply integrating BAP, QCAP-number and QCAP-specific can be found in (Zhang *et al.*, 2010). This paper stands apart from other works due to the assumptions it made. Cranes cannot operate at every berth segment because of cable lengths, so each crane is defined with a coverage range. Although the crane assignment is "variable-in-time", the number of adjustments is limited. Since too many adjustments reduce the applicability of the allocation, such a restriction makes sense. An MILP formulation is developed and a subgradient optimization algorithm is applied

to solve the problem. The problem instances are taken from real-life data.

BAP and QCAP-number is modeled in a monolithic binary integer program in (Türkoğulları *et al.*, 2012c). The crane assignment is assumed to be "time invariable" and the problem can be solved optimally up to 60 vessels. The same objective function presented in (Zhang *et al.*, 2010), is also used here. Another formulation including QCAP-specific is proposed but optimal solution cannot be obtained for all instances. Instead, a post-processing procedure, that checks whether the optimal solution of the first model satisfies the sufficient and necessary conditions for the optimality of the second model, is applied. If this condition is not satisfied, a cut is added to the model and the first formulation is solved iteratively until a solution which obeys the given optimality condition is obtained. The authors give a similar formulation integrating BAP and QCAP-number in (Türkoğulları *et al.*, 2012a), allowing "variable-in-time" crane assignment this time. Setup cost is incurred if the number of assigned cranes to a vessel is changed by one unit between two consecutive time periods. The model is solved optimally up to 60 vessels. A heuristic giving a QC schedule to the optimal plan is applied with the aim of minimum specific crane shift. The heuristic is proved to find optimal schedule if the input has a special characteristic.

In (Türkoğulları *et al.*, 2012b), the problem presented in (Türkoğulları *et al.*, 2012a) with a different QC setup definiton, is decomposed into a master and subproblem and cutting plane algorithm is applied. The setup cost is incurred if an idle QC begins to serve a vessel or a QC moves from a vessel to another or a QC goes idle after serving a vessel. The master problem solves BAP and QCAP-number and the subproblem solves QCAP-specific. The optimal value of the master problem is updated after the subproblem is solved and the master problem is run again with this new information. This feedback loop stops when the minimum total objective value is reached. The subproblem is modeled as a minimum cost network flow problem and solved by a branch-and-bound algorithm initially. However the second approach which develops a new graph and finds shortest path on it is proved to be a more efficient way to solve the subproblem. The optimal objective values obtained outperforms the

results of existing heuristics.

3. INTEGRATED BERTH ALLOCATION AND QUAY CRANE ASSIGNMENT PROBLEM

The problem of assigning berthing positions and service durations to vessels with the aim of optimizing a performance measure is called the berth assignment problem (BAP). The volume of international trade and particularly the share of containerized sea transportation have increased in recent years. This upward trend forces terminal operators to manage their vessel handling schedule efficiently. Otherwise, an inefficient schedule might cause longer waiting times, delays in service and consequently disappointment of vessel operators. Clearly, solving BAP alone does not guarantee an applicable schedule in real life. In practice, the quay crane assignment problem (QCAP-number), which allocates cranes to vessels at the port, is solved following the berth related decisions. However, the service duration and the departure times of vessels are directly related to the number of allocated quay cranes. Placing a vessel into an available berthing position does not mean that an idle quay crane is ready to serve that vessel. So, the restrictions imposed by quay cranes should be taken into account while allocating berths to obtain more implementable plans. The intimate nature of these two resources inspired the idea of integrating BAP and QCAP-number. The integrated problem of BAP and QCAP-number is called BACAP (Berth Allocation and Quay Crane Assignment Problem) in this report, following the naming convention used in (Türkoğulları *et al.*, 2012b).

3.1. Model Assumptions

Container terminal related problems are not uniform and the variants of the same problem with different assumptions can be very distinct from each other. Likewise, our BACAP model is characterized by the following assumptions:

- The berth layout is continuous and it is discretized by dividing it into equal berthing segments. Each berth segment corresponds to 50 meters.

- The planning horizon is discretized by dividing it into equal time periods. Each time period corresponds to an hour.
- A berth segment cannot be occupied by more than one vessels in a time period, i.e. overlapping is not allowed.
- Handling times depend on the number of cranes allocated and they are expressed in QC-hours.
- A quay crane can be assigned to only one vessel in a time period.
- A vessel cannot berth before its arrival time and the loading/unloading operations begin as soon as a vessel berths. The vessel departs upon the termination of loading/unloading operations.
- A vessel cannot change its berthing position during service and preemption is not allowed. However, the number of allocated quay cranes can be adjusted in time.
- The number of cranes that can be assigned to a vessel in a time period is limited from below by contracts and from above by the vessel length.The total number of allocated QCs cannot exceed the total number of QCs at the port.
- Each vessel is defined with a desired berthing position and a due time for departure.

The BACAP parameters and their concise definitions can be seen in Table 3.1. Also, Table 3.2 reveals the decision variables to be determined.

Once a problem is solved, the obtained berth plan can be represented in a time-space diagram as shown in Figure 3.1, where the horizontal and the vertical axes stand for time periods and berth segments, respectively. Each vessel is represented by a rectangle whose height is equal to the length of the vessel and width is equal to the time interval it stays at the berth. The berth sections occupied by a vessel and the berthing time of a vessel can be determined by only detecting the left-lower grid square of the corresponding rectangle. The y coordinate of the left-lower grid square represents the berthing position p_i of vessel i, and the x coordinate gives the berthing time, t_i. Furthermore, the number of cranes allocated to a vessel is shown by the number inside the rectangle. Dividing the rectangle by a vertical line means

Table 3.1. Parameters of the model.

Parameter	Definition
V	number of vessels
T	number of time periods
B	number of berthing segments
N	number of quay cranes available
$\underline{k}^i$	minimum number of cranes that can be assigned to vessel i
$\overline{k}^i$	maximum number of cranes that can be assigned to vessel i
l_i	length of vessel i
e_i	arrival time of vessel i
s_i	desired berthing section of vessel i
d_i	due time of vessel i
w_i	handling time of vessel i in QC-hours

Table 3.2. Decision variables of the model.

Decision Variable	Definition
p_i	berthing position of vessel i
t_i	berthing time of vessel i
q_i	departure time of vessel i
y_{ij}	number of cranes assigned to vessel i in period j

that the number of cranes is adjusted in that time period. The last time period that any QC is assigned to a vessel, i.e. the x coordinate of the right end of the rectangle gives the departure time q_i. In this example, $V = 4, T = 26, B = 16, N = 7$. Thus, it can be concluded that vessel i occupies berth segments $[p_i, p_i + 1, \ldots, p_i + l_i - 1]$ for the time interval $[t_i, \ldots, q_i]$. Consider the feasible solution in Figure 3.1 where overlapping of rectangles is prohibited. The berthing times are $t_1 = 2, t_2 = 2, t_3 = 4, t_4 = 9$; berthing positions are $p_1 = 11, p_2 = 7, p_3 = 4, p_4 = 7$ and completion times are $q_1 = 8, q_2 = 5, q_3 = 13, q_4 = 14$. We can also read quay crane allocations, for the first vessel: $y_{12} = y_{13} = y_{14} = 3,\ y_{15} = y_{16} = y_{17} = 4$; for the second vessel: $y_{22} = y_{23} = 3,\ y_{24} = 2,\ y_{25} = 1$; for the third vessel: $y_{34} = y_{35} = ... = y_{3,13} = 2$; and lastly for the fourth vessel: $y_{49} = y_{4,10} = 4,\ y_{4,11} = y_{4,12} = 3,\ y_{4,13} = y_{4,14} = 4$.

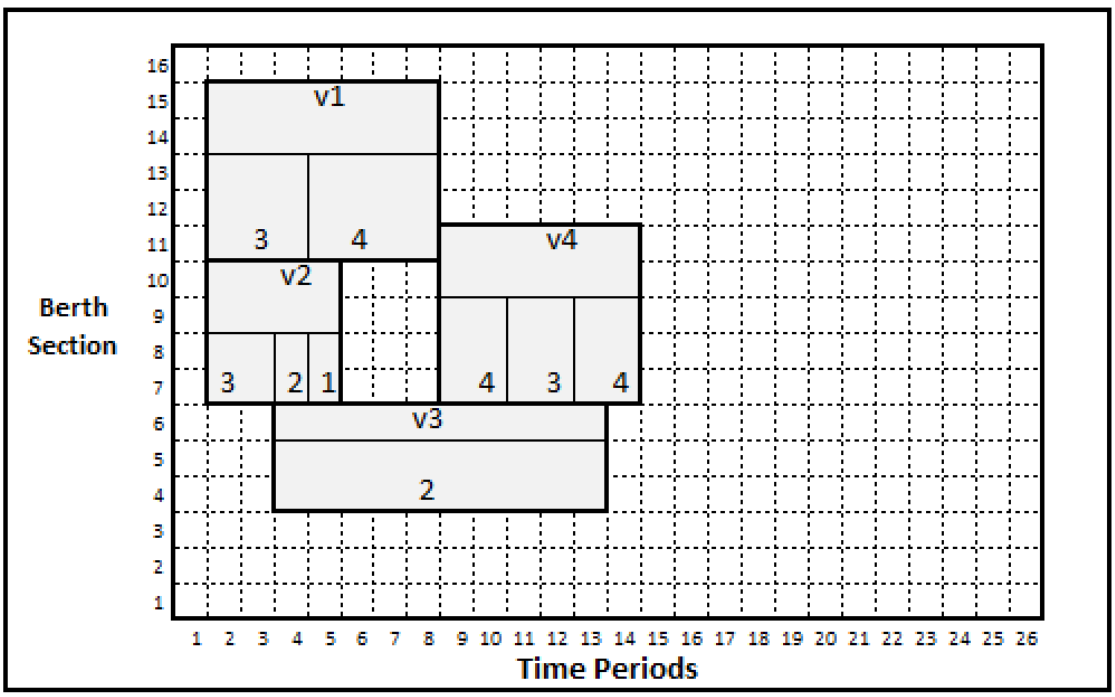

Figure 3.1. A Feasible Solution.

The objective of BACAP is to minimize the total cost emerging from berth and crane schedule. The cost structure and the objective function are elaborated in the next section.

3.2. Cost Structure

The vessels arrive at the port and they berth whenever an appropriate berth section and necessary quay cranes are available. If it cannot berth upon the arrival, then the vessel has to wait until the conditions are fulfilled for service provision. Waiting before service is not welcomed by the vessel operators and cause dissatisfaction of terminal customers. Besides, vessels are mostly used in international transshipments and they plan to visit at many ports throughout their voyage. Vessel operators usually establish a due time to depart from each harbor to catch up their pre-determined schedule. Therefore, departure after due date is another source of dissatisfaction for transshipment companies. Considering the facts mentioned above, waiting times before berthing and departure after due time are penalized and counted as cost items.

Containers unloaded from a vessel are transferred to storage areas by yard trucks and they are stored with the help of yard cranes. Similarly, the containers in storage area follow the reverse path to be loaded to a vessel at the berth. Desired berthing position of a vessel is specified by selecting the nearest berthing segment to the area where its containers are/will be kept. Provided that the vessel berths at a farther position, trucks have to travel a longer way, and the interior traffic created by many trucks slows down the loading/unloading operations. In the BACAP model proposed in this study, the trucks are assumed to be idle whenever needed so that the handling times are not affected by the distance to the storage areas. However the deviation from desired berthing position is penalized in the objective function.

Another component of the cost structure is the setup costs of quay cranes. As mentioned above, cranes slide over a rail track and setup of a crane requires moving to the proper berth section. This movement probably pushes the other cranes to move since crossing of cranes is not allowed. In BACAP, we are only interested in the number of cranes so whenever the number of QCs allocated to a vessel varies, a setup cost is incurred. The definition of cost coefficients and their brief explanations can be found in Table 3.3.

Table 3.3. Cost coefficients in the objective function.

Parameter	Definition
c_{i1}	Cost of one unit deviation from the desired berthing section for vessel i
c_{i2}	Cost of berthing one period later than the arrival time for vessel i
c_{i3}	Cost of departing one period later than the due time for vessel i
c_{i4}	Cost of changing the number of assigned cranes by one unit for vessel i

To sum up, all the terms contributing to the total cost constitutes the following objective function:

$$\begin{aligned} \min z = & \sum_{i=1}^{V} c_{i1} \left| p_i - s_i \right| + \sum_{i=1}^{V} c_{i2} \max(t_i - e_i, 0) \\ & + \sum_{i=1}^{V} c_{i3} \max(q_i - d_i, 0) + \sum_{i=1}^{V} \sum_{j=2}^{T} c_{i4} \left| y_{ij} - y_{i,j-1} \right| \end{aligned} \tag{3.1}$$

Suppose that the parameter list in Table 3.4 is given for the feasible solution in Figure 3.1. Then the resulting objective value is calculated below:

- The total cost of deviation from the desired berthing position:

$$10 \times |11 - 10| + 10 \times |7 - 7| + 10 \times |4 - 7| + 10 \times |7 - 8| = 40.$$

- The total cost of waiting before berthing:

$$10 \times \max(0, 2-2) + 10 \times \max(0, 2-2) + 10 \times \max(0, 4-4) + 10 \times \max(0, 9-7) = 20.$$

Table 3.4. A sample.

Parameter	1	2	3	4
$\underline{k}^i$	3	1	2	3
$\overline{k}^i$	4	3	2	4
l_i	5	4	3	5
e_i	2	2	4	7
s_i	10	7	7	8
d_i	10	8	18	16
w_i	25	9	20	22
c_{i1}	10	10	10	10
c_{i2}	10	10	10	10
c_{i3}	10	10	10	10
c_{i4}	1	1	1	1

- The total cost of departing after due time:

$$10\times\max(0,8-10)+10\times\max(0,5-8)+10\times\max(0,13-18)+10\times\max(0,14-16)=0.$$

- The total setup cost of QCs:

$$(3+1+4)+(3+1+1+1)+(2+2)+(4+1+1+4)=28.$$

The overall cost of the given solution in Figure 3.1 is $z=50+20+0+28=98$.

3.3. Solution Procedure: A Tabu Search Metaheuristic

A metaheuristic is a solution method which combines the tools of local improvement processes with intelligent search strategies in order to avoid get stuck in local optima. In complex solution spaces, local improvement procedures like ascent, descent

methods may result in bad solutions in terms of proximity to the optimum. A process, which helps move from one solution to another utilizing a well-defined neighborhood structure, is coupled with the classical local search algorithms in metaheuristics so that the search can be done in a wide area while avoiding the risk of trapping in local optima.

The metaheuristic term is first used in (Glover, 1986) where *tabu search* algorithm is also introduced. Tabu search metaheuristic, which is elaborated, later on (Glover and Laguna, 1997), exploits local search methods with a short-term memory in the simplest version. A tabu list is kept for a number of iterations to avoid cycling back to previously visited solutions. As an additional feature, a long-term memory is also employed to detect more promising areas in the search space.

The BAP with dynamic arrival times is proved to be NP-hard in (Hansen and Oğuz, 2006), (Lim, 1998) and (Imai *et al.*, 2005). Also, MILP formulations are constructed for BAP or BACAP in a number of papers and most of them can only be solved for a few vessels which are of no practical use. A recent work, (Türkoğulları *et al.*, 2012a), developed a BACAP model and succeeded to reach optimality up to 60 vessels. Since exact algorithms require longer times, different metaheuristics are used such as genetic algorithm, tabu search, simulated annealing or hybrid methods to find near optimal values in shorter amount of times for these problems. Particularly, (Cordeau *et al.*, 2005) proposes a tabu search for BAP while (Meisel and Bierwirth, 2009) and (Ak, 2008) modifies the tabu search for BACAP.

The tabu search (TS) algorithm presented in this study is similar to the work of (Ak, 2008) in terms of TS structure. The neighborhood search is done in two nested layers. At the outer layer, the algorithm switches from one vessel priority list to another by swap operations. While the priority list is kept unchanged, the inner layer searches a binary vector which holds two different rules for placing the corresponding vessel. These two arrays, priority list and rule vector, represent a feasible solution to the problem.

3.3.1. Solution Representation and Generation

Each feasible solution is encoded by a vector pair of size V. The first vector is the priority list P which is a sequence of vessels, the other one is the binary-valued rule list R. Suppose that a solution is represented by x. Then, (P_x, R_x) pair generates a unique feasible solution according to the following instructions: Pick vessels from P beginning with the first vessel. Let i be the vessel with k^{th} highest priority i.e. $P(k) = i$. Check the rule list R. $R(i) = 0$ means that the vessel i prefers berthing as soon as a berth segment of length equal to l_i becomes available after its arrival e_i. Finding available berth section is not sufficient. The rectangle corresponding to vessel i must not overlap with the other $k-1$ pre-scheduled rectangles in order to maintain feasibility. So, the selected berth sections should be unoccupied by another vessel throughout the time the vessel i stays at the berth. The service duration cannot be exactly known without allocating quay cranes but an upper bound can be calculated assuming that the vessel is assigned $\underline{k}^i$ many QCs in every time period. Denote the number of idle QCs in period j by AC_j. A better upper bound can be found by assigning the $\min(\underline{k}^i, \max(AC_j, \underline{k}^i))$ for each time period but additional computation is necessary. The earliest time after e_i when vessel i can berth without overlapping is determined as the berthing time t_i. The berthing position satisfying these conditions at t_i are listed as candidate berthing positions. If there are multiple candidates for berthing, the berth segment with the smallest distance from the desired berthing position of vessel, s_i, is chosen as the berthing position p_i.

Although berthing at the earliest time seems rational, this kind of First-Fit replacement is not always intelligent enough to find the optimal schedule. At optimality, some vessels might prefer to wait until a nearer berthing position to s_i becomes available instead of berthing immediately. So, an alternative rule is implemented which is denoted by 1 in R vector. $R(i) = 1$ means that the vessel prefers to berth obeying the feasibility conditions so that the berthing position p_i and time t_i minimizes its *individual* cost of waiting time and deviation from s_i. This process may seem unpractical since it requires many comparisons. However, this challenge can be handled by

finding the earliest time, say t_i^{temp}, that vessel i can berth at s_i without overlapping. Checking the time periods after t_i^{temp} is meaningless because we cannot find a better berthing position than s_i. So our search is confined to the berthing positions in the time interval: $[e_i, t_i^{temp}]$.

Following the determination of p_i and t_i, QC allocation must be done. Then for each time period j, assign $\min(AC_j, \overline{k}^i)$ cranes to vessel i as long as it stays at the berth. If $\min(AC_j, \overline{k}^i)$ is less than $\underline{k}^i$, then the difference $\underline{k}^i$- $\min(AC_j, \overline{k}^i)$ is transferred from another vessel l which is being served at time j and uses more than $\underline{k}^l$ QCs. If there are more than one vessel using more QCs than its minimum number of needed cranes, pick the vessel with the highest number of QC allocation in time period j. If the difference $\underline{k}^i$- $\min(AC_j, \overline{k}^i)$ is more than one, cranes are taken one by one from the vessel with the highest number of QCs in each turn. While transferring cranes, the completion time of vessel l can be changed since its QC usage is decreased in period j. The vessel l is not allowed to give its cranes to vessel i if the new completion time, q_l exceeds its due date d_l. Non-overlapping of rectangles must be maintained, lower and upper bounds for QC usage must be respected during this step. The last period that the vessel i gets service, is assigned as the completion time, q_i, of vessel i.

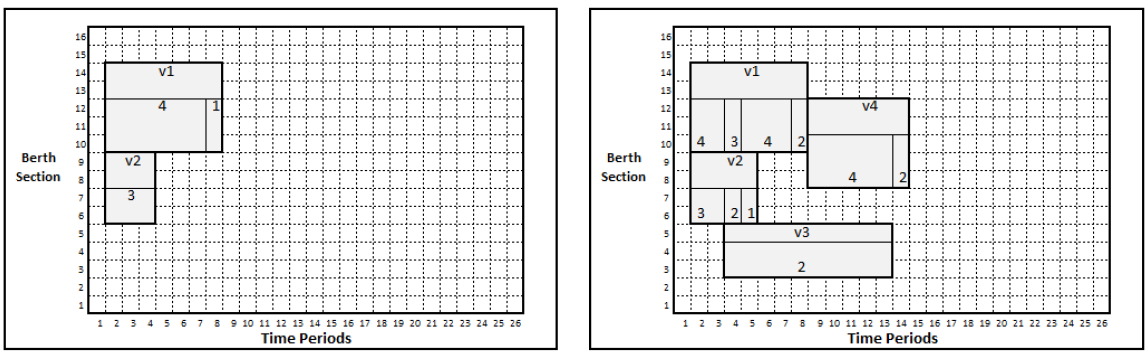

Figure 3.2. Solution Obtained from $P = \{1, 2, 3, 4\}$ and $R = \{0, 0, 0, 0\}$.

Consider the instance given in Table 3.4 with $P = \{1, 2, 3, 4\}$ and $R = \{0, 0, 0, 0\}$ where $V = 4, B = 16, T = 26, N = 7$. Vessel 1 can berth as soon as it arrives at its favorite berthing position regardless of the R list. Since the R list is composed of 0's

in this example, each berth prefers to berth upon the arrival. Vessel 1 berths in time period 2 at its desired berth section 10. At most, 4 craness can be allocated to vessel 1. Except for the last period, 4 QCs are assigned. Although the minimum number of cranes is 2, only one crane is given in the last time period. The reason is that there remains only1 QC-hour workload to be handled. Since the service of vessel 1 is finished in time period 8, the completion time is determined to be 8. Vessel 2 berths in a similar fashion, however it berths 1 berth section away from s_2 to avoid overlapping as can be seen in Figure 3.2. At the arrival time of vessel 3, berth sections are available but there is no idle crane. According to the solution decoding given above, vessel 3 can berth and some cranes can be transferred to vessel 3 from vessels 1 and 2 in time period 4. The assigned QCs to vessel 3 after transfer is *equal* to the minimum number of QCs which vessel 3 needs. Also, vessels 1 and 2, which give their cranes out, must update their crane distribuiton and renew their completion times if necessary. Vessel 4 has to wait two time periods before berthing since there is no available berth segment at e_4 fitting vessel 4. Now, the total objective value can be calculated as 96.

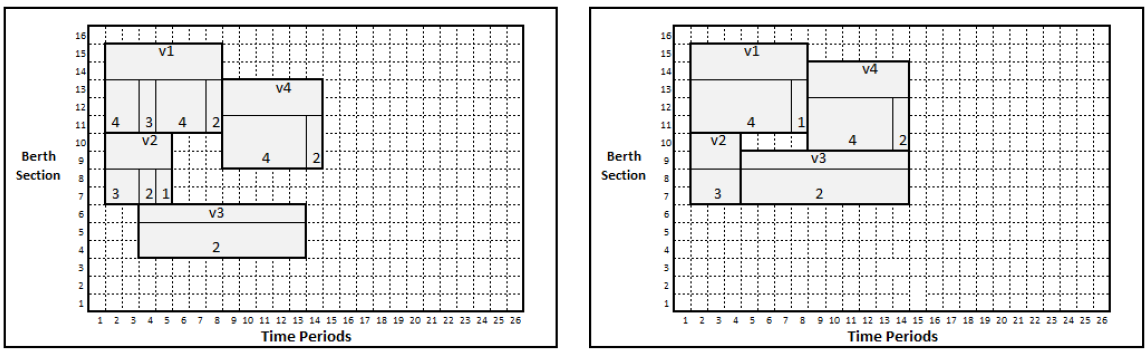

Figure 3.3. Solution Obtained from $P_1 = \{2, 1, 3, 4\}$ and $R_1 = \{0, 0, 0, 0\}$ and $P_2 = \{2, 1, 3, 4\}$ and $R_2 = \{0, 0, 1, 1\}$.

A neighboring list to $P_1 = \{2, 1, 3, 4\}$ can be obtained by swapping the vessels 1 and 2 in P. Keeping the rule list $R_1 = \{0, 0, 0, 0\}$ the same, the solution shown in the left chart of Figure 3.3 can be obtained from these two lists and the resulting objective value is computed as 86. As mentioned earlier, a neighboring solution can either be obtained modifying P or R. By changing some zero values to one in R_1, the

new rule list $R_2 = \{0, 0, 1, 1\}$ is constructed. $P_2 = \{2, 1, 3, 4\}$ is the same vector as P_1. The objective value of the corresponding solution of (P_2, R_2) pair is found as 76. The resulting solution can be visualized in the right chart of Figure 3.3.

3.3.2. Initial Solution

A feasible solution can be obtained by applying the solution generation procedure described above from a pair of priority and rule list (P, R). This initial solution is denoted by x_0 and the vectors producing x_0 are called P_0 and R_0, respectively. Our candidates for P_0 vector are produced as follows:

- The priority list ordered according to the increased arrival times i.e. the list constructed using First Come First Served (FCFS) rule: P_{arr}
- The priority list ordered according to the increased due times i.e. the list constructed using Earliest Due Date (EDD) rule: P_{due}
- The priority list ordered randomly: P_{rand}

The first two candidates are also proposed in (Ak, 2008), since they can be constructed intuitively by any terminal operator. For most heuristic approaches, beginning from a sufficiently good solution improves the quality of the final result and speed of the algorithm. So, these two vectors are expected to produce satisfactorily good initial solutions due to the fact that waiting times and tardiness are both minimized in the objective function.

The third candidate is a randomly ordered priority list. However, the vessels are not selected by a pure random process. Suppose K is a positive integer less than the number of vessels, V. P_{arr} is taken and divided into $\lceil V/K \rceil$ groups such that, first K vessels of P_{arr} is in the first group, second K vessels of P_{arr} is in the second group and so on and the remaining vessels of P_{arr} are in the last group. Each group is randomly ordered within itself and new randomized groups are concatenated respecting the initial order of groups in the P_{arr} list. This kind of randomization prevents giving very high

priorities to late arriving vessels and very low priorities to early arriving vessel and leads to better quality solutions than the pure random process does.

R_0 is initially set to zero vector of length V. The pair producing the solution with the smallest objective function among (P_{arr}, R_0), (P_{due}, R_0) and (P_{rand}, R_0) is assigned as the initial priority and rule vectors (P_0, R_0) and the corresponding solution is set as the initial solution x_0.

3.3.3. Outer Search

Neighborhood Structure: Suppose that the current priority list is denoted by P_c and the current solution produced by (P_c, R_0) is labeled as x_c. The priority and rule list of the best solution found, x_{best}, is denoted by (P_{best}, R_{best}). At the beginning of the algorithm $(P_{best}, R_{best}) = (P_0, R_0)$ and $P_c = P_0$ .Here, the same neighborhood structure provided in (Ak, 2008) is employed as follows: Choose a vessel k among m selected vessels from P_c. Find the r closest vessels to vessel k, where closeness is defined in terms of the absolute difference of indices in P_c. Pick a vessel, say vessel l, among the n randomly selected vessels of r closest ones to vessel k. Swap vessels k and l, and name the new neighboring priority list as P_n. Produce n_1 many neighboring lists of P_c and if (k, l) swap move is not in the tabu list of the outer search (TL_1), run the inner search procedure for each of the neighboring priority lists. The inner search will return a rule list R_n for each of P_n. Now, the pair (P_n, R_n) is a neighboring pair for outer search (see Figure 3.4). The best (smallest) cost neighbor (P_n^*, R_n^*) is labeled as (k^*, l^*).

Solution Update: The best valued neighboring solution is set as the current solution: $x_c = x_n^*$ and $P_c = P_n^*$. If the solution, x_n^*, of $(P_n^*,\ R_n^*)$ has a better objective value than the best solution, then the latter is updated: $x_{best} = x_n^*$ and $(P_{best}, R_{best}) = (P_n^*, R_n^*)$.The swap move (k^*, l^*) is added into the tabu list of the outer search and delete the moves whose tabu tenure is passed are deleted. Tabu duration of the outer search, (TT_1), is dynamically determined at each iteration. It is distributed with a

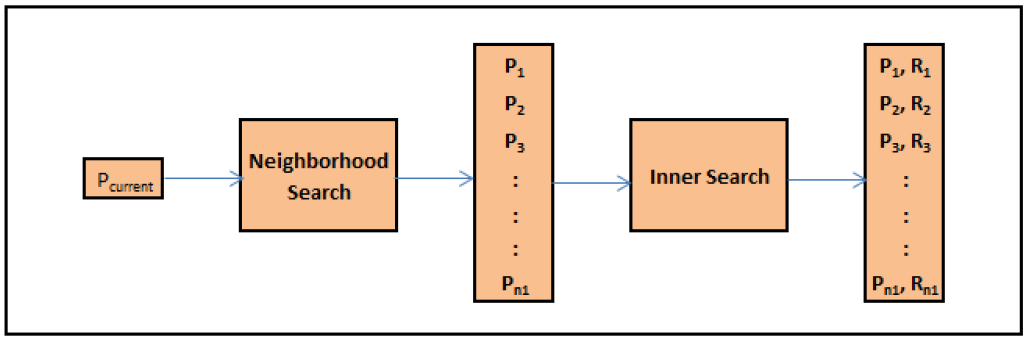

Figure 3.4. Neighboring Solution Generation of Outer Search.

uniform distribution on the interval (a_1, b_1).

Termination Criterion: The outer search is stopped if the best solution is not improved for a pre-determined number (C_1) of successive iterations.

3.3.4. Inner Search

Neighborhood Structure: A priority list, P, is input to the inner search. Keeping P unchanged, we aim to find a rule vector which gives the best possible solution when combined with P. The initial rule vector for the inner search, R_0, is a vector of zeroes. Initially, (P, R_0) pair is set to (P, R_c) and (P, R_{best}). Also, u_0 which is the initial solution of inner search is set to u_c and u_{best}. Since P is not changed during inner search, we can store the best and current solutions of inner search by keeping the necessary rule vectors. A neighboring rule vector of R_c is obtained by changing the rule of *only one vessel* from 1 to 0 or from 0 to 1. So there can be at most V neighboring rule list at one iteration of the inner search i.e $n_2 \leq V$. The neighboring rule vector R_f is obtained by altering the value of the f^{th} vessel, i.e, the f^{th} index of R_c. The objective value of solution related to pair (P, R_f) is computed. If move f is not in the tabu list of the inner search, TL_2, or if f is in the TL_2 but (P, R_f) produces a better objective value than u_{best} does, then R_f is kept as candidate for new R_c in the next iteration. The smallest cost neighboring solution (P, R_f^*) is labeled as f^*.

Solution Update: The best valued neighboring solution is set to current solution: $u_c = u_f^*$ and $R_c = R_f^*$. If the solution, u_f^*, of (P, R_f^*) has a better objective value than the best solution, then the latter is updated: $u_{best} = u_f^*$ and $(P, R_{best}) = (P, R_f^*)$. The move f^* is added into the tabu list of the inner search and the moves whose tabu tenure is passed are deleted. Tabu duration of the inner search, (TT_2), is dynamically determined at each iteration. It is distributed with a uniform distribution on the interval (a_2, b_2).

Termination Criterion: The inner search is stopped if the best solution is not improved for a pre-determined number (C_2) of successive iterations. Then R_n which is equal to R_{best} is returned as the result of this procedure. The pair (P, R_n) generates a neighboring solution in the outer search.

3.4. Local Improvement Algorithms

According to the solution generation approach described in Section 3.3.1., the vessels with high priorities are capable of both berthing in the earliest possble time and gain as much QCs as possible. However, the low priority-vessels are contented with a feasible berthing position and they can only be assigned the remaining QCs. In short, priority of vessels determine both berth assignment decisions and quay crane allocation decisions. The optimal solution of BACAP might be a solution which can never be obtained by this solution generation method. For instance, at optimality, there may be some vessels with low priority in berthing decisions but high priority in QC allocation decisions. Moreover, no control mechanism is presented up to now, minimizing the setup cost of cranes. These shortcomings are aimed to be repaired by local improvement algorithms presented in this section.

3.4.1. Smoothing Quay Crane Utilization

Smoothing or leveling the QC usage is a necessary procedure to decrease the setup cost incurred by QCs. Smoothing operation can be done as follows:

- If QCs of a vessel are adjusted, i.e., the number of assigned QCs are changed many times, the setup cost clearly increases. Instead, QCs may be distributed homogeneously, without changing the completion time of a vessel. New QC distribution must obey the availability of cranes.
- Suppose the service of a vessel is completed by assigning at most v QCs in a time period. This maximum number can be reduced to, say $v-1$ or less by postponing the completion time. The new completion time can only be extended until coming across to another vessel or its own due time.

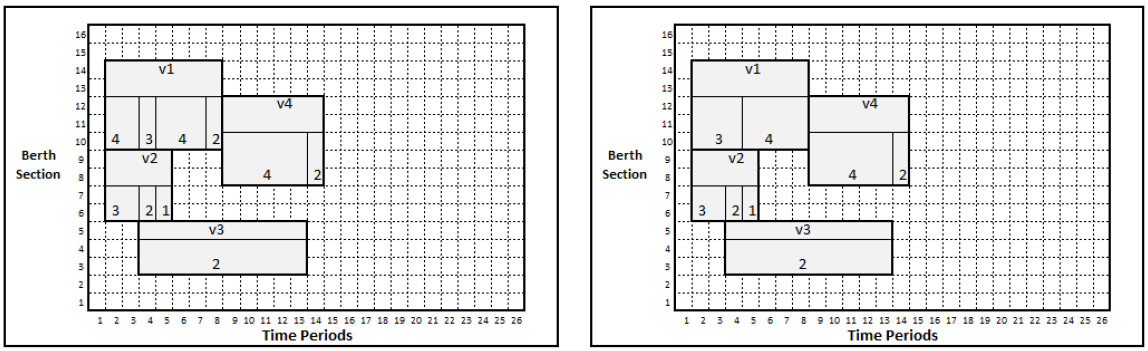

Figure 3.5. Quay Crane Smoothing without Changing Completion Times.

The first case can be visualized in Figure 3.5. The first vessel has many QC adjustments and the setup cost of vessel 1 calculated as: $4+1+1+2+2=10$. After smoothing, the vessel's QC usage is distributed over the same time periods minimizing the number of adjustments. The new setup cost of vessel 1 is computed as: $3+1+4=8$.

The second case is materialized in Figure 3.6. Consider the left chart of Figure 3.6. The completion time of the first vessel cannot be changed due to the fact that vessel 4 occupies some of the berth sections of vessel 1 as soon as vessel 1 departs. Also, no smoothing can be done on vessel 3 since the QC utilization is at its minimum level (2 QCs) for that vessel. On the other hand, maximum QC allocation is four for vessel 4 and 3 for vessel 2. The setup cost of vessel 2 is $3+1+1+1=6$ and that of vessel 4 is $4+2+2=8$. We can lower the QC usage per period and the completion times of these vessels can be postponed. Recall that postponing can be done until due times as

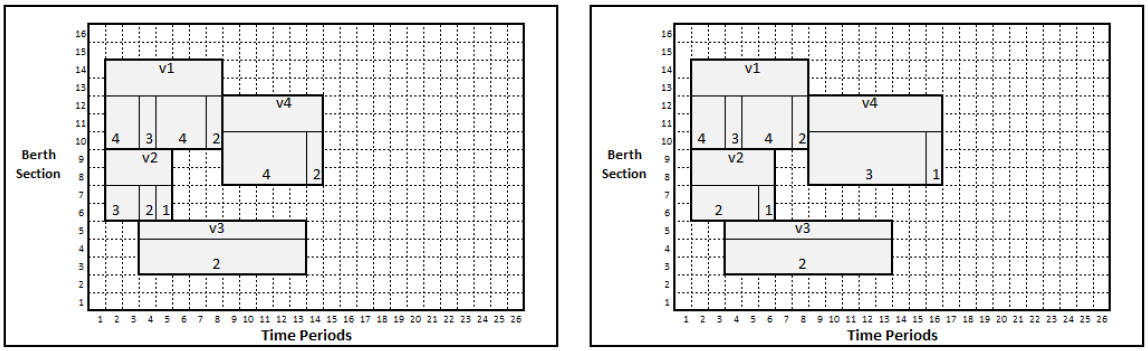

Figure 3.6. Quay Crane Smoothing Changing Completion Times.

long as the minimum number of QC requirements are respected and overlapping does not occur. All feasibility conditions are satisfied in the new berth plan and the new setup costs for vessel 2 is $2+1+1=4$ and that for vessel 4 is $3+2+1=6$ as shown in the right chart.

Given a solution obtained from priority list P, the complete smoothing procedure can be implemented beginning with the first vessel in P by following the steps below:

- *Step1:* Denote the maximum number of allocated QCs to vessel i as $v_i^{\max}$. Detect the time periods with $v_i^{\max}$ QC usage and if there are multiple periods label the earliest one as j. Beginning from the berthing time of vessel i, find periods that vessel i uses less than $v_i^{\max}-1$ cranes and label the earliest one as h. If QCs are not fully utilized in period h, decrease the QC consumption at j by one unit and increase the QC usage at h by one unit. Update $v_i^{\max}$ and repeat this process until maximum and minimum values of QC usage get close enough. If no time period with label h is found go to the step 2.
- *Step2:* Check whether any of the berth sections occupied by vessel i is assigned to another vessel just after i departs or whether $q_i \geq d_i$. If so, end this step. If not, detect the time period with $v_i^{\max}$ QC usage and if there are multiple periods label the earliest one as j. If the QC usage in q_i is less than $v_i^{\max}$ and there are idle cranes in period q_i, decrease the QC consumption at j by one unit and increase

the QC usage at q_i by one unit. If QC usage in q_i is not less than $\underline{k}^i$ and there are unassigned QCs in period $q_i + 1$, decrease the QC consumption in j by one unit and increase the QC usage at $q_i + 1$ by one unit and update the completion time as $q_i = q_i + 1$. Repeat this step until the conditions are met to end this step or the QC usage of vessel i reaches at its minimum level. Remember that allocated QC number cannot be less than $\underline{k}^i$ in any time period except for q_i and the total crane usage of all vessels cannot exceed N in a time period.

In (Ak, 2008), another layer, which makes tabu search iterations to find the best QC allocation, is added to the model. However, adding another layer to this model will lead to longer computational times and make the heuristic inefficient. We already know that setup cost of cranes is much less compared to berth dependent costs, so leveling the QC utilization with an intuitive approach is expected to produce near optimal results in shorter times. A similar QC leveling procedure can be found in (Meisel and Bierwirth, 2009) which assigns limited number of cranes to each vessel initially and distribute the remaining cranes to vessels following a priority list.

3.4.2. Spatial Improvement Procedure

In our solution generation approach, the vessels with higher priority are more likely to berth to their berthing positions while the lower priority ones berth at the possible best berthing positions. At the end, the difference of individual costs of vessels may differ a lot. However, vessels with nearly equal costs may produce better results than the greedily replaced vessels. Spatial improvement procedure only modifies the berthing positions of vessels without changing berthing and completion times or QC allocation.

Consider the instance given in Table 3.5. Assume that $c_{i1} = c_{i2} = c_{i3} = 10$ and $c_{i4} = 0$. Also, $V = 3, B = 16, T = 26, N = 7$. Trying different (P, R) pairs we can obtain different solutions for BACAP (see Figure 3.7). In part A, $P = \{1, 2, 3\}$ is given and the best rule list, which gives the smallest objective value when combined with

Table 3.5. A sample instance with three vessels.

Parameter	1	2	3
$\underline{k}^i$	2	2	2
$\overline{k}^i$	4	3	4
l_i	5	4	5
e_i	2	2	4
s_i	10	9	7
d_i	23	24	25
w_i	23	40	30

$P_A = \{1, 2, 3\}$, is found as $R_A = \{0, 0, 0\}$. For this case whatever the rule list is, the same solution is obtained. As a matter of fact, even totally different (P, R) pairs may generate the same solution in some situations. The objective value of this berth plan is found to be 90. The solution in part B is obtained from the pair $P_B = \{1, 3, 2\}$ and $R_B = \{0, 1, 0\}$. Here, only the second vessel berths according to the rule denoted by 1. R_B is selected in such a way that the rule vector produces the best solution when combined with P_B. The solutions in other parts are also generated similarly. Although, all possible candidates are tried according to our solution decoding method, the optimal solution is not met for this tiny instance. Actually whatever the problem size is, such a challenge can always be the case if more than two large vessels prefer berthing at very near sections in almost the same time periods.

The optimal solution can be generated by shifting three vessels in part A spatially upwards. The new objective value is found to be 70 and the new solution is presented in Figure 3.8. In this solution, none of the vessels individually minimizes its cost but the total cost is minimized. Since none of the vessels has higher priority than the others in terms of berthing position selection, no priority list can produce this solution. Given a solution, spatial improvement procedure can be outlined as follows:

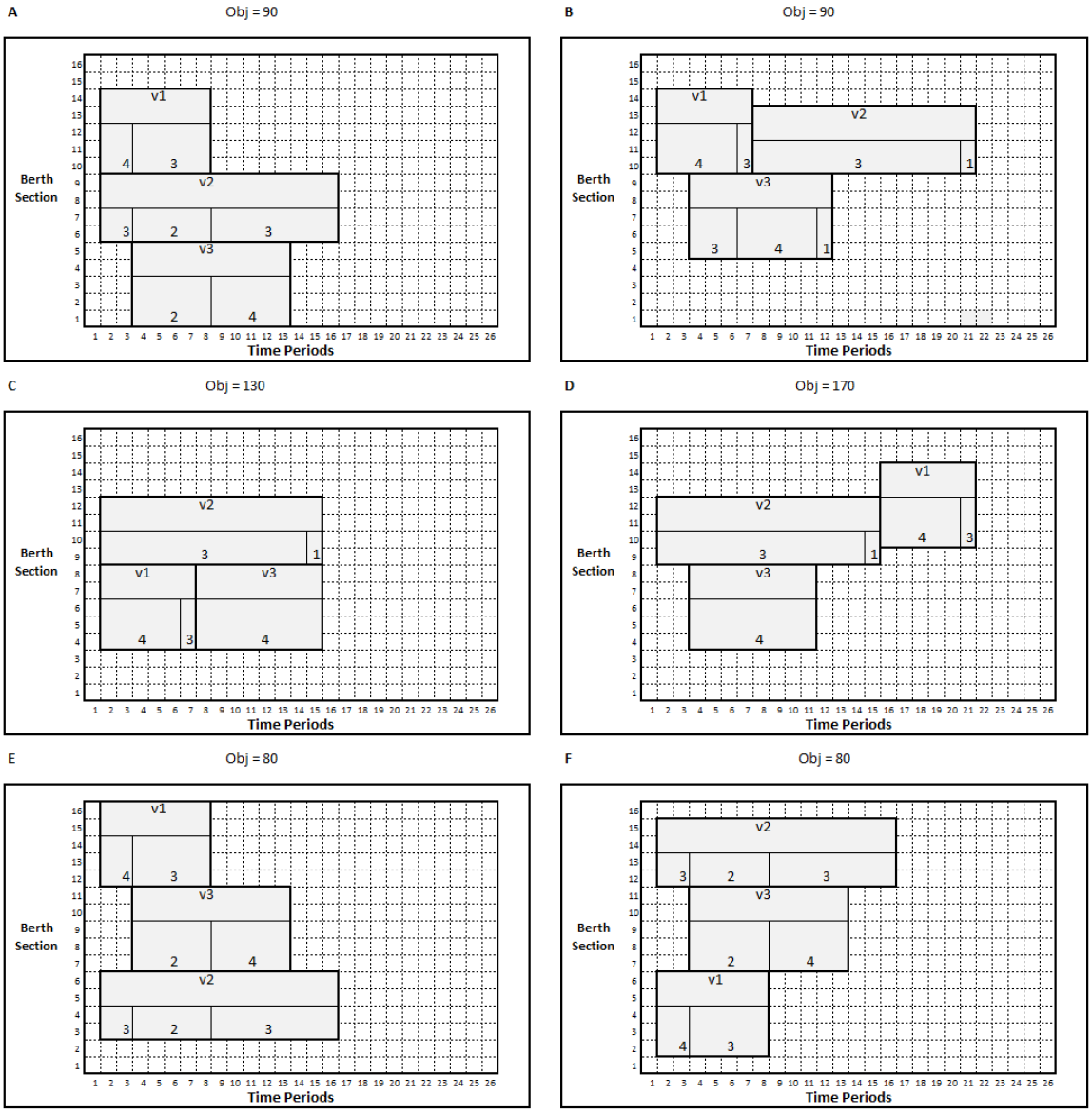

Figure 3.7. Different Solutions for Different (P, R) Pairs.

- Find the clusters consisting of spatially adjacent vessels in the time-space diagram. For example, in Figure 3.7, all three vessels in parts A, B, C, E and F constitute a cluster. Except that, in Part D, only vessels 2 and 3 constitute a cluster because vessel 1 is not adjacent to any vessel. It can move in the vertical direction freely. If the clusters contain at least 3 vessels, move this cluster up and down to see if there is any improvement in the objective value. We eliminate the clusters with 2 vessels because if only 2 vessels have conflicting desired berth segments, the best solution can be found by simply altering the priorities in the tabu search.

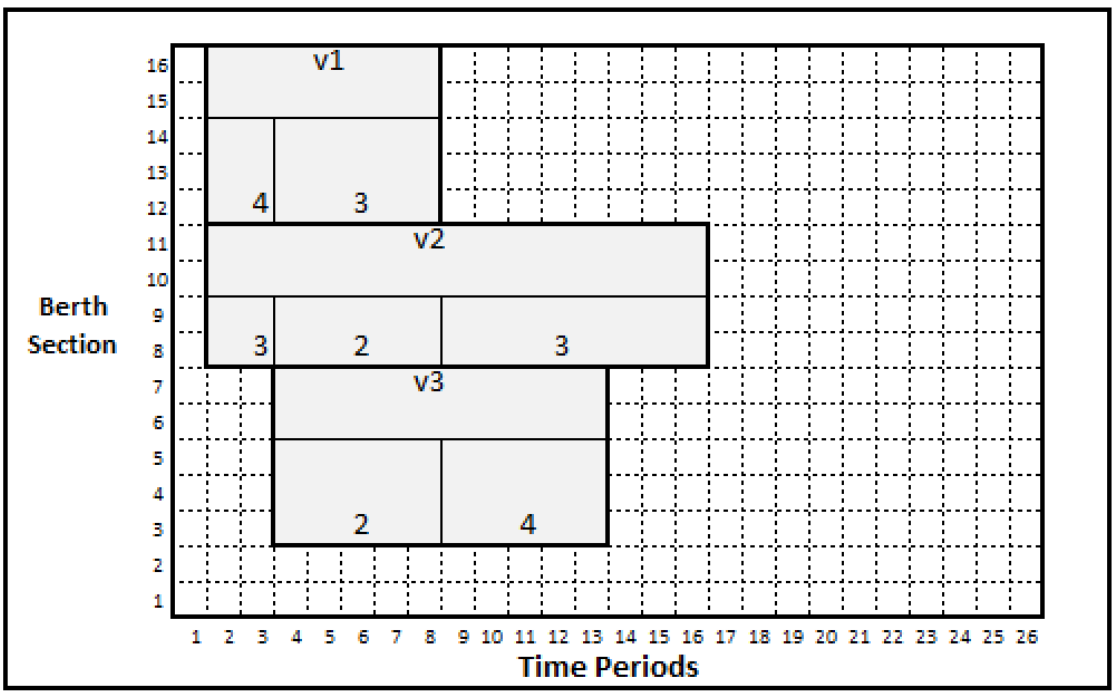

Figure 3.8. Optimal Solution After Spatial Improvement.

3.4.3. Temporal Improvement Procedure

3.4.3.1. Checking Berthing Times. The priority list P determines the priorities of vessels both for berthing decisions and QC allocations simultaneously. Suppose that two vessels are served during the same time periods and the first vessel can choose its

desired berth segment while the second one has to berth at the best available berth section. Since the first vessel has higher priority, it would get the highest possible number of QCs while the second only can get the remaining or minimum necessary QCs for itself. What if the berth segments of the second vessel are also desired for other vessels and the berth sections of the first vessel are less popular? Then, a better solution is to assign the maximum number of QCs to the second vessel. However, such a solution cannot be generated in the tabu search algorithm. Some local refinements are necessary.

For the purpose of reaching a better solution, berthing times of every vessel located in the time-space diagram is checked. If $t_i > e_i$ for any vessel i, then the vessel(s) which prevent vessel i to berth earlier is (are) detected and put in the set V_s^i. If the QC allocation of vessels in V_s^i are not at its maximum level for any period j, some QCs are transferred to vessels in V_s from other vessels (the set S_s^i) which are served simultaneously with V_s^i. First it should be controlled whether the completion times of vessels in S_s^i can be put off or not. If the completion times can be postponed, we can transfer some vessels from S_s^i to V_s^i respecting minimum-maximum restrictions of QC assignment. As a result, the service of vessels in V_s^i is finished earlier while the ones in S_s^i depart later when compared to the beginning. If V_s^i is finished earlier, then vessel i can berth in an earlier time period, too.

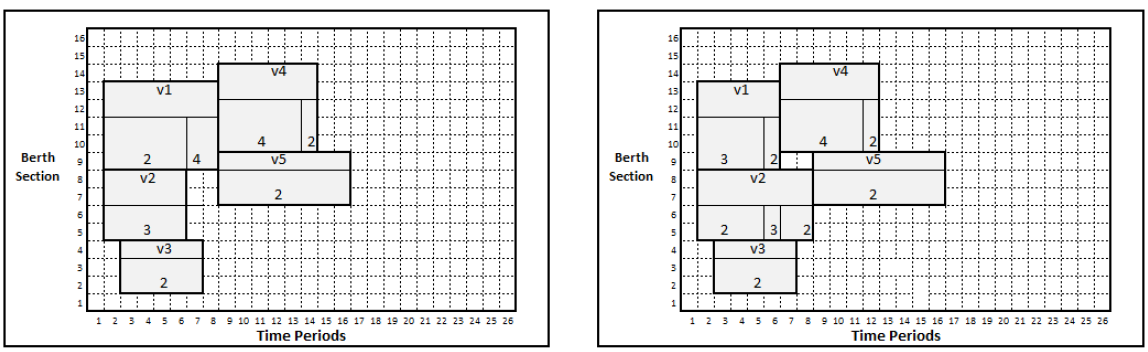

Figure 3.9. Checking and Improving Berthing Times.

The described procedure is implemented in Figure 3.9. Consider the vessel 4

shown in the left berth plan. It berths in time period 9. However, the arrival time of vessel 4 is given as 7. In this case, the vessel 4 has to wait because the vessel 1 is still being served at the arrival of vessel 4. So we can check whether the service of vessel 1 can be finished earlier. We can increase the number of allocated QCs to vessel 1 and decrease the number of QCs assigned to vessel 2. Vessel 2 can give out its QCs because postponing the completion of vessel 2 does not lead to overlapping or overdue. After updating berthing times, the new solution will clearly have a smaller objective function value.

3.4.3.2. Checking Departure Times. In some solutions, the vessels with low priorities may be assigned very early due times, i.e, the time interval between the arrival and due times may be too short. To recover the costs incurred by such a situation, checking whether the departure times of vessels exceed the due times is helpful. If a vessel departs after its due time, then the number of QCs allocated is increased by transferring from other concurrently served vessels as described in the previous section.

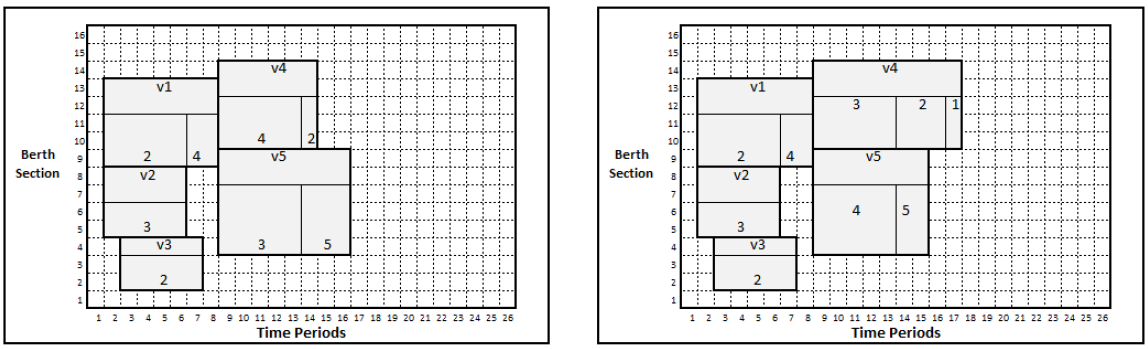

Figure 3.10. Checking and Improving Departure Times.

Consider the solution in the left chart of Figure 3.10. Vessel 5 departs in time period 16, whereas the due time parameter of vessel 5 is determined as 15. The service of vessel 5 could not be finished earlier because the most of the QCs was utilized for vessel 1, which is also at the berth. The overdue of the vessel 5 can be recovered by assigning more QCs to it. Since there is no idle crane, the number of QCs assigned to vessel 1 is decreased by one unit between periods 9 and 13. Vessel 1 can give out

its QCs because we can put off the departure time of vessel 1. We would not put off the completion time of vessel 1, if there is any possibility that vessel 1 departs after its due time or there occurs an overlapping with another vessel in the time-space diagram. The new solution, which is presented on the right chart of Figure 3.10 shows that vessel 5 now departs in period 15 without any overdue and hence with a lower total cost.

In conclusion, temporal shifts of vessels only adjust the berthing or completion times and never modify the berthing positions. Changes on QC allocation is inevitable since temporal improvement can only be done by playing with QC distribution. A strong assumption which guarantees that the temporal improvement processes produce better objective values is that costs related to berth assignment are much important than QC setup costs.

3.4.4. Integration of Local Improvement Procedures to Tabu Search

The local refinement policies described above is applied to every solution generated from any (P, R) pair in the following order:

- A solution is obtained from a (P, R) pair and p_i, t_i, q_i and y_{ij} are found for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- Berthing times are checked and improved and t_i, q_i and y_{ij} are updated for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- Completion times are checked and improved and t_i, q_i and y_{ij} are updated for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- Spatial improvement procedure is applied and p_i is updated for $i = 1, \ldots, V$.
- Assigned QCs are smoothed beginning with the first vessel in P. y_{ij} and q_i are updated for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.

3.5. Experimental Results for BACAP

3.5.1. Determination of Tabu Search Parameters

In heuristic approaches, the choice of parameters has a significant effect on the quality of the solution method. A parameter set is determined through a preliminary study. Different trials has proven that the parameters in Table 3.6 produce good quality solutions preserving the speed advantage of the heuristic.

Table 3.6. TS parameters.

Parameter	Value
Number of groups to produce P_{rand} (K)	5
Number of neighbors produced in outer search (n_1)	$(5V/6)$
Number of neighbors produced in inner search (n_2)	(V)
Tabu tenure of outer search $(TT_1 \sim (a_1, b_1))$	$\sim U(3, 5)$
Tabu tenure of inner search $(TT_2 \sim (a_2, b_2))$	$\sim U(3, 5)$
Max # of successive outer iterations without improvement (C_1)	5
Max # of successive inner iterations without improvement (C_2)	10

Except for the termination criterion of outer and inner iterations, all TS parameters is set either as a function of V or a random variable. K is selected as 5 so, P_{rand} will be generated after all vessels are divided into clusters of 5 vessels. n_1 is determined as a function of V and n_2 is assigned to its upper bound to balance the diversification and time usage.

3.5.2. Instance Generation

The tabu search algorithm with local improvement procedures (TS) is experimented on 106 different instances. It is initially applied to the instance with 21 vessels given in (Zhang *et al.*, 2010). Six more instances are derived from that instance by

taking the first 3, 6, 9, 12, 15, 18 vessels which arrive earlier than the others.

The remaining instances are produced under three different conditions to see whether TS is an efficient method under different assumptions. Firstly, interarrival times of vessels are taken as a random variable uniformly distributed over the interval $(0, 20)$. The instances in this first group is called as instances with *loose arrival times*. The second and the third set of instances are generated assuming that interarrivals are uniformly distributed over the intervals $(0, 15)$ and $(0, 10)$. These sets are named as instances with *moderate arrival times* and *tight arrival times* respectively. In all instances, l_i changes between 3 and 8, $\underline{k}^i$ is assigned to 2 for vessels of length 3, 4 and 5 and assigned to 3 for the remaining larger vessels. Considering the safety distances between QCs, $\overline{k}^i$ is determined as $l_i - 1$. w_i is considered as a function of l_i, since the container carrying capacity of the vessel is proportional to its size. w_i is uniformly distributed between $0.2l_i$ and $1.5l_i$. s_i is another random variable generated from $U(1, B - l_i)$. The due time, d_i, is found by $e_i + \lceil w_i/\overline{k}^i \rceil + U(5, 15)$. We want to produce a similar data set given in (Zhang *et al.*, 2010) to be able to reach parameter values which make sense in a real life application. Moreover, the same container terminal environment of (Zhang *et al.*, 2010) is used, i.e., $B = 24$ and $N = 12$. The instance sizes are fixed at 60 vessels, while generating the data. Data sets with less number of vessels are derived from them by taking first 10, 15, 20,. . . , 55 vessels.

TS algorithm is developed in a C# environment on Microsoft Visual Studio 2010. The results are compared with the optimal results obtained by the exact model proposed in (Türkoğulları *et al.*, 2012a) with the help of IBM ILOG CPLEX 11.0. All the experiments are performed on a computer with Microsoft Windows Server 2003 x64 Edition operating system and Intel Xeon CPU X5460 3.16 GHz processor with 27.9 GB RAM.

The program is run for two cases. In the first case the setup cost is ignored by letting $c_{i4} = 0$ and $c_{i4} = 10$ in the second case. The results of experiments which is run with the instance of (Zhang *et al.*, 2010) can be found in Tables 3.7 and 3.8.

Table 3.7. The instance given in (Zhang *et al.*, 2010) with $c_{i4} = 0$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	3	2,000	10.7	2,000	2,000	0.1	0	0.0
2	6	13,000	64.1	13,000	13,400	1.8	0	3.1
3	9	15,000	120.3	15,000	15,000	6.5	0	0.0
4	12	15,000	182.6	15,000	15,000	12.4	0	0.0
5	15	27,000	209.0	27,000	27,000	38.1	0	0.0
6	18	30,000	350.7	30,000	30,000	61.2	0	0.0
7	21	30,000	342.7	30,000	30,000	87	0	0.0
Avg		18,857	182.9	18,857	18,914	29.6	0	0.4

Table 3.8. Test results for data given in (Zhang *et al.*, 2010) with $c_{i4} = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	3	2,160	12.0	2,160	2,160	0.1	0.00	0.00
2	6	13,420	105.7	13,420	13,444	2.4	0.00	0.18
3	9	15,700[a]	155.1	15,680	15,680	8.1	-0.13	-0.13
4	12	15,920[a]	256.5	15,900	15,900	16.8	-0.13	-0.13
5	15	28,180[a]	264.7	28,160	28,160	37.8	-0.07	-0.07
6	18	31,400	507.6	31,400	31,400	90.9	0.00	0.00
7	21	31,520	573.6	31,520	31,720	147.8	0.00	0.63
Avg		19,757	267.9	19,749	19,781	43.4	0.00	0.20

[a]Optimal value is higher then Best TS because the exact method does not allow to use less than minimum number of cranes in the last period of vessels.

TS algorithm is run 3 times for each instance, the best and the average objective values of TS are reported in the columns represented by Best TS and Avg TS. The values in the TS Time column is equal to the total duration of three runs in seconds. The percent deviation of best and average TS values from the optimal solution is calculated as:

$$Best\,\% = \frac{(Best\,TS - Opt)}{Opt} \times 100. \tag{3.2}$$

$$Avg\,\% = \frac{(Avg\,TS - Opt)}{Opt} \times 100. \tag{3.3}$$

The negative values for percent deviation stems from the diffrence in the assumption between TS and the exact solution of (Türkoğulları *et al.*, 2012a) about the QC usage in the last service period of a vessel. The exact solution prevents assigning less than $\underline{k}^i$ to vessel i in any time period. In our solution approach, the service given to vessel i is exactly equal to the workload of that vessel. So, QC allocation may be below $\underline{k}^i$ in the last period of service. Since no extra QC usage is allowed in our model, setup costs might be below the optimal solutions for some instances. The negative deviations are excluded while calculating the average value of percent deviations.

The results of the remaining instances with $c_{i4} = 0$ is given in Tables 3.9, 3.10 and 3.11. TS finds optimal values for 31 instances out of 33 in loose arrival times. 29 out of 33 instances and 23 out of 33 instances are found to be optimal in moderate and tight arrival times, respectively. The worst values for percent deviations are reported as 6.90% and 8.05% for Best % and Avg %, respectively. Notice that the average percent deviations for both best and avreage TS values get higher as vessels arrive at the port more often. This result is expected due to the fact that as vessels arrive more frequently, the problem becomes more difficult. Furthermore, the rate of increase in Avg % is more than the rate of increase in Best % as problem gets harder. These results reveal that the probability of reaching an optimal solution in one TS run is more

likely with loose arrival times. Another point is that no integer solution is found for instances 28, 29, and 32 of tight arrival times in 200,000 seconds. The execution time of the exact method is limited by 200,000 seconds, thus non-optimal values are marked with (*), indicating the best integer solution. All in all, TS results give satisfactory results that are close to the optimal value in significantly shorter times for all generated instances.

The results of the remaining instances with $c_{i4} = 10$ are given in Tables 3.12, 3.13 and 3.14. 10, 6, 3 instances of loose, moderate and tight arrival times are found to be optimal. Optimality is not reached in many cases because we are only contented with a resource leveling heuristic to minimize setups to make advantage of speed. TS algorithm gives good results provided that setup costs is comparatively much less than the berthing related costs. Thus, the average percent deviations for best TS values remain within the range of 1% for all of the three cases.

As a further study, TS is run for 10 instances with different frequency of arrival times in the case of $c_{i4} = 50$ and $c_{i4} = 100$. We only select 10 instances due to time restrictions. The results shown in Tables 3.15 and 3.16 indicate that the average percent deviation for best values is within 2%, which is an acceptable level. This preliminary analysis points out that TS algorithm can be an efficient tool to solve BACAP for different values of c_{i4} up to some extent.

Table 3.9. Test results for loose arrival times with $c_{i4} = 0$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	15,000	243.8	15,000	15,000	3.6	0.00	0.00
2	10	18,000	155.7	18,000	18,000	11.9	0.00	0.00
3	10	0	40.5	0	0	1.5	0.00	0.00
4	15	27,000	570.0	27,000	27,000	12.6	0.00	0.00
5	15	30,000	314.1	30,000	30,000	35.9	0.00	0.00
6	15	0	75.7	0	0	4.6	0.00	0.00
7	20	30,000	715.2	30,000	30,000	42.3	0.00	0.00
8	20	38,000	498.3	38,000	38,000	78.5	0.00	0.00
9	20	3,000	110.1	3,000	3,000	12.5	0.00	0.00
10	25	39,000	622.6	39,000	39,000	97.5	0.00	0.00
11	25	43,000	1,299.0	43,000	43,000	159.8	0.00	0.00
12	25	4,000	172.5	4,000	4,000	25.8	0.00	0.00
13	30	45,000	1,245.2	45,000	45,000	153.3	0.00	0.00
14	30	56,000	2,067.0	56,000	56,000	403.6	0.00	0.00
15	30	9,000	244.0	9,000	9,000	52.3	0.00	0.00
16	35	57,000	1,484.0	57,000	57,000	299.4	0.00	0.00
17	35	70,000	3,607.0	70,000	70,200	655.2	0.00	0.29
18	35	18,000	333.0	18,000	18,000	115.7	0.00	0.00
19	40	60,000	1,321.7	60,000	60,000	754.2	0.00	0.00
20	40	76,000	2,647.0	76,000	76,200	1,106.9	0.00	0.26
21	40	25,000	481.0	25,000	25,000	234.9	0.00	0.00
22	45	69,000	1,392.9	69,000	69,000	1,056.0	0.00	0.00
23	45	80,000	3,311.0	80,000	80,400	1,237.7	0.00	0.50
24	45	30,000	540.1	30,000	30,200	290.2	0.00	0.67
25	50	75,000	1,612.9	75,000	75,000	1,274.1	0.00	0.00
26	50	83,000	5,836.0	83,000	83,000	1,987.0	0.00	0.00
27	50	30,000	636.8	31,000	31,000	429.3	3.33	3.33
28	55	87,000	2,236.0	87,000	87,000	2,063.4	0.00	0.00
29	55	87,000	7,089.0	87,000	87,200	2,690.8	0.00	0.23
30	55	40,000	3,457.1	40,000	40,800	1,503.0	0.00	2.00
31	60	90,000	5,997.1	90,000	90,000	2,435.7	0.00	0.00
32	60	97,000	7,711.0	97,000	97,800	3,655.7	0.00	0.82
33	60	41,000	8,357.9	42,000	42,000	1,629.8	2.44	2.44
Avg		44,606	2,012.9	44,667	44,752	742.9	0.17	0.32

Table 3.10. Test results for moderate arrival times with $c_{i4} = 0$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	14,000	60.7	14,000	14,000	7.6	0.00	0.00
2	10	1,000	36.4	1,000	1,000	1.0	0.00	0.00
3	10	24,000	72.9	24,000	24,400	7.4	0.00	1.67
4	15	19,000	110.4	19,000	19,000	30.4	0.00	0.00
5	15	4,000	66.4	4,000	4,000	3.5	0.00	0.00
6	15	28,000	152.0	28,000	28,400	15.8	0.00	1.43
7	20	24,000	159.7	24,000	24,000	44.0	0.00	0.00
8	20	4,000	152.8	4,000	4,000	8.4	0.00	0.00
9	20	39,000	235.3	39,000	39,400	45.2	0.00	1.03
10	25	26,000	200.5	26,000	26,000	58.1	0.00	0.00
11	25	8,000	265.0	8,000	8,000	29.5	0.00	0.00
12	25	44,000	323.0	44,000	44,400	88.3	0.00	0.91
13	30	34,000	274.7	34,000	34,000	127.8	0.00	0.00
14	30	17,000	402.6	17,000	17,000	68.5	0.00	0.00
15	30	48,000	540.2	48,000	49,200	174.6	0.00	2.50
16	35	40,000	385.5	40,000	40,000	233.5	0.00	0.00
17	35	38,000	937.4	39,000	39,800	110.1	2.63	4.74
18	35	52,000	764.7	52,000	52,600	240.2	0.00	1.15
19	40	49,000	557.4	49,000	49,000	269.2	0.00	0.00
20	40	48,000	994.2	48,000	49,400	330.6	0.00	2.92
21	40	56,000	800.5	56,000	57,000	580.5	0.00	1.79
22	45	52,000	783.2	52,000	52,000	445.8	0.00	0.00
23	45	57,000	1,691.3	57,000	59,000	578.7	0.00	3.51
24	45	61,000	3,225.0	61,000	62,200	823.9	0.00	1.97
25	50	52,000	944.9	52,000	52,000	502.2	0.00	0.00
26	50	68,000	2,690.5	69,000	70,800	1,230.5	1.47	4.12
27	50	64,000	8,726.0	64,000	64,400	1,394.3	0.00	0.63
28	55	66,000	1,151.0	66,000	66,000	968.3	0.00	0.00
29	55	72,000	3,807.1	73,000	76,200	2,139.4	1.39	5.83
30	55	64,000	10,792.0	64,000	65,200	2,108.1	0.00	1.88
31	60	71,000	1,511.0	71,000	72,200	1,402.0	0.00	1.69
32	60	83,000	11,109.9	85,000	87,000	3,408.0	2.41	4.82
33	60	64,000	20,108.0	64,000	64,800	2,812.7	0.00	1.25
Avg		42,152	2,243.4	42,303	42,921	614.8	0.24	1.33

Table 3.11. Test results for tight arrival times with $c_{i4} = 0$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	8,000	72.2	8,000	8,000	2.2	0.00	0.00
2	10	9,000	71.7	9,000	9,000	3.7	0.00	0.00
3	10	26,000	116.7	26,000	26,000	11.2	0.00	0.00
4	15	18,000	115.6	18,000	18,000	14.2	0.00	0.00
5	15	20,000	107.0	20,000	20,000	34.6	0.00	0.00
6	15	27,000	192.3	27,000	27,667	25.8	0.00	2.47
7	20	27,000	152.2	27,000	27,000	38.4	0.00	0.00
8	20	24,000	228.2	24,000	24,667	47.9	0.00	2.78
9	20	52,000	608.2	52,000	52,667	86.2	0.00	1.28
10	25	38,000	256.3	38,000	38,667	93.6	0.00	1.75
11	25	38,000	319.5	38,000	38,667	147.5	0.00	1.75
12	25	67,000	1,620.0	67,000	67,667	258.5	0.00	1.00
13	30	65,000	1,264.1	66,000	66,667	540.3	1.54	2.56
14	30	46,000	514.8	46,000	47,667	305.5	0.00	3.62
15	30	71,000	1,777.6	71,000	72,000	484.9	0.00	1.41
16	35	80,000	2,069.9	81,000	82,667	920.9	1.25	3.33
17	35	58,000	1,030.4	58,000	59,000	844.8	0.00	1.72
18	35	74,000	4,720.4	76,000	76,667	1,128.6	2.70	3.60
19	40	87,000	2,305.9	87,000	88,333	1,553.0	0.00	1.53
20	40	72,000	1,870.8	72,000	74,667	1,665.5	0.00	3.70
21	40	78,000	8,333.8	78,000	79,333	1,224.5	0.00	1.71
22	45	95,000	5,667.6	95,000	96,333	1,971.0	0.00	1.40
23	45	78,000	2,142.1	79,000	79,667	2,690.2	1.28	2.14
24	45	97,000	5,857.0	97,000	98,333	2,460.2	0.00	1.37
25	50	109,000	22,473.8	111,000	111,333	3,152.1	1.84	2.14
26	50	106,000	36,802.8	109,000	110,667	4,210.2	2.83	4.40
27	50	109,000	9,090.9	112,000	114,667	3,356.4	2.75	5.20
28	55	–[a]	200,000.0	121,000	121,667	4,892.1	–	–
29	55	–[a]	200,000.0	153,000	155,333	6,645.9	–	–
30	55	116,000	15,989.0	124,000	125,333	4,260.9	6.90	8.05
31	60	116,000	89,987.9	120,000	121,000	7,692.9	3.45	4.31
32	60	–[a]	200,000.0	175,000	178,000	11,289.0	–	–
33	60	123,000	10,349.1	128,000	130,000	8,674.2	4.07	5.69
Avg		64,467	25,033.6	73,121	74,162	2,143.2	0.95	2.30

[a]No integer solution is returned by CPLEX in 200,000 seconds

Table 3.12. Test results for loose arrival times with $c_{i4} = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	15,760	121.9	15,760	15,760	13.5	0.00	0.00
2	10	18,660	181.9	18,660	18,660	15.6	0.00	0.00
3	10	580	46.0	580	580	2.1	0.00	0.00
4	15	28,180	285.0	28,180	28,180	51.3	0.00	0.00
5	15	31,060	314.4	31,100	31,108	61.7	0.13	0.15
6	15	900	68.8	900	900	6.9	0.00	0.00
7	20	31,480	412.3	31,500	32,096	162.6	0.06	1.96
8	20	39,440	498.1	39,480	39,496	120.8	0.10	0.14
9	20	4,180	103.3	4,180	4,180	17.8	0.00	0.00
10	25	40,820	544.5	40,840	40,840	243.9	0.05	0.05
11	25	44,860	793.7	44,900	44,916	179.5	0.09	0.12
12	25	5,460	194.3	5,460	5,460	36.1	0.00	0.00
13	30	47,240	1,010.7	47,260	47,260	427.8	0.04	0.04
14	30	58,300	1,678.4	58,340	58,356	447.4	0.07	0.10
15	30	10,820	248.2	10,820	10,820	61.7	0.00	0.00
16	35	59,660	1,412.3	59,680	59,680	865.2	0.03	0.03
17	35	72,580	2,268.5	72,720	72,720	673.7	0.19	0.19
18	35	20,200	353.2	20,200	20,200	142.0	0.00	0.00
19	40	62,960	1,736.7	63,000	64,208	1,329.3	0.06	1.98
20	40	78,860	5,015.4	78,900	78,980	1,358.8	0.05	0.15
21	40	27,520	715.1	27,520	27,520	274.6	0.00	0.00
22	45	72,300	2,775.5	72,340	72,340	1,816.5	0.06	0.06
23	45	83,060	6,979.0	83,200	83,200	1,741.6	0.17	0.17
24	45	32,780	896.8	32,780	33,580	337.8	0.00	2.44
25	50	78,720	17,830.2	78,760	78,760	2,398.2	0.05	0.05
26	50	86,340	11,220.9	86,480	86,492	2,468.5	0.16	0.18
27	50	33,060	1,115.4	34,060	34,060	509.7	3.03	3.02
28	55	91,140	12,203.9	91,180	93,564	3,085.5	0.04	2.66
29	55	90,700	12,926.0	90,840	90,864	3,539.7	0.15	0.18
30	55	43,380	2,974.9	44,380	44,396	1,726.8	2.31	2.34
31	60	94,440	30,542.1	94,500	95,892	3,980.7	0.06	1.54
32	60	101,140	31,966.6	102,340	103,156	5,028.0	1.19	1.99
33	60	44,680	11,198.4	45,700	45,700	2,237.1	2.28	2.28
Avg		47,008	4,867.6	47,168	47,392	1,071.6	0.31	0.66

Table 3.13. Test results for moderate arrival times with $c_{i4} = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	14,620	61.5	14,620	14,620	10.2	0.00	0.00
2	10	1,520	27.1	1,520	1,520	1.7	0.00	0.00
3	10	24,700	78.1	24,800	24,800	12.2	0.40	0.41
4	15	19,980	107.7	20,000	20,008	37.6	0.10	0.14
5	15	4,740	40.1	4,740	4,740	5.1	0.00	0.00
6	15	29,000	158.1	29,100	30,048	29.1	0.34	3.61
7	20	25,320	110.5	25,340	25,364	47.6	0.08	0.17
8	20	5,120	155.2	5,120	5,120	14.3	0.00	0.00
9	20	40,360	259.6	40,460	40,836	78.0	0.25	1.18
10	25	27,560	170.6	27,580	27,604	75.5	0.07	0.16
11	25	9,400	374.6	9,400	9,400	34.4	0.00	0.00
12	25	45,760	621.7	45,860	46,248	167.3	0.22	1.07
13	30	35,940	405.3	35,950	35,974	148.5	0.03	0.10
14	30	18,800	338.3	18,800	18,800	75.8	0.00	0.00
15	30	50,080	877.4	50,200	50,200	224.4	0.24	0.24
16	35	42,320	619.1	42,340	42,732	260.8	0.05	0.97
17	35	40,080	1,378.1	41,100	41,460	118.1	2.54	3.44
18	35	54,420	1,361.9	54,520	57,020	310.1	0.18	4.78
19	40	51,660	914.4	51,700	51,740	305.3	0.08	0.16
20	40	50,420	3,216.1	52,560	53,293	426.2	4.24	5.70
21	40	58,900	2,022.1	59,020	60,928	292.8	0.20	3.44
22	45	54,940	1,162.8	54,980	55,188	576.0	0.07	0.45
23	45	59,740	5,279.7	61,900	62,887	609.2	3.62	5.27
24	45	64,260	2,695.7	64,320	66,076	596.3	0.09	2.83
25	50	55,260	1,713.9	55,300	55,328	859.4	0.07	0.12
26	50	71,120	7,665.7	72,400	74,020	1,396.4	1.80	4.08
27	50	67,560	3,307.6	67,640	71,404	975.1	0.12	5.69
28	55	69,600	2,172.7	69,660	69,676	1,166.6	0.09	0.11
29	55	75,400	10,961.8	78,540	78,880	2,374.2	4.16	4.62
30	55	67,860	7,148.0	67,980	71,112	1,438.5	0.18	4.79
31	60	74,900	8,928.2	74,960	75,572	1,954.7	0.08	0.90
32	60	86,780	35,464.0	88,000	91,007	4,350.6	1.41	4.87
33	60	68,160	27,100.0	68,280	70,533	1,988.9	0.18	3.48
Avg		44,433	3,845.4	44,809	45,580	635.2	0.63	1.90

Table 3.14. Test results for tight arrival times with $c_{i4} = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	8,640	67.2	8,640	8,640	3.2	0.00	0.00
2	10	9,560	115.5	9,580	9,580	7.2	0.21	0.21
3	10	26,720	130.6	26,840	27,173	15.2	0.45	1.70
4	15	18,980	129.9	18,980	19,007	20.6	0.00	0.14
5	15	20,900	339.4	20,920	20,920	45.6	0.10	0.10
6	15	28,000	174.3	28,120	28,780	33.5	0.43	2.79
7	20	28,260	175.8	28,260	28,293	49.0	0.00	0.12
8	20	25,300	374.5	25,340	25,687	74.3	0.16	1.53
9	20	53,460	2,196.2	53,620	53,713	142.4	0.30	0.47
10	25	39,720	317.3	39,820	40,380	113.1	0.25	1.66
11	25	39,600	707.6	39,700	40,367	183.4	0.25	1.94
12	25	68,820	3,232.9	70,980	71,687	359.6	3.14	4.17
13	30	67,200	1,522.0	68,160	68,213	654.3	1.43	1.51
14	30	47,920	892.9	48,020	48,377	369.9	0.21	0.95
15	30	73,730	4,839.4	74,580	75,860	724.0	1.15	2.89
16	35	82,540	6,142.8	82,620	83,320	1,031.9	0.10	0.95
17	35	60,360	2,727.5	60,520	60,873	941.1	0.27	0.85
18	35	76,560	13,462.7	76,780	77,473	984.1	0.29	1.19
19	40	89,860	42,595.9	90,080	92,080	1,701.1	0.24	2.47
20	40	74,760	2,192.2	75,940	77,273	1,936.5	1.58	3.36
21	40	80,940	39,004.3	81,160	82,473	1,752.7	0.27	1.89
22	45	98,240	50,053.1	99,260	99,287	2,219.0	1.04	1.07
23	45	81,060	3,693.9	81,200	83,920	3,186.9	0.17	3.53
24	45	100,340	166,844.8	100,660	102,033	2,117.6	0.32	1.69
25	50	112,640	134,656.7	114,980	116,740	3,320.4	2.08	3.64
26	50	109,460	46,139.4	111,740	114,107	4,949.7	2.08	4.25
27	50	$116,960^*$	200,000.0	120,100	121,753	2,952.0	2.68	4.10
28	55	120,100	199,240.9	125,320	125,373	7,042.5	4.35	4.39
29	55	$161,140^*$	200,000.0	149,240	154,360	8,928.6	-7.40	-4.21
30	55	–[a]	200,000.0	127,380	127,393	7,677.3	–	–
31	60	120,340	170,020.9	125,600	126,653	8,961.3	4.37	5.25
32	60	–[a]	200,000.0	172,800	177,247	14,144.4	–	–
33	60	–[a]	200,000.0	137,220	137,540	10,409.1	–	–
Avg		63,000	57,333.0	75,581	76,563	2,637.9	0.96	2.03

[a]No integer solution is returned by CPLEX in 200,000 seconds

Table 3.15. Test results for some instances with $c_{i4} = 50$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
Tight-2	10	11,800	177.8	11,900	11,900	4.0	0.85	0.85
Tight-6	15	32,000	179.5	32,600	33,267	29.2	1.88	3.96
Tight-7	20	33,300	222.9	33,300	33,467	43.3	0.00	0.50
Loose-11	25	52,300	1,591.1	52,500	52,500	273.8	0.38	0.38
Mod-14	30	26,000	700.2	26,000	26,000	104.3	0.00	0.00
Loose-18	35	29,000	358.0	29,000	29,000	192.3	0.00	0.00
Mod-21	40	70,500	2,139.6	72,400	72,467	583.5	2.70	2.79
Loose-22	45	85,450	3,565.1	86,700	87,633	2,039.3	1.46	2.56
Mod-25	50	68,300	1,574.4	68,500	68,500	1,519.0	0.29	0.29
Loose-28	55	107,700	10,361.2	109,800	111,100	4,077.9	1.95	3.16
Mod-31	60	90,500	4,589.3	90,800	90,800	3,978.3	0.33	0.33
Avg		55,168	2,314.5	55,773	56,058	1,167.7	0.89	1.35

Table 3.16. Test results for some instances with $c_{i4} = 100$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
Tight-2	10	14,600	200.9	14,600	14,600	5.5	0.00	0.00
Tight-6	15	37,000	240.0	40,000	40,000	23.9	8.11	8.11
Tight-7	20	39,600	195.5	41,400	42,067	46.4	4.55	4.55
Loose-11	25	61,600	3,066.7	62,000	62,000	307.5	0.65	0.65
Mod-14	30	35,000	866.5	35,000	35,000	121.0	0.00	0.00
Loose-18	35	40,000	901.8	40,000	40,000	257.1	0.00	0.00
Mod-21	40	85,000	3,817.5	86,600	86,600	511.0	1.88	1.88
Loose-22	45	$102,000^*$	40,961.0	105,200	105,267	2,748.8	3.14	3.14
Mod-25	50	84,600	2,530.1	85,000	85,000	1,940.9	0.47	0.47
Loose-28	55	$128,400^*$	25,637.5	131,600	132,533	5,314.5	2.49	2.49
Mod-31	60	110,000	16,643.2	110,600	110,733	5,328.3	0.55	0.55
Avg		67,073	8,641.9	68,364	68,527	1,509.5	1.99	1.99

4. INTEGRATED BERTH ALLOCATION AND SPECIFIC QUAY CRANE ASSIGNMENT PROBLEM

The problem of allocating specific cranes to vessels is known as the QCAP-specific. In Chapter 3, a BACAP model, which aims to determine a berth plan and a QC allocation to load/unload vessels, is provided and solved. The total setup cost of quay cranes is one of the measures to be minimized in that problem. Setup occurs in BACAP only when the number of allocated QCs to a vessel is changed between two consecutive time periods. A QC schedule can be obtained from a BACAP solution applying an intuitive heuristic. However, when we want to obtain a quay crane schedule by a post-process, we may encounter some additional QC setups. Although the number of QCs assigned to a vessel remains the same, the group of cranes assigned to it has to be modified from time to time to be able to reach a feasible quay crane schedule. In this case, the BACAP solution may not be the most preferred solution in real life, since the setups emerging from QC schedule is not taken into account. A BACAP solution gives a feasible quay crane schedule without any additional setup cost if and only if the solution satisfies a condition presented in (Türkoğulları *et al.*, 2012a). Instead of applying a post-process to the incumbent solution of BACAP, solving BAP and QCAP-specific simultaneously would produce better solutions. The integrated problem of BAP and QCAP-specific is called BACASP in this study, following the convention in (Türkoğulları *et al.*, 2012b).

4.1. Additional Assumptions

The assumptions made in BACAP about berth related decisions and quay cranes remain valid for this problem as well. Furthermore, since specific cranes are also aimed to be found in BACASP some additional assumptions should be made as follows:

- The quay cranes are labeled from 1 to N in the same increasing order as the berth sections, i.e., the beginning of the berth is labeled as crane 1 and the farthest

one is labeled as crane N, where N is the total number of QCs available at the terminal.

- The crane g is always on the left of crane $g+1$, and on the right of crane $g-1$. Therefore, cranes cannot cross each other; they are always in the same order assuming they move on a rail track.
- If cranes g and $g+M$ are assigned to the same vessel, say i, where $M = 1, \ldots, N$, then the cranes between g and $g+M$ are also assigned to vessel i.

Table 4.1. Decision variables of the BACASP.

Decision Variable	Definition
p_i	berthing position of vessel i
t_i	berthing time of vessel i
q_i	departure time of vessel i
θ_{ijkl}	1 if crane k is moved from i to j between periods l and $l+1$

The new problem BACASP requires no additional parameters. So, the same parameters presented in Table 3.1 can also be used for this problem. Although the new decision variables are presented in Table 4.1, the only difference from BACAP is the last decision variable since specific QC allocation is seeked: θ_{ijkl}, where $i, j = 0, 1, \ldots, V$, $k = 1, \ldots, N$ and $l = 1, ..., T$. Here, vessel 0 is dummy vessel which is introduced to represent idle cranes that are assigned to no vessels.

A solution of BACASP can be represented on a time-space diagram exactly following the way described in Chapter 3. The main variation is that only the numbers of assigned QCs are shown in rectangles in a BACAP solution but here the group of vessels are written in vessel i in the form of $[lm_i - rm_i]$. rm_i represents the rightmost crane and lm_i stands for the leftmost crane assigned to i. This notation implies that the cranes between rm_i and lm_i are also assigned to vessel i. In the example illustrated in Figure 4.1, vessel 1 is assigned cranes 4, 5, 6, 7 during periods 2, 3, and cranes 5, 6, 7 between periods 4 and 8. Crane schedules of vessels 2 and 3 also can be seen from

the time-space diagram.

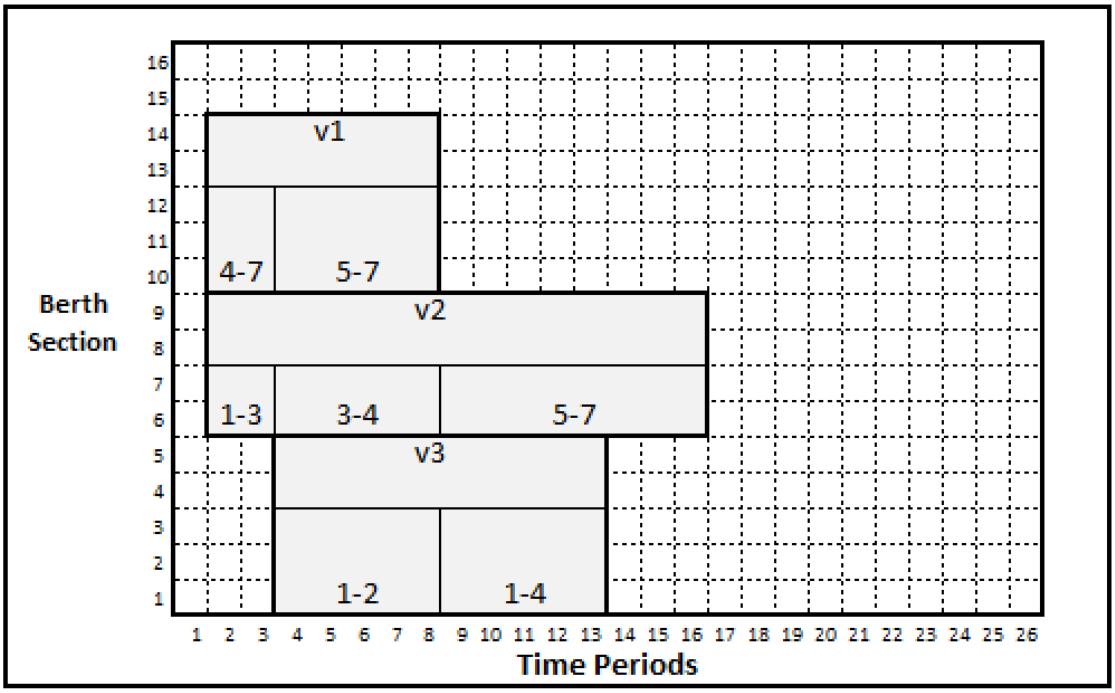

Figure 4.1. A Feasible Solution for BACASP.

4.1.1. Objective Function of BACASP

The cost of berthing away from the desired berthing position, berthing after the arrival time and departing after the due time are included as cost items in BACASP. The setup of a QC is equal to the "*move*" of a specific QC here. Hence, the cost of QC setup is also formulated in a totally different way from the setup cost of BACAP model. The cost coefficients of the objective function in BACASP can be seen in Table 4.2.

Moving a QC incurs a setup cost denoted by c_4. We say that crane k serves vessel 0 if it is idle in any time period. A move occurs if an idle crane k serves vessel i in time period l and serves vessel j in time period $l-1$ where $i = 0, 1, \ldots, N$, $j = 0, 1, \ldots, N$ and $i \neq j$. In other words if crane k moves to a vessel, or a crane k moves from a vessel

Table 4.2. Cost coefficients of the objective function in BACASP.

Parameter	Definition
c_{i1}	Cost of one unit deviation from the desired berthing section for vessel i
c_{i2}	Cost of berthing one period later than the arrival time for vessel i
c_{i3}	Cost of departing one period later than the due time for vessel i
c_4	Cost of moving a quay crane

to another one or a crane k goes idle after serving a vessel, setup occurs.

The objective function can be written as follows:

$$\begin{aligned} \min z = & \sum_{i=1}^{V} c_{i1} \left| p_i - s_i \right| + \sum_{i=1}^{V} c_{i2} \max(t_i - e_i, 0) \\ & + \sum_{i=1}^{V} c_{i3} \max(q_i - d_i, 0) + \sum_{i=0,1,\ldots,N} \sum_{j=0,1,\ldots,N} \sum_{k=1}^{N} \sum_{l=1}^{T} c_4 \theta_{ijkl} \end{aligned} \tag{4.1}$$

All QCs are idle at $l = 0$. Then we can compute the objective value using the parameters given in Table 3.5 and the following cost coefficients: $c_{i1} = c_{i2} = c_{i3} = 10$ and c_4=1.

- The total cost of deviation from desired berthing position:

$$10 \times |10 - 10| + 10 \times |6 - 9| + 10 \times |1 - 7| = 70.$$

- The total cost of waiting before berthing:

$$10 \times \max(2-2,0) + 10 \times \max(2-2,0) + 10 \times \max(4-4,0) = 0.$$

- The total cost of departing after due time:

$$10 \times \max(8-23,0) + 10 \times \max(16-24),0) + 10 \times \max(13-25,0) = 0.$$

- The total setup cost of QCs:

$$7+3+5 = 15.$$

The overall cost of the given solution in Figure 4.1 is $z = 70+0+0+15 = 85$.

4.1.2. Solution Method

In BACAP model presented in Section 3.3.1, a priority vector P, which keeps a sequence of vessels in decreasing order of priority is defined. In other words, the first vessel in P has the highest priority for berthing decisions and QC allocations. Also a rule array, R, which is a binary vector of length V, is introduced where 0 and 1 represent two different rules to allocate berths and QCs. In summary, vessels are placed on time-space diagram beginning with the first vessel on P according to the rule pointed out by the corresponding element in R. After a solution is generated, some local improvement procedures are applied and the objective value of the resulting solution is computed. Different (P, R) pairs are visited by employing a tabu search.

In BACASP, QC schedule needs to be determined before calculating the objective function. Given the values of decision variables p_i, t_i and q_i for $i = 1, \ldots, V$ and the number of QCs assigned to each vessel, we can build a directed graph and solve the shortest path problem on it by applying the method proposed in (Türkoğulları *et al.*, 2012b). This method returns the QC schedule with the minimum total setup cost.

Therefore the proposed tabu search with local improvements for BACAP can easily be modified to a solution approach for BACASP. After a solution is generated from any (P, R) pair and local refinement methods are implemented, then we can find a QC schedule by the method of (Türkoğulları *et al.*, 2012b) and calculate the objective function following the steps below:

- Find the time periods that a vessel leaves or arrives. As long as a vessel does not arrive or leave, the QC allocation does not change. In our BACAP model, we assign QCs in the last period of service in order to finish the remaining work just in time. For instance, if the remaining workload of a vessel in the last period is 1 QC-hour, then only 1 QC is assigned. Thus, we have to detect all the time periods that any change occurs in the number of allocated cranes to any one of the vessels: arrival times, departure times and last service periods of *some* vessels. Define these time periods as *stages*.
- For each stage, construct nodes as many as "possible QC schedules" for the corresponding time period. Each node is defined as an array of length N. The indices of it represent cranes and the values in the array holds the vessels. For example, the array $(1, 2, 2, 3)$ means that crane 1 is allocated to vessel 1, crane 2 and 3 are allocated to vessel 2 and crane 4 is assigned to vessel 3.
- In this graph suppose that K stages are found, then the graph is constructed as a K-partite graph such that: The arcs can only be drawn from all nodes of stage $t-1$ to all nodes of stage t, $t = 2, \ldots, K$. A weight is defined for each arc (a, b) and computed as the number of moves of QCs to go from node a to b. For instance, suppose that node a is the array (0, 1, 1, 2) and b is (1, 1, 0, 0). Since the values of the first, third and fourth indices are changed, the weight of arc from a to b is computed as 3.
- Once all nodes and arcs are defined and the graph is constructed, the source node whose outgoing arcs go to the nodes of stage 1 and the sink node whose incoming arcs come from the nodes of stage K is added. The weights of arcs linked to source and sink nodes are written. Finally, we have a graph whose nodes in stage m, $m = 1, ..., K$ stands for the possible QC allocations in time period m and

whose arcs represent the number of moves between two QC allocations.

- After the whole graph is drawn, the Shortest Path Problem (SPP) between source and sink nodes can be solved by the well-known Dijkstra's Algorithm which was first presented in (Dijkstra, 1959). The length of the obtained shortest path gives the smallest possible number of QC moves and the path will give the QC schedule with minimum setup.

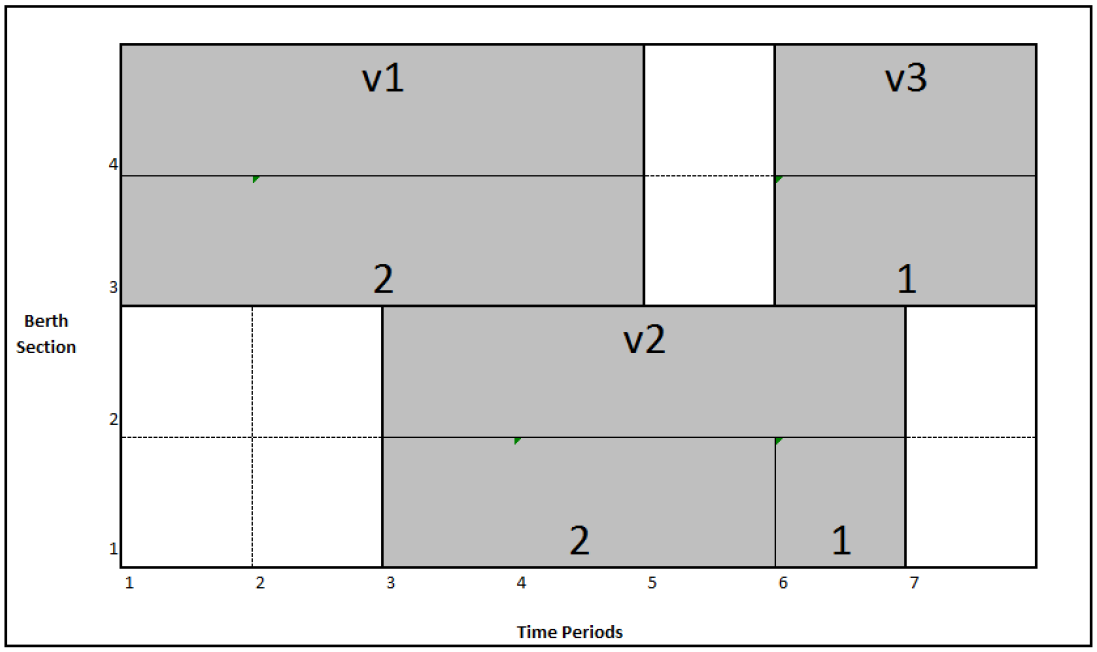

Figure 4.2. A Feasible Solution Before Finding Specific Quay Cranes.

Consider the BACAP solution given in Figure 4.2. Suppose that vessels are ordered in increasing berthing times. We know berthing positions, berthing times, completion times and the number of allocated QCs. The graph, which is revealed in Figure 4.3, can be constructed applying the method described above. Dijkstra's algorithm provides the following shortest path:

$$Source \rightarrow (0,0,1,1) \rightarrow (2,2,1,1) \rightarrow (2,2,0,0) \rightarrow (2,3,0,0) \rightarrow Sink.$$

Transformation of this path into a QC schedule is as follows: Cranes 3 and 4 are

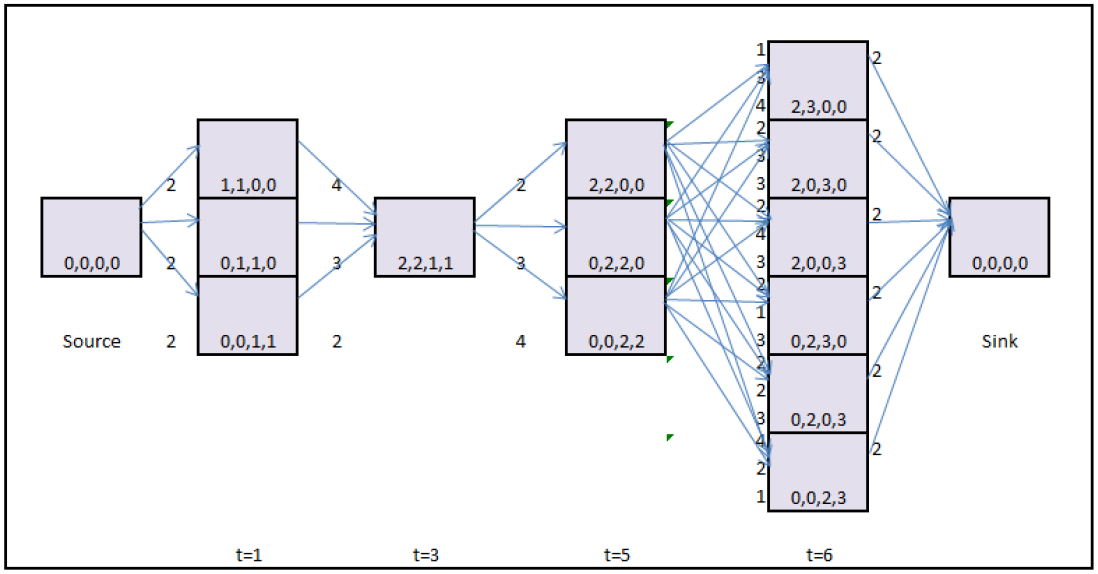

Figure 4.3. The Directed Graph to Obtain a Quay Crane Schedule.

assigned to vessel 1 in time period 1, cranes 1 and 2 are allocated to vessel 2 and cranes 3 and 4 remain in service of vessel 1 in period 3. Cranes 1 and 2 still serve vessel 2 in period 5. Crane 2 is assigned to vessel 2 and crane 3 is allocated to vessel 3 in period 6. The BACASP solution can be visualized in Figure 4.4.

Solving SPP on the graph representation of a BACAP solution is equivalent to solving the dynamic program, which is proposed in (Park and Kim, 2003) in order to find the QC schedule with minimum setup cost. Even though, SPP is known as a problem that can be solved in a polynomial time, the number of nodes may be huge for large instances and longer planning horizons. Thus, it must be kept in mind that SPP might lose its efficiency for larger instances.

4.1.3. Integration of Shortest Path Problem to Tabu Search Algorithm

Each visited BACAP solution is transformed into a BACASP solution by applying the following processes in the same order:

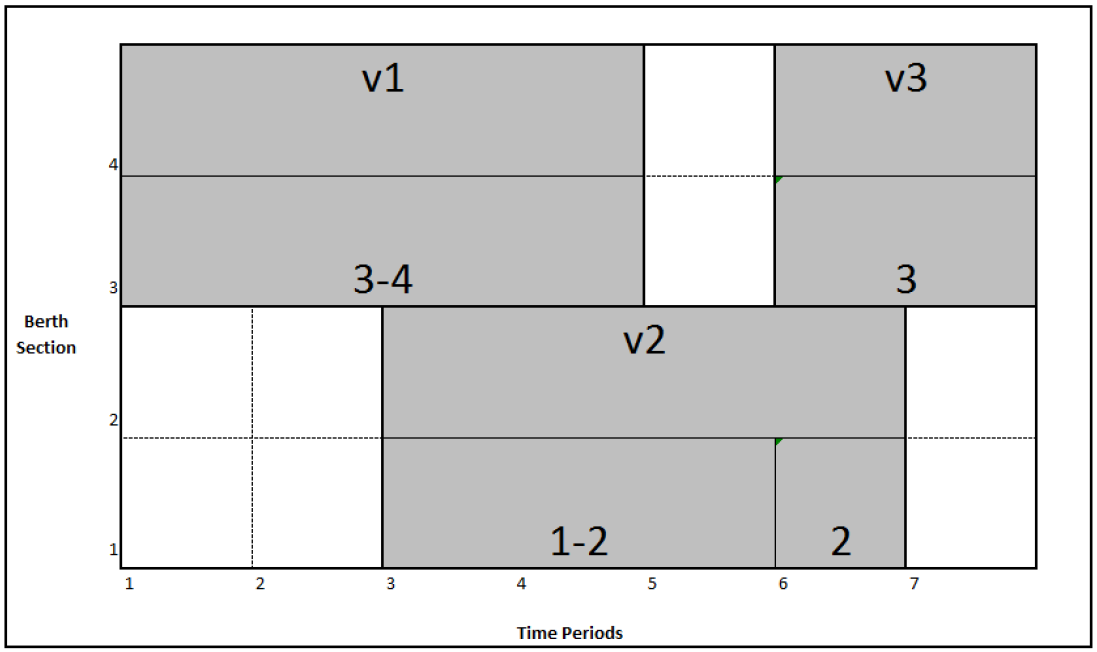

Figure 4.4. The BACASP Solution.

- A solution is obtained from a (P, R) pair and p_i, t_i, q_i and y_{ij} are found for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- Berthing times are checked and improved and t_i, q_i and y_{ij} are updated for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- Completion times are checked and improved and t_i, q_i and y_{ij} are updated for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- Spatial improvement procedure is applied and p_i is updated for $i = 1, \ldots, V$.
- Assigned QCs are smoothed beginning with the first vessel in P. y_{ij} and q_i are updated for $j = 1, \ldots, T$ and $i = 1, \ldots, V$.
- The shortest path problem is solved on the graph which is derived from the BACAP solution and the minimum cost-QC schedule is found.
- The objective value of the BACASP solution is computed.

As a conclusion, we have a tabu search metaheuristic which explores the solution space of BACASP.

4.2. Experimental Results for BACASP

TS parameters and cost coefficients c_{i1}, c_{i2} and c_{i3} are determined as explained in Section 3.5. The setup cost of BACASP, c_4, is set to 10 and the new TS algorithm with SPP is experimented on 106 instances which are generated for BACAP model. The construction algorithm of QC schedule graph and the Dijkstra algorithm is coded and integrated with BACAP solution in C# environment on Microsoft Visual Studio 2010. The results are compared with the optimal results obtained by the exact model proposed in (Türkoğulları *et al.*, 2012b) with the help of IBM ILOG CPLEX 11.0. All the experiments are carried out on a computer with Microsoft Windows Server 2003 x64 Edition operating system and Intel Xeon CPU X5460 3.16 GHz processor with 27.9 GB RAM.

TS algorithm for BACASP is run 3 times for each instance, the best and the average objective values are reported in the columns represented by Best TS and Avg TS. The values in TS Time column is equal to the total duration of three runs in seconds. The percent deviation of best and average TS values from the optimal solution is calculated as:

$$Best\,\% = \frac{(Best\,TS - Opt)}{Opt} \times 100. \tag{4.2}$$

$$Avg\,\% = \frac{(Avg\,TS - Opt)}{Opt} \times 100. \tag{4.3}$$

The results are given in Tables 4.3, 4.4, 4.5, 4.6. In some instances, TS produces better values than the optimal. The source of discrepancy is the different assumptions of TS and exact method about QC allocation approach in the last period of vessels. The execution time of exact method is bounded by 200,000 seconds, so non-optimal values are marked with (*), indicating the best integer solution. If the best found integer is not sufficiently close to the optimal, negative percent deviations may emerge.

The average value of deviations is calculated ignoring the negative values.

Due to the fact that BACASP is a more difficult problem than BACAP, CPLEX could not solve 23 out of 106 instances to optimality within the given time limit. In the group where vessels arrive more frequently, no integer solution is obtained for the instances above 45 vessels. TS algorithm for BACASP can be advantageous when the interarrival times of vessels are shorter. In general, the average percent deviations for best values are within 2% while the average percent deviations for average values do not exceed 3% for different interarrival times. The highest deviations are recorded as 4.57% and 7.70% for best and average values. Another visible consequence is that the mean percent deviatons for both incumbent and average TS results for BACASP are slightly worse than the results for BACAP. Although the QC schedule is found with minimum setup cost, TS algorithm becomes a bit inefficient when the definition of setup cost is changed. Nevertheless, the results of TS algorithm for BACASP is promising for $c_4 = 10$.

Additional experiments are performed for $c_4 = 50$ and $c_4 = 100$ and the results can be seen in Tables 4.7 and 4.8. The interesting result is that increasing the value of c_4 makes the problem even harder to solve. Because, instances such as Loose-28 and Mod-31, whose optimal values are found when $c_4 = 10$, cannot be solved by CPLEX when $c_4 = 50$ and $c_4 = 100$. The average percent deviations for best TS results remain under 3% and 5% for $c_4 = 50$ and $c_4 = 100$, respectively.

Table 4.3. The instance given in (Zhang *et al.*, 2010) with $c_4 = 0$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	3	2,160	48.6	2,160	2,160	2.4	0.00	0.00
2	6	13,330	3,024.2	13,380	13,386	13.7	0.38	0.42
3	9	15,510	655.8	15,600	15,600	50.2	0.58	0.58
4	12	15,670	5,552.6	15,770	15,770	86.4	0.64	0.64
5	15	27,840	7,447.5	27,980	27,980	180.1	0.50	0.50
6	18	30,980	10,338.2	31,130	31,130	341.3	0.48	0.48
7	21	31,100	16,613.9	31,240	31,248	760.7	0.45	0.48
Avg		19,513	6,240.1	19,609	19,611	205.0	0.43	0.44

Table 4.4. Test results for loose arrival times with $c_4 = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	15,650	2,362.6	15,650	15,650	20.7	0.00	0.00
2	10	18,530	549.1	18,530	18,530	48.8	0.00	0.00
3	10	460	100.5	450	450	6.0	-2.17	-2.17
4	15	27,840	11,026.2	27,970	27,970	87.6	0.47	0.47
5	15	30,810	9,908.9	30,910	30,910	278.0	0.32	0.32
6	15	720	150.5	710	710	34.4	-1.39	-1.39
7	20	31,060	19,691.4	31,210	31,210	291.3	0.48	0.48
8	20	39,070	14,774.7	39,250	39,250	1,648.8	0.46	0.46
9	20	3,940	5,989.6	3,970	3,970	117.9	0.76	0.76
10	25	40,360	22,757.9	40,530	40,536	952.8	0.42	0.44
11	25	44,380	65,880.4	44,620	44,620	2,250.6	0.54	0.54
12	25	5,170	6,385.3	5,220	5,220	304.3	0.97	0.97
13	30	46,630	33,965.1	46,850	46,858	1549.2	0.47	0.49
14	30	57,690	97,016.3	57,990	58,018	5,037.9	0.52	0.57
15	30	10,410	10,145.6	10,500	10,500	667.3	0.86	0.86
16	35	58,960	38,034.4	59,180	59,180	2272.8	0.37	0.37
17	35	72,050	94,601.2	72,280	72,300	4,415.4	0.32	0.35
18	35	19,680	15,423.6	19,780	19,780	951.3	0.51	0.51
19	40	62,120	39,918.2	62,420	62,420	3451.2	0.48	0.48
20	40	78,240	125,033.4	78,490	78,900	7,669.8	0.32	0.84
21	40	26,890	18,690.6	27,020	27,020	2,005.9	0.48	0.48
22	45	71,490	47,900.1	71,740	71,748	5644.8	0.35	0.36
23	45	$82,200^*$	200,000.0	82,690	83,318	8,095.1	0.60	1.36
24	45	32,060	28,034.1	33,220	33,220	3,627.6	3.62	3.62
25	50	77,780	48,566.4	78,070	78,070	7971.6	0.37	0.37
26	50	–[a]	200,000.0	85,940	86,390	9,447.0	–	–
27	50	32,300	35,826.9	33,480	33,480	5,692.0	3.65	3.65
28	55	90,090	168,627.4	90,380	90,388	11676.3	0.32	0.33
29	55	–[a]	200,000.0	90,070	90,812	10,431.1	–	–
30	55	42,710	48,039.1	43,760	43,760	9,231.3	2.46	2.46
31	60	93,180	196,453.7	93,620	93,626	17653.2	0.47	0.48
32	60	–[a]	200,000.0	101,550	102,384	12,816.0	–	–
33	60	43,750	57,500.2	44,920	45,354	11,982.7	2.67	3.67
Avg		40,483	62,525.9	46,757	46,865	4,494.9	0.83	0.92

[a]No integer solution is returned by CPLEX in 200,000 seconds

Table 4.5. Test results for moderate arrival times with $c_4 = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	14,480	3,410.8	14,540	14,540	39.0	0.41	0.41
2	10	1,380	2,376.7	1,400	1,400	53.3	1.45	1.45
3	10	24,510	4,113.5	24,690	25,303	44.1	0.73	3.24
4	15	19,720	4,559.2	19,780	19,798	129.0	0.30	0.40
5	15	4,570	2,743.5	4,670	4,670	253.9	2.19	2.19
6	15	28,740	6,039.6	28,920	30,147	147.2	0.63	4.89
7	20	24,980	6,233.8	25,090	25,102	367.9	0.44	0.49
8	20	4,830	6,021.4	4,950	4,950	485.3	2.48	2.48
9	20	40,000	10,526.2	40,200	41,427	396.0	0.50	3.57
10	25	27,160	8,788.3	27,250	27,934	713.5	0.33	2.85
11	25	9,020	30069,1	9,160	9,180	1,333.6	1.55	1.77
12	25	45,220	16,136.8	45,460	46,387	642.5	0.53	2.58
13	30	35,420	12,017.9	35,440	35,542	1,410.8	0.06	0.34
14	30	18,290	35150,8	18,520	18,527	1,742.9	1.26	1.29
15	30	49,410	20,224.2	49,700	50,627	1,134.6	0.59	2.46
16	35	41,710	23,070.8	41,850	41,882	2,438.2	0.34	0.41
17	35	39,490	49191,4	40,850	41,443	2,670.5	3.44	4.95
18	35	53,630	27,240.7	53,920	54,233	2,046.8	0.54	1.12
19	40	50,970	32,400.4	51,130	51,750	5,845.6	0.31	1.53
20	40	49,690	11,125.0	52,990	53,320	4,335.0	6.64	7.31
21	40	57,970	49,175.7	58,280	58,280	2,769.5	0.53	0.53
22	45	54,190	38,215.1	54,390	55,430	7,565.5	0.37	2.29
23	45	59,110	144058,6	61,290	63,660	6,854.9	3.69	7.70
24	45	63,200	50,222.0	63,560	64,777	4,331.9	0.57	2.49
25	50	54,450	39,015.3	54,550	54,577	10,429.3	0.18	0.23
26	50	$118,380^*$	200,000.0	78,590	78,907	8,545.4	-33.61	-33.34
27	50	66,410	63,606.5	66,770	68,337	7,547.1	0.54	2.90
28	55	68,730	71,513.2	68,840	68,877	10,869.6	0.16	0.21
29	55	–[a]	200,000.0	81,900	82,240	10,957.2	–	–
30	55	66,590	76,862.4	66,990	67,930	11,025.1	0.60	2.01
31	60	74,000	184,331.9	74,100	74,120	17,004.8	0.14	0.16
32	60	–[a]	200,000.0	92,180	93,887	14,283.3	–	–
33	60	66,920	106,020.7	67,220	68,447	17,460.7	0.45	2.28
Avg		43,005	50,896.3	44,823	45,383	4,723.5	1.07	2.22

[a]No integer solution is returned by CPLEX in 200,000 seconds

Table 4.6. Test results for tight arrival times with $c_4 = 10$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
1	10	8,480	2,765.6	8,550	8,550	21.7	0.83	0.83
2	10	9,480	539.8	9,500	9,500	71.7	0.21	0.21
3	10	26,510	3,095.3	26,700	27,667	35.7	0.72	4.36
4	15	18,720	5,635.9	18,780	18,780	107.0	0.32	0.32
5	15	20,690	7,180.4	20,770	21,470	257.8	0.39	3.77
6	15	27,750	6,997.0	27,960	29,233	136.0	0.76	5.35
7	20	27,900	7,263.4	28,000	28,000	629.9	0.36	0.36
8	20	24,940	14,204.8	25,090	25,433	515.0	0.60	1.98
9	20	53,030	74,831.5	53,390	54,353	388.2	0.68	2.50
10	25	39,230	11,108.6	40,350	40,350	1,112.8	2.85	2.85
11	25	39,150	18,873.0	39,310	39,653	1,645.7	0.41	1.29
12	25	68,250	178,100.0	68,580	69,590	741.0	0.48	1.96
13	30	66,480	25,262.7	68,640	68,663	4,068.8	3.25	3.28
14	30	47,310	23,450.2	47,540	47,930	2,908.2	0.49	1.31
15	30	–[a]	200,000.0	73,020	74,637	1,332.1	–	–
16	35	–[a]	200,000.0	82,980	82,980	5,421.4	–	–
17	35	59,620	39,851.1	61,860	61,883	5,088.9	3.76	3.80
18	35	–[a]	200,000.0	77,070	77,427	3,175.5	–	–
19	40	–[a]	200,000.0	89,370	89,970	7,975.7	–	–
20	40	73,910	83,323.3	77,290	77,317	4,919.2	4.57	4.61
21	40	–[a]	200,000.0	82,320	83,053	3,656.0	–	–
22	45	–[a]	200,000.0	98,650	99,987	9,407.2	–	–
23	45	80,370	127,332.4	83,560	84,247	12,588.3	3.97	4.82
24	45	–[a]	200,000.0	98,920	101,110	8,444.5	–	–
25	50	–[a]	200,000.0	112,020	112,680	11,431.2	–	–
26	50	–[a]	200,000.0	112,970	113,670	14,761.0	–	–
27	50	–[a]	200,000.0	117,200	117,800	13,313.0	–	–
28	55	–[a]	200,000.0	124,360	125,020	14,326.1	–	–
29	55	–[a]	200,000.0	156,350	157,343	16,552.0	–	–
30	55	–[a]	200,000.0	120,430	122,780	15,900.5	–	–
31	60	–[a]	200,000.0	129,460	130,423	27,930.0	–	–
32	60	–[a]	200,000.0	173,500	177,630	18,423.7	–	–
33	60	–[a]	200,000.0	129,830	131,007	18,205.6	–	–
Avg		40,695	116,055.0	75,282	76,065	6,833.1	1.45	2.56

[a]No integer solution is returned by CPLEX in 200,000 seconds

Table 4.7. Test results for some instances with $c_4 = 50$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
Tight-2	10	11,400	475.0	11,500	11,500	111.3	0.88	0.88
Tight-6	15	30,750	5,660.3	31,800	32,316	148.5	3.41	5.10
Tight-7	20	31,500	7,800.0	32,000	32,000	829.4	1.59	1.59
Loose-11	25	49,900	29,666.4	51,100	51,150	1,711.8	2.40	2.51
Mod-14	30	23,450	45,797.2	24,600	24,600	2,538.1	4.90	4.90
Loose-18	35	26,400	15,478.0	26,900	26,900	1,778.7	1.89	1.89
Mod-21	40	65,850	37,812.4	67,400	67,866	5,155.8	2.35	3.06
Loose-22	45	81,100	110,287.5	84,850	84,883	5,858.1	4.62	4.67
Mod-25	50	63,850	46,143.4	64,700	64,700	15,521.1	1.33	1.33
Loose-28	55	–[a]	200,000.0	106,000	107,783	12,886.5	–	–
Mod-31	60	–[a]	200,000.0	85,850	85,900	14,876.3	–	–
Avg		42,689	63,556.4	53,336	53,600	5,583.2	2.60	2.88

[a]No integer solution is returned by CPLEX in 200,000 seconds

Table 4.8. Test results for some instances with $c_4 = 100$.

#	V	Opt. Obj. Value	CPU time(s) (Exact)	Best TS	Avg TS	CPU time(s) (TS)	Best %	Avg %
Tight-2	10	13,800	484.8	14,000	14,000	72.2	1.45	1.45
Tight-6	15	34,500	5,304.8	36,600	37,333	122.4	6.09	8.21
Tight-7	20	36,000	8,076.0	37,000	37,333	898.5	2.78	3.70
Loose-11	25	56,800	76,854.8	59,200	59,400	1,420.3	4.23	4.58
Mod-14	30	29,900	45,512.9	32,200	32,200	2,589.1	7.69	7.69
Loose-18	35	34,800	14,108.3	35,800	35,800	1,861.8	2.87	2.87
Mod-21	40	75,700	38,991.4	79,600	79,600	3,171.6	5.15	5.15
Loose-22	45	–[a]	200,000.0	95,900	97,960	6,418.5	–	–
Mod-25	50	75,700	49,748.1	77,400	77,400	16,683.3	2.25	2.25
Loose-28	55	–[a]	200,000.0	123,300	124,333	12,728.4	–	–
Mod-31	60	–[a]	200,000.0	101,700	101,700	15,694.2		
Avg		44,650	63,908.1	62,973	63,369	5,605.5	4.06	4.49

[a]No integer solution is returned by CPLEX in 200,000 seconds

5. CONCLUSION

In this study, we develop a solution approach which integrates a TS algorithm employing a two-layered neighborhood structure and some local improvement procedures. Local improvement procedures refine the tabu search algorithm in order to visit the unexplored solutions for BACAP. The solution procedure for BACASP is quite similar to that of BACAP. Every BACAP solution can be represented as a graph and a QC schedule is generated by solving a shortest path algorithm on that graph. A BACAP solution with a crane schedule is equivalent to a BACASP solution. Hence, TS algorithm and local procedures are modified so that the metaheuristic visits BACASP solutions at each iteration.

Instances are generated under three different conditions assuming that arrival times are loose, moderate and tight. BACAP is solved on 106 different instances. TS algorithm is run three times; the average objective value and the incumbent value for each instance are reported. The performance of the solution approach is measured by the percent deviation from the optimum or best integer found by the exact method presented in (Türkoğulları *et al.*, 2012a). Average of percent deviations is found to be the smallest when arrival times are loose. As the arrivals become more frequent, the percent deviations become greater. However, the TS algorithm keeps producing near optimal solutions in shorter times despite the tightening arrival times. Another important observation is that the TS algorithm is capable of finding the optimal solutions of 85% of the instances when crane setups are ignored. It can be concluded that as the setup cost of QCs are increased, the quality of the TS algorithm deteriorates. Additional experiments show that TS algorithm maintains to produce good quality solutions as setup cost is increased up to some extent. The CPU times for TS are visibly less than the CPU times for the exact method.

TS algorithm for BACASP is implemented to the same 106 instances. TS algorithm is run three times; the average objective value and the incumbent value for each

instance are reported. The TS solutions are compared with optimum or best integer found by the exact method presented in (Türkoğulları *et al.*, 2012b). Not surprisingly, the average of percent deviations for loose arrival times is smaller than the average of percent deviations for moderate and tight arrival times. The percent deviations on average stay within 3% away from the optimal. TS algorithm is proved to give slightly better results for BACAP compared to BACASP. Furthermore, the probability of finding optimal solutions by TS is decreased when applied to BACASP. The gap between the computational times of TS and the exact method are even greater for BACASP.

The performance of TS algorithm can be improved by incorporating a neighborhood structure for QC allocation. The modified neighborhood structure may increase the flexibility of the algorithm against increasing setup costs. The same problem BACAP and BACASP can be solved by different assumptions such as berth dependent handling times or quay cranes with coverage ranges. Apart from this, quayside operations can be integrated with yard operations as a further study since these operations are closely related by nature.

REFERENCES

Ak, A., 2008, *Berth and Quay Crane Scheduling: Problems, Models and Solution Methods*, Ph.D. Thesis, Georgia Institute of Technology.

Bierwirth, C. and F. Meisel, 2010, "A Survey of Berth Allocation and Quay Crane Scheduling Problems in Container Terminals", *European Journal of Operational Research*, Vol. 202, pp. 615-627, 2010.

Briano, C., E. Briano, and A. G. Bruzzone, 2005, "Models for Support Maritime Logistics: A Case Study for Improving Terminal Planning", *Proceedings of the 19th European Conference on Modelling and Simulation (ECMS)*, pp. 199-205.

Brown, G. G., S. Lawphongpanich, and K. P. Thurman, 1994, "Optimizing Ship Berthing". *Naval Research Logistics*, Vol. 41, No. 1, pp. 1-15.

Cordeau, J. F., G. Laporte, P. Legato, and L. Moccia, 2005, "Models and Tabu Search Heuristics for the Berth-allocation Problem", *Transportation Science*, Vol. 39, No. 4, pp. 526-538.

Daganzo, C. F., 1989, "The Crane Scheduling Problem", *Transportation Research Part B*, Vol. 23, No. 3, pp. 159-175.

Peterkofsky, R. I., and C. F. Daganzo, 1990, "A Branch and Bound Solution Method for the Crane Scheduling Problem", *Transportation Research Part B*, Vol. 24, No. 3, pp. 159-172.

Dijkstra, E. W., 1959, "A Note on Two Problems in Connection with Graphs", *Numerische Math*, Vol. 1, pp. 269-271.

Edmond, E. D., and R. P. Maggs, 1978, "How Useful Are Queue Models in Port Investment Decisions for Container Berths", *Journal of the Operational Research Society*, Vol. 29, No. 8, pp. 741-750.

Giallombardo, G., L. Moccia, M. Salani, and I. Vacca, 2008, "The Tactical Berth Allocation Problem with Quay Crane Assignment and Transshipment-related Quadratic Yard Costs", *Proceedings of the European Transport Conference (ETC)*, pp. 1-27.

Glover, F., 1986, "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, Vol. 5, pp. 533-549.

Glover, F., and M. Laguna, 1997, *Future Paths for Integer Programming and Links to Artificial Intelligence*, Kluver Academic Publishers.

Guan, Y., and R. K. Cheung, 2004, "The Berth Allocation Problem: Models and Solution Methods", *OR Spectrum*, Vol. 26, No. 1, pp. 75-92.

Guan, Y., W. Q. Xiao, R. K. Cheung, and C. L. Li, 2002, "A Multiprocessor Task Scheduling Model for Berth Allocation: Heuristic and Worst-case Analysis", *Operations Research Letters*, Vol. 30, No. 5, pp. 343-350.

Han, M., P. Li, and J. Sun, 2006, "The Algorithm for Berth Scheduling Problem by the Hybrid Optimization Strategy GASA", *Proceedings of the Ninth International Conference on Control, Automation, Robotics and Vision (ICARCV'06)*, Washington DC, pp. 1-4.

Hansen, P., and C. Oğuz, 2003, "A Note on Formulations of Static and Dynamic Berth Allocation Problems", *Les Cahiers du GERAD*, Vol. 30, pp. 1-17.

Hansen, P., and C. Oğuz, Mladenovic, N., 2008, "Variable Neighborhood Search for Minimum Cost Berth Allocation", *European Journal of Operational Research*, Vol. 191, No. 3, pp. 636-649.

Hendriks, M. P. M., M. Laumanns, E. Lefeber, and J. T. Udding, 2008, "Robust Periodic Berth Planning of Container Vessels", *Proceedings of the Third German–Korean Workshop on Container Terminal Management: IT-based Planning and Control of Seaport Container Terminals and Transportation Systems*, pp. 1-13.

Imai, A., H. C. Chen, E. Nishimura, and S. Papadimitriou, 2008a, "The Simultaneous Berth and Quay Crane Allocation Problem", *Transportation Research Part E*, Vol. 44, No. 5, pp. 900-920.

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

Imai, A., E. Nishimura, and S. Papadimitriou, 2001, "The Dynamic Berth Allocation Problem for a Container Port", *Transportation Research Part B*, Vol. 35, No. 4, pp. 401-417.

Imai, A., E. Nishimura, and S. Papadimitriou, 2008b, "Berthing Ships at a Multi-user Container Terminal with a Limited Quay Capacity". *Transportation Research Part E*, Vol. 44, No. 1, pp. 136-151.

Imai, A., X. Sun, E. Nishimura, and S. Papadimitriou, 2005, "Berth Allocation in a Container Port: Using a Continuous Location Space Approach", *Transportation Research Part B*, Vol. 39, No. 3, pp. 199–221.

Kim, K.H., and K. C. Moon, 2003, "Berth Scheduling by Simulated Annealing", *Transportation Research Part B*, Vol. 37, No. 6, pp. 541-560.

Lai, K.K., and K. Shih, 1992, "A Study of Container Berth Allocation", *Journal of Advanced Transportation*, Vol. 26, No. 1, pp. 45-60.

Li, C. L., X. Cai, and C. Y. Lee, 1998, "Scheduling with Multiple-job-on-one-processor Pattern", *IIE Transactions*, Vol. 30, No. 5, pp. 433-445.

Lim, A., 1998, "The Berth Planning Problem", *Operations Research Letters*, Vol. 22, No 2, pp. 105-110.

Lim, A., 1999, "An Effective Ship Berthing Algorithm", *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99- vol-1)*, pp. 594-599.

Lokuge, P., and D. Alahakoon, 2007, "Improving the Adaptability in Automated Vessel Scheduling in Container Ports Using Intelligent Software Agents", *European Journal of Operational Research*, Vol.177, No. 3, pp. 1985–2015.

Mauri, G.R., A. C. M. Oliveira, and L. A. N. Lorena, 2008, "A Hybrid Column Generation Approach for the Berth Allocation Problem", *Eighth European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2008)*, pp. 110-122.

Meier, L., and R. Schumann, 2007, "Coordination of Interdependent Planning Systems, a Case Study", In: Koschke, R., Otthein, H., Rödiger, K.-H., Ronthaler, M. (Eds.), *Lecture Notes in Informatics (LNI) P-109*, Köllen Druck+Verlag GmbH, Bonn, pp. 389-396.

Meisel, F., and C. Bierwirth, 2009, "Heuristics for the Integration of Crane Productivity in the Berth Allocation Problem", *Transportation Research Part E*, Vol. 45, No. 1, pp. 196-209.

Moorthy, R., and C. P. Teo, 2006, "Berth Management in Container Terminal: the Template Design Problem", *OR Spectrum*, Vol. 28, No. 4, pp. 495-518.

Nishimura, E., A. Imai, and S. Papadimitriou, 2001, "Berth Allocation Planning in the Public Berth System by Genetic Algorithms", *European Journal of Operational Research*, Vol. 131, No 2, pp. 282-292.

Park, K.T., and K. H. Kim, 2002, "Berth Scheduling for Container Terminals by Using a Subgradient Optimization Technique", *Journal of the Operational Research Society*, Vol. 53, No. 9, pp. 1054-1062.

Park, Y.M., and K. H. Kim, 2003, "A Scheduling Method for Berth and Quay Cranes", *OR Spectrum*, Vol. 25, No. 1, pp. 1-23.

Schonfeld, P., and S. Frank, 1984, "Optimizing the Use of a Containership Berth", *Transportation Research Record*, Vol. 984, pp. 56-62.

Tavakkoli-Moghaddam, R., A. Makui, S. Salahi, M. Bazzazi, and F. Taheri, 2009, "An Efficient Algorithm for Solving a New Mathematical Model for a Quay Crane Scheduling Problem in Container Ports", *Computers and Industrial Engineering*, Vol. 56, No. 1, pp. 241-248.

Türkoğulları, Y. B., C. Taşkın, N. Aras, and K. Altınel, 2012, "Berth Allocation and Specific Quay Crane Assignment in Container Terminals", Technical Report, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Istanbul, FBE-IE.

Türkoğulları, Y. B., C. Taşkın, N. Aras, and K. Altınel, 2012, "Optimal Berth Allocation, Quay Crane Assignment and Quay Crane Scheduling", Technical Report, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Istanbul, FBE-IE.

Türkoğulları, Y. B., C. Taşkın, N. Aras, and K. Altınel, 2012, "Simultaneous Optimization of Berth Allocation, Quay Crane Assignment and Quay Crane Scheduling Problems in Container Terminals", Technical Report, Institute for Graduate Studies in Science and Engineering, Boğaziçi University, Istanbul, FBE-IE-03/2012-03.

UNCTAD, 2012, *Review of Maritime Transport, United Nations Conference on Trade and Development*, http://www.unctad.org, accessed at January 2013.

Wang, F., and A. Lim, 2007. "A Stochastic Beam Search for the Berth Allocation Problem", *Decision Support Systems*, Vol. 42, No. 4, pp. 2186-2196.

Zhang, C., L. Zheng, Z. Zhang, L. Shi, and A. Armstrong, 2010. "The Allocation of Berths and Quay Cranes by Using a Sub-Gradient Optimization Technique", *Computers and Industrial Engineering*, Vol. 58, pp. 40-50.

Zhou, P., H. Kang, and L. Lin, 2006. "A Dynamic Berth Allocation Model Based on Stochastic Consideration", *Proceedings of the Sixth World Congress on Intelligent Control and Automation (WCICA 2006)*, Vol. 2, pp. 7297–7301.