CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Column Generation Framework for Berth Scheduling at Port Terminals

Yousra Saadaoui
Nitish Umang
Emma Frejinger

May 2015

CIRRELT-2015-15

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Column Generation Framework for Berth Scheduling at Port Terminals

Yousra Saadaoui, Nitish Umang*, Emma Frejinger

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Computer Science and Operations Research, Université de Montréal, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

Abstract. In this paper, we propose a methodological framework based on column generation to solve large problem instances of the berth allocation problem (BAP) at port terminals. In past research, it has been conclusively shown that the BAP can be effectively modeled and solved as a set partitioning (SP) problem for relatively large problem size. However a major drawback of this approach is the explosion in the number of feasible assignments of vessels with increase in problem size. This is because the assignments (columns) are generated apriori in a static manner and provided as an input to the optimization model, which causes the optimization solver to run out of memory. In this study, we show how this issue can be resolved by generating the assignments in a dynamic fashion. We propose a column generation based algorithm to address the problem that can be easily adapted to solve any variant of the BAP based on different spatial and temporal attributes. We test our approach on a discrete berth allocation model with dynamic vessel arrivals and berth dependent handling times. Computational experiments on a set of artificial instances indicate that the proposed methodology can solve even very large problem sizes to optimality or near optimality in computational time of only a few minutes.

Keywords: Port logistics, berth scheduling, column generation, set-partitioning, mixed integer programming.

Acknowledgements. This research was funded by Mitacs accelerate and Natural Sciences and Engineering Research Council of Canada (NSERC) discovery grants.

---

* Corresponding author: Nitish.Umang@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2015

1 Introduction

A port terminal provides transfer facilities for containers among sea vessels and land transportation modes. It is a zone of the port where vessels dock on a berth and containers are loaded, unloaded and stored in a buffer area called yard. The berth allocation problem (BAP) is one of the most critical and widely studied problems in seaside port operations planning. It refers to the problem of assigning a set of vessels to a given berth layout in a given planning horizon with a certain objective. The objectives include minimization of the service times to vessels, minimization of port stay time, minimization of number of rejected vessels, minimization of deviation between actual and planned berthing schedules etc. The existing models for BAP in operations research literature can be classified on the basis of the temporal attributes such as vessel arrival process and handling times of vessels as well as spatial attributes relating to the berthing layout and draft restrictions. We refer the reader to Bierwirth and Meisel (2010) to study the different classification schemes for the berth allocation problem.

In the operations research literature on port operations planning, there are several studies on the BAP. While most of these studies have focused on heuristics to obtain good suboptimal solutions to the BAP, a few recent studies propose exact methods to solve realistic sized instances of the BAP in reasonable computation time. In this paper, we focus on the set partitioning (SP) approach to solve the BAP, with the objective to address the limitations of this approach in solving large problem instances. In such cases, the method can be very slow or can cause the optimization solver to run out of memory owing to an explosion in the number of variables and constraints in the SP formulation. The main contribution of this study is to propose a novel dynamic column generation scheme that is capable of solving very large problem size to optimality or near optimality in small computation time. Moreover the approach can be easily adapted to solve any variant of the BAP with respect to the berthing layout or any other attribute(s).

The remainder of the paper is organized as follows. In section 2 we provide a literature review of the past studies on the BAP with a focus on exact algorithms to solve the BAP. In section 3, we provide a mathematical definition of the problem. In section 4, we describe the SP method and propose the dynamic column generation based algorithm to address the limitations of the SP approach. Results and analysis from computational experiments on a set of artificial instances are presented in section 5. Finally in section 6, the key findings of the study are summarized and some possible directions for future research are discussed.

2 Literature review

The berth allocation problem (BAP) has been extensively covered in the past literature. Some of the important works include Imai et al. (1997), Imai et al. (2001), Imai et al. (2003), Imai et al. (2005), Kim and Moon (2003), Cordeau et al. (2005), Buhrkal et al. (2011) and Umang et al. (2013). Comprehensive literature surveys covering optimization based approaches to the BAP can be found in Bierwirth and Meisel (2010), Steenken et al. (2004), Stahlbock and Voss (2008) and more recently Bierwirth and Meisel (2015).

In this section, we focus on literature on exact algorithms to solve the BAP. Buhrkal et al. (2011) show that the SP method outperforms the previously known best models including the multi-depot vehicle routing problem with time windows (MDVRPTW) based formulation by Cordeau et al. (2005) and the re-formulation of Monaco and Sammarra (2007) of an earlier model by Imai et al. (2001), to solve the dynamic discrete berth allocation problem. In Buhrkal et al. (2011), the SP model is able to solve all instances to optimality containing up to 60 vessels in computational time of less than 30 seconds.

Umang et al. (2013) demonstrate the superiority of the SP approach for a slightly more complex variant of the dynamic BAP with a hybrid berthing layout in the context of bulk terminals. In this study, the SP approach can solve instances up to 40 vessels within a computational time limit of 2 hours. However for some large instances with 40 vessels, the CPLEX solver runs out of memory, and to fix this issue, the planning horizon is partitioned into larger time buckets of 2 hours to make the method affordable.

In a few other studies, authors propose exact algorithms to solve the BAP in integration with other optimization problems. For example, Vacca et al. (2013) study the integrated berth allocation and quay crane assignment problem, and propose a branch-and-price algorithm based on Dantzig-Wolfe decomposition of the original MIP formulation. The maximum problem size in this study contains 20 vessels and 5 berths over a planning horizon of one week. Turkogullari et al. (2013) propose a cutting plane algorithm to derive an optimal solution for the berth allocation quay crane assignment and scheduling problem (BACASP) from the optimal solution of the berth allocation and quay crane assignment problem (BACAP). The authors are able to solve large instances of BACASP containing up to 60 vessels in few hours when multiple cuts are applied. Closely related to our work, Robenek et al. (2014) propose a branch-and-price approach to solve the dynamic hybrid BAP in integration with the yard assignment problem in bulk terminals. The authors apply column generation to the integrated model re-formulated as a set-partitioning problem, and show that instances containing up to 40 vessels can be solved in computational time of few hours.

In this research, our objective is to present a general methodological framework based on column generation to solve large problem instances of the BAP. We focus on addressing the limitations of the SP method when it is not affordable to generate all the columns apriori and

feed them as input to the optimization model. We propose a dynamic column generation scheme in which the SP model is the master problem and test our approach on a discrete dynamic BAP model. Results of computational experiments on artificial instances containing up to 120 vessels and 10 berths show that the proposed methodology can easily solve very large problem instances to optimality in small computational time of few minutes.

3 Problem statement

The column generation based approach proposed in this paper can be easily adapted to solve any variant of the BAP based on spatial attributes such as the berthing layout and draft restrictions, and temporal attributes such as vessel arrival times and handling times. In this study, to test and validate our approach, we consider a berth allocation model with dynamic vessel arrivals where a vessel can be berthed only after its arrival at the port. The berthing layout is discrete implying that a given vessel can occupy exactly one discrete berth section at a given time. The handling time of a vessel is assumed to depend on the berthing location of the vessel along the quay. The planning horizon is considered to be fixed and partitioned into discrete time buckets. A single variable co-ordinate system is defined along the quay, with the origin at the left extreme of the quay, as shown in Figure 1.

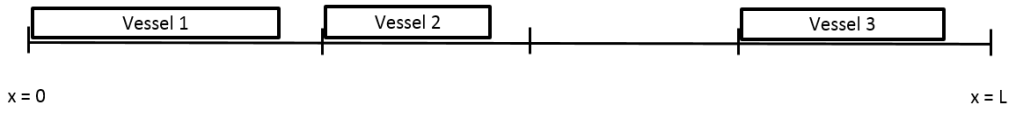

Figure 1: *Discrete berthing layout, showing a feasible assignment of vessels to berth sections at a single point in time*

For a given set of vessels N and set of berths M, the problem can be formulated as a mixed integer linear program (MILP) as described in this section. The decision variables in the problem include

- $m_i \geq 0$, starting time of processing of vessel $i \in N$;
- x_{ik} binary, equals 1 if vessel $i \in N$ is berthed at section $k \in M$, 0 otherwise;
- y_{ij} binary, equals 1 if vessel $i \in N$ is berthed to the left of vessel $j \in N$ without any overlapping in space, 0 otherwise;
- z_{ij} binary, equals 1 if handling of vessel $i \in N$ finishes before the start of handling of vessel $j \in N$, 0 otherwise

The objective is to minimize the sum of the service times of all vessels berthing at the port, mathematically expressed as

$$\min \sum_{i \in N} (m_i - a_i + \sum_{k \in M} (h_{ik} x_{ik})) \tag{1}$$

$$\tag{2}$$

where a_i is the arrival time of vessel $i \in N$ and h_{ik} is the handling time of vessel $i \in N$ berthed at section $k \in M$.

The dynamic arrival constraints can be simply stated as

$$m_i - a_i \geq 0 \quad \forall i \in N \tag{3}$$

In any feasible solution, each vessel should occupy exactly one berth, expressed as follows

$$\sum_{k \in M} x_{ik} = 1 \quad \forall i \in N \tag{4}$$

Let l_i and d_i be the length and draft of vessel $i \in N$ respectively, and L_k and D_k be the length and draft of berth $k \in M$ respectively. Constraints (5)-(6) ensure that length and draft of any vessel do not exceed the length and draft of it's occupied berth.

$$\sum_{k \in M} (L_k - l_i)\, x_{ik} \geq 0 \quad \forall i \in N \tag{5}$$

$$\sum_{k \in M} (D_k - d_i)\, x_{ik} \geq 0 \quad \forall i \in N \tag{6}$$

Finally to ensure that two vessels do not occupy the same berth at any given time, we have constraints (7) that define variables y_{ij}, constraints (8) that define variables z_{ij}, and the non-overlapping constraints (9) that link variables y_{ij} to z_{ij}.

$$\sum_{k \epsilon M} (k x_{jk}) + B(1 - y_{ij}) \geq \sum_{k \epsilon M} (k x_{ik}) + 1 \quad \forall i, j \in N, i \neq j \tag{7}$$

$$m_j + B(1 - z_{ij}) \geq m_i + \sum_{k \in M} (h_{ik} x_{ik}) \quad \forall i, j \in N, i \neq j \tag{8}$$

$$y_{ij} + y_{ji} + z_{ij} + z_{ji} \geq 1 \quad \forall i, j \in N, i \neq j \tag{9}$$

, where B is a large positive constant.

4 Methodology

In this section, we present the set partitioning (SP) method to solve realistic sized instances of the BAP, and propose a dynamic column generation based scheme to address the limitations of generating columns apriori for the SP approach. Note that the methods discussed in this section are easily extendable to any variant of the BAP.

4.1 Set partitioning model: Static column generation

As shown in previous research (Umang et al. (2013), Buhrkal et al. (2011)), the berth allocation problem with known arrival and handling times can be effectively modeled and solved as a set-partitioning (SP) problem for relatively large problem size. Since our primary goal in this paper is to address the limitations of the SP approach, we briefly describe the approach in this section. In the SP method, the set of all feasible single-vessel berthing assignments is generated *a priori* and is denoted by the set P. Note that a berthing assignment for a single vessel specifies the berth sections that will be occupied by the vessel, its berthing time, and its completion time (equal to the berthing time plus the handling time). The assignment matrix contains a column for each of the $|P|$ assignments, and is composed of upper submatrix A and lower submatrix B. Each column p in submatrix A has a single non-zero value, where row i contains the value one if the berthing assignment is for vessel $i \in N$. Submatrix B contains a single row for each (berth section, time bucket). Non-zero values in submatrix B are equal to one if the vessel berthing assignment specified by column p requires that the vessel occupies the (section, time) represented by the row.

To illustrate this idea, consider an example with two vessels, two discrete berth sections and three discrete time periods in the planning horizon with the input data shown in Tables (1)-(2). Vessel 1 cannot occupy berth 2 because of length restrictions and vessel 2 cannot be berthed before time t=1. The draft restrictions in this example are redundant since the depth of each section is greater than the draft of both the vessels. The resulting assignment matrix comprising of two feasible assignments of vessel 1 and three feasible assignments of vessel 2 is shown in Table 3. The first column represents the berthing assignment of vessel 1 to berth 1 from time 0-2, the third column represents the berthing assignment of vessel 2 to berth 1 from time 1-3 and so on.

Table 1: Input data for vessels for a toy example

Vessel	Arrival Time	Length (m)	Draft (m)	Handling Time
Vessel 1	0	180	6	2
Vessel 2	1	80	6	2

Table 2: Input data for berth sections for a toy example

Berth	Length (m)	Draft (m)
Berth 1	200	8
Berth 2	100	8

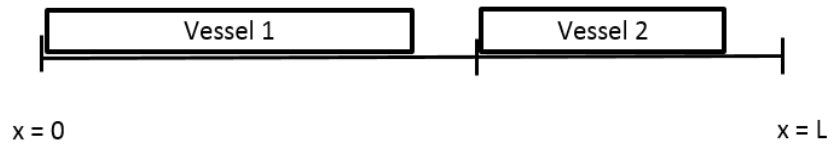

Figure 2: *Toy example of set partitioning to solve the BAP with* $|N| = 2$, $|M| = 2$ *and* $|H| = 3$

Table 3: Assignment matrix for a toy example of SP model

Vessel 1	1	1	0	0
Vessel 2	0	0	1	1
Berth 1, Time 0-1	1	0	0	0
Berth 1, Time 1-2	1	1	1	0
Berth 1, Time 2-3	0	1	1	0
Berth 2, Time 0-1	0	0	0	0
Berth 2, Time 1-2	0	0	0	1
Berth 2, Time 2-3	0	0	0	1

We assume the following input data to be available for the SP model:

$H =$ set of discrete time intervals in the planning horizon
$P =$ set of feasible assignments
$t =$ $1, ..., |H|$ discrete time intervals in the planning horizon
$p =$ $1, ..., |P|$ feasible assignments
$d_p =$ delay associated with assignment p
$h_p =$ handling time associated with assignment p

The assignment matrix coefficients are defined as follows.

$$A_{ip} = \begin{cases} 1 & \text{if assignment } p \text{ represents a feasible assignment for vessel } i; \\ 0 & \text{otherwise.} \end{cases}$$

$$b_p^{kt} = \begin{cases} 1 & \text{if section } k \text{ is occupied at time } t \text{ in assignment } p; \\ 0 & \text{otherwise.} \end{cases}$$

There is only a single decision variable in the SP model for selection of feasible assignments in the optimal solution which is defined as follows.

$$\lambda_p = \begin{cases} 1 & \text{if assignment } p \text{ is part of the optimal solution;} \\ 0 & \text{otherwise.} \end{cases}$$

The SP model is formulated as shown below:

$$\min \sum_p (d_p \lambda_p + h_p \lambda_p) \tag{10}$$

$$\text{s.t.} \sum_p (A_{ip} \lambda_p) = 1 \qquad \forall i \in N \tag{11}$$

$$\sum_p (b_p^{kt} \lambda_p) \leq 1 \quad \forall k \in M, \forall t \in H \tag{12}$$

$$\lambda_p \in \{0, 1\} \qquad \forall p \in P \tag{13}$$

The objective (10) is to minimize the total service cost of the vessels berthing at the port, which includes the total berthing delays and the total handling cost of the vessels. Constraints (11) ensure that each vessel has exactly one feasible assignment in the optimal solution. Constraints (12) ensure that a given section at a given time is occupied by at most one vessel.

A major limitation of the set partitioning approach is the explosion in the number of variables and constraints with increase in problem size, because of which the method may be very slow and the solver can run out of memory when the number of feasible assignments is too large as determined by the problem size defined by the number of vessels and number of sections along the quay, the length of the planning horizon, the redundancy of one or more constraints such as the draft restrictions etc. In some such cases it is possible to make the method affordable at the risk of losing optimality by partitioning the planning horizon into fewer discrete time buckets of larger size. The root cause of this problem is that the assignments (columns) are generated apriori and fed as an input to the model. To address this shortcoming of the model, we propose a dynamic column generation (CG) scheme in the following section.

4.2 Dynamic column generation

We now propose a novel scheme to solve the the SP formulation (10)-(13), in which the feasible assignments (columns) are generated in a dynamic way, to reduce the computational load on the solver. In the dynamic scheme, the SP formulation is referred to as the *restricted master problem*. It is called restricted since the active pool of columns, denoted by Ω, does not contain all possible feasible assignments to the problem instance, but only a subset of the assignments.

We start with an initial feasible solution obtained from a first-come-first-served ordering of the vessels based on the vessel arrival times. The algorithm used to generate an initial feasible solution is described by Algorithm 1.

Algorithm 1 Heuristic to obtain an initial feasible solution for the dynamic CG

Require: Set N of vessels sorted by arrival times, set M of berths
  for $i = 1 \rightarrow N$ **do**
    if BerthIsAvailable **then**
      Assign vessel to berth with lowest handling time for vessel i
    end if
    if BerthIsNotAvailable **then**
      Assign vessel to berth with earliest completion time for vessel i
    end if
  end for

The initial solution comprising of $|N|$ columns is added to Ω, so that Ω contains $|N|$ active columns at the start of the column generation process. We relax the integrality constraints (13) in the SP formulation, to obtain the the LP relaxation of the restricted master problem as shown below.

$$\min \sum_{p} (d_p \lambda_p + h_p \lambda_p) \tag{14}$$

$$\text{s.t.} \sum_{p} (A_{ip} \lambda_p) = 1 \qquad \forall i \in N \tag{15}$$

$$\sum_{p} (b_p^{kt} \lambda_p) \leq 1 \quad \forall k \in M, \forall t \in H \tag{16}$$

$$\lambda_p \in [0, 1] \qquad \forall p \in \Omega \tag{17}$$

The LP relaxation is solved and the dual variable values corresponding to constraints (15)-(16) are calculated. In each iteration of the CG algorithm, $|N|$ subproblems are solved, one for each vessel in the problem instance. The idea is to price out columns with negative reduced cost that can potentially improve the solution, and add them to the current pool of active columns Ω. We now describe the subproblem formulation in detail. Note that since the subproblem is solved separately for each vessel, the index i is dropped from all parameters and decision variables. The following input data is used to solve the subproblem.

$\pi =$ the dual variables corresponding to constraints (15)
$\tau_{kt} =$ the dual variables corresponding to constraints (16)
$a =$ the arrival time of the vessel
$l =$ the length of the vessel
$d =$ the draft of the vessel
$L_k =$ length of berth k
$D_k =$ draft of berth k
$h_k =$ handling time at berth k
$B =$ large positive constant

The decision variables in the subproblem are

m ≥ 0, starting time of processing of the vessel;
x_k binary, equals 1 if berth $k \in M$ is occupied by the vessel, 0 otherwise;
θ_t binary equals 1 if the vessel is served at time t, 0 otherwise;
σ_{kt} binary equals 1 if berth k is occupied at time t, 0 otherwise;

The subproblem is formulated as follows

$$\min(m - a + \sum_{k \in M}(h_k x_k) - \pi - \sum_{t \in H}\sum_{k \in M}(\tau_{kt}\sigma_{kt})) \tag{18}$$

$$m - a \geq 0 \tag{19}$$

$$\sum_{k \in M}(L_k - l)x_k \geq 0 \tag{20}$$

$$\sum_{k \in M}(D_k - d)x_k \geq 0 \tag{21}$$

$$\sum_{k \in M} x_k = 1 \tag{22}$$

$$\sum_{t \in H} \theta_t = \sum_{k \in M}(h_k x_k) \tag{23}$$

$$t + B(1 - \theta_t) \geq m \qquad \forall t \in H \tag{24}$$

$$t \leq m + \sum_{k \in M}(h_k x_k) + B(1 - \theta_t) - 1 \qquad \forall t \in H \tag{25}$$

$$\sigma_{kt} \geq x_k + \theta_t - 1 \quad \forall t \in H, \forall k \in M \tag{26}$$

$$\sigma_{kt} \leq x_k \quad \forall t \in H, \forall k \in M \tag{27}$$

$$\sigma_{kt} \leq \theta_t \quad \forall t \in H, \forall k \in M \tag{28}$$

$$x_k \in \{0, 1\} \qquad \forall k \in M \tag{29}$$

$$\theta_t \in \{0, 1\} \qquad \forall t \in H \tag{30}$$

$$\sigma_{kt} \in \{0, 1\} \quad \forall k \in M, \forall t \in H \tag{31}$$

The subproblem (18)-(31) can be easily solved using a polynomial time heuristic or directly using an optimization solver. When solving the subproblem for a given vessel $i \in N$, the objective (18) is to price out the most negative reduced cost column for that vessel. Constraints (19)-(22) are as described earlier for the original formulation in Section 3. Constraints (23)-(25) are used to mathematically define the variables θ_t. The decision variables σ_{kt} are linked to variables x_k and θ_t using (26)-(28).

The |N| columns that are priced out in each iteration of the column generation, one for each vessel, are added to the active pool of columns Ω, and the restricted master problem (14)-(17) is re-solved. The dual variable values are obtained, the |N| subproblems are solved and the process is continued iteratively until there are no negative reduced cost columns for any vessel. The main steps in the proposed dynamic column generation algorithm are shown schematically in Figure 3 below.

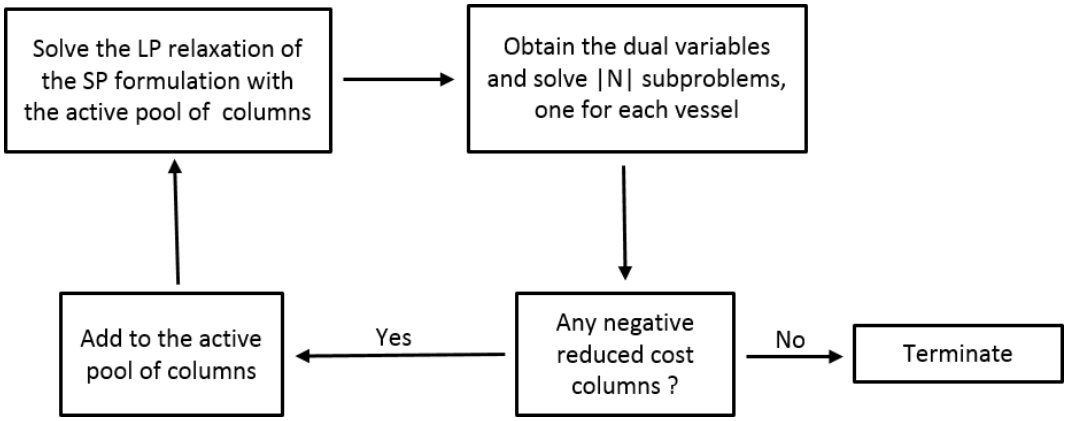

Figure 3: *The dynamic column generation scheme for the BAP*

Obtaining an integer solution

The solution of the LP relaxation obtained after the convergence of the column generation algorithm is typically fractional, and is a *lower bound* to the original problem. To obtain an integer solution, we use a simple approach, that based on our numerical results, works extremely well for the problem studied in this paper. Once the column generation terminates, the integrality constraints (13) are reinforced and the problem is re-solved using the active pool of columns Ω at the end of the column generation process. The integer solution obtained is a valid *upper bound* to the problem we are interested in solving. If the *gap* between the *upper bound* and the *lower bound* is zero, then the upper bound is also the optimal solution. If the *gap* is non-zero, the optimal solution can be determined through a branch-and-bound (B&B) algorithm, where column generation needs to be applied at each node of the B&B tree.

5 Computational study

In this section, we compare the different formulations presented earlier in the paper. All algorithms are implemented in JAVA programming language. The computation experiments were run on an Intel(R) Core(TM) i7 -3770 CPU(3.40 GHz) processor and used a 64-bit version of CPLEX 12.6.

5.1 Generation of test instances

We generate a test bed of 56 instances with 7 different instance sizes of 8 instances each. The instance size is defined by the number of vessels in the instance, the largest instance size contains |N| = 120 vessels. The vessel lengths lie in the range 80 to 200 meters and the vessel drafts vary between 6 to 12 meters. The berthing layout used in the model is given in Table 4. The number of berths in all instances is |M|=10 berths, the berth length varies between 120 to 220 meters, and the berth depth is between 8 to 15 meters. For each instance size, there are four congested instances in which all vessel arrivals are within the time window t=0 and t=|N|, and four mildly congested instances in which vessel arrivals are within t=0 and t=2*|N|. The handling times of the vessels are dependent on the berthing position of the vessel and vary between 6 to 20 hours.

Table 4: Berthing layout used in the model

Berth	Length (m)	Depth (m)
1	217	15
2	215	15
3	189	14
4	191	8
5	136	13
6	174	14
7	178	8
8	178	12
9	182	14
10	213	8

5.2 Results

The results from the computation experiments are shown in Tables 5-6. The following inferences can be drawn from the results' tables.

- The MIP (1)-(9) is able to solve all instances containing up to 25 vessels in few seconds, except instance B4 for which the computation time is a bit higher. All mildly congested instances belonging to instance sets C, D and E, except instance D6, could be solved to optimality within the CPLEX time limit of 1800 seconds. However not even one congested instance could be solved within this time limit. For large problem instances in sets F and G containing 100 and 120 vessels respectively, the optimality gaps are very large for the congested instances, and much smaller for the mildly congested instances.

- The SP model clearly outperforms the MIP and is able to solve all instances containing up to 80 vessels within few seconds. However for larger sized instances containing 100 and more vessels, the CPLEX solver runs out of memory owing to an explosion in the number of feasible assignments in those problem instances.

- For the dynamic CG algorithm, we report the number of iterations and the number of columns generated in the column generation process as indicated by #iter and #columns respectively, the solution of the LP relaxation indicated by lb, the integer feasible solution obtained at the end of the column generation process indicated by ub, the optimality gap, and the computation time in minutes and seconds. The results look very promising, the algorithm is able to solve all instances to optimality in computation time of few minutes. The highest computation time is only a little over 15 minutes for the congested instance G4. It is interesting to note that the optimality gap is zero or very close to zero for all instances. This means that for each instance, the pool of active columns at the end of the CG process contains the set of feasible columns that correspond to the optimal (or near optimal) solution, and it is not required to implement branch and bound to further close the gap between the lower bound and the upper bound.

The above observations clearly establish the superiority of the proposed dynamic column generation algorithm to solve large problem instances of the BAP in small computation time. Note that while for the berth allocation model considered in this study the SP approach is able to solve instances containing up to 80 vessels, the solver can run out of memory for a smaller number of vessels for other berthing layouts, for a higher number of berths, for a longer planning horizon and/or when one or more constraints such as the draft restrictions are redundant. For example, as shown in Umang et al. (2013), the SP model runs out of memory for instances containing 40 vessels and 30 berthing sections for the dynamic hybrid BAP model considered in their paper.

Furthermore it is clear from the computational results that the time complexity of the BAP increases with both the problem size as given by the number of vessels in this study, and the level of congestion as determined by the inter-arrival times for given problem size.

Instance	Congestion	MIP			SP		Dynamic CG					
|N|=10, |M|=10		obj	gap	time(s)	obj	time(s)(H= 120)	# iter	# columns	lb	ub	gap	time(H= 120)
A1	Yes	88	0.00%	0.403	88	1.105	9	34	88	88	0.00%	7s
A2	Yes	81	1.23%	0.276	81	0.692	4	20	81	81	0.00%	3s
A3	Yes	84	1.00%	0.235	84	0.854	8	21	84	84	0.00%	6s
A4	Yes	79	0.00%	0.508	79	0.806	7	40	79	79	0.00%	7s
A5	Mild	84	0.00%	0.298	84	0.716	5	26	84	84	0.00%	4s
A6	Mild	79	0.00%	0.198	79	0.730	2	11	79	79	0.00%	1s
A7	Mild	80	1.25%	0.234	80	0.755	6	21	80	80	0.00%	4s
A8	Mild	75	0.00%	0.340	75	0.741	5	18	75	75	0.00%	3s
|N|=25, |M|=10		obj	gap	time(s)	obj	time(s)(H= 120)	# iter	# columns	lb	ub	gap	time(H= 120)
B1	Yes	233	0.00%	8.008	233	2.027	9	121	233	233	0.00%	23s
B2	Yes	204	0.00%	2.110	204	2.139	7	116	204	204	0.00%	18s
B3	Yes	207	0.00%	6.785	207	2.229	10	134	207	207	0.00%	28s
B4	Yes	244	0.00%	1708.224	244	2.078	13	183	244	244	0.00%	39s
B5	Mild	213	0.00%	0.630	213	1.989	6	49	213	213	0.00%	9s
B6	Mild	186	0.00%	0.803	186	2.073	5	53	186	186	0.00%	7s
B7	Mild	200	0.00%	1.266	200	2.009	8	81	200	200	0.00%	15s
B8	Mild	208	0.00%	1.776	208	1.921	8	65	208	208	0.00%	12s
|N|=40, |M|=10		obj	gap	time[a](s)	obj	time(s)(H= 168)	# iter	# columns	lb	ub	gap	time(H= 168)
C1	Yes	420	13.67%	-	411	7.418	16	393	410.33	411	0.16%	2m11s
C2	Yes	355	2.33%	-	355	6.639	11	277	355	355	0.00%	1m23s
C3	Yes	415	6.12%	-	411	4.719	14	365	409	409	0.00%	1m37m
C4	Yes	356	3.60%	-	355	6.711	13	275	355	355	0.00%	1m38s
C5	Mild	332	0.00%	6.108	332	4.702	6	122	332	332	0.00%	22s
C6	Mild	330	0.00%	13.698	330	4.961	10	126	330	330	0.00%	37s
C7	Mild	356	0.00%	8.536	356	4.217	11	150	356	356	0.00%	40s
C8	Mild	315	0.00%	3.252	315	6.158	9	102	315	315	0.00%	32s
|N|=60, |M|=10		obj	gap	time[a](s)	obj	time(s)(H= 168)	# iter	# columns	lb	ub	gap	time(H= 168)
D1	Yes	602	14.27%	-	574	11.747	15	518	572.25	577	0.83%	2m57s
D2	Yes	589	14.25%	-	565	9.54	16	560	564.6	565	0.07%	3m7s
D3	Yes	709	25.01%	-	648	10.547	20	630	645.8	649	0.50%	3m53s
D4	Yes	561	8.49%	-	554	10.107	16	428	554	554	0.00%	2m49s
D5	Mild	505	0.01%	56.381	505	9.056	8	200	505	505	0.00%	44s
D6	Mild	515	3.80%	-	508	7.942	13	247	508	508	0.00%	1m7s
D7	Mild	518	0.01%	424.157	518	8.517	10	242	518	518	0.00%	1m3s
D8	Mild	498	0.00%	34.767	498	9.204	8	186	498	498	0.00%	45s

Table 5: Computational comparison between the algorithms for instances A, B, C and D

[a]"-" indicates that the CPLEX time limit of 1800 seconds was reached.
[b]"+" indicates that the CPLEX solver ran out of memory.

Instance	Congestion	MIP			SP		Dynamic CG					
|N|=80, |M|=10		obj	gap	time[a](s)	obj	time(s)(H= 168)	# iter	# columns	lb	ub	gap	time(H= 168)
E1	Yes	956	27.04%	-	867	27.251	20	1012	858.85	870	1.30%	5m41s
E2	Yes	900	24.17%	-	801	13.511	22	830	793.75	798	0.54%	5m31s
E3	Yes	836	16.46%	-	791	24.143	18	807	788.75	791	0.29%	4m28s
E4	Yes	830	15.63%	-	797	24.036	15	690	794.5	796	0.19%	3m29s
E5	Mild	666	0.01%	620.353	666	9.454	12	292	665.5	666	0.08%	1m9s
E6	Mild	663	0.01%	760.616	663	10.192	13	373	663	663	0.00%	1m22s
E7	Mild	679	0.01%	77.305	679	9.418	15	229	679	679	0.00%	1m25s
E8	Mild	686	0.01%	206.189	686	8.603	10	229	686	686	0.00%	53s
|N|=100, |M|=10		obj	gap	time[a](s)	obj [b]	time[b](s)(H= 240)	# iter	# columns	lb	ub	gap	time(H= 240)
F1	Yes	1208	28.22%	-	+	+	27	1066	1023.97	1031	0.69%	12m33s
F2	Yes	1160	26.01%	-	+	+	22	954	1012	1016	0.40%	9m17s
F3	Yes	1377	35.70%	-	+	+	17	1169	1014	1016	0.20%	7m33s
F4	Yes	1025	20.20%	-	+	+	19	951	926.9	927	0.01%	7m57s
F5	Mild	850	0.96%	-	+	+	9	366	849	849	0.00%	2m1s
F6	Mild	857	2.64%	-	+	+	9	362	852	852	0.00%	1m55s
F7	Mild	919	6.04%	-	+	+	14	428	897	897	0.00%	2m53s
F8	Mild	793	0.34%	-	+	+	9	304	793	793	0.00%	1m38s
|N|=120, |M|=10		obj	gap	time[a](s)	obj [b]	time[b](s)(H= 264)	# iter	# columns	lb	ub	gap	time(H= 264)
G1	Yes	2565	59.43%	-	+	+	25	3339	1315.172	1321	0.44%	9m32s
G2	Yes	1667	39.15%	-	+	+	22	1410	1201.533	1206	0.37%	13m20s
G3	Yes	1569	32.33%	-	+	+	17	1234	1255.5	1257	0.12%	11m9s
G4	Yes	1681	42.34%	-	+	+	23	1383	1137.75	1146	0.73%	15m9s
G5	Mild	1016	1.04%	-	+	+	12	407	1014	1014	0.00%	3m18s
G6	Mild	994	1.51%	-	+	+	11	384	988	988	0.00%	2m44s
G7	Mild	1053	3.59%	-	+	+	10	409	1040	1040	0.00%	2m36s
G8	Mild	935	0.37%	-	+	+	10	306	935	935	0.00%	2m34s

Table 6: Computational comparison between the algorithms for instances E, F and G

[a] "-" indicates that the CPLEX time limit of 1800 seconds was reached.
[b] "+" indicates that the CPLEX solver ran out of memory.

6 Conclusions and future work

In this paper, we have proposed a dynamic column generation framework to solve the berth allocation problem (BAP). The most important contribution of this study is to demonstrate the superiority of the proposed algorithm in solving large problem instances of BAP to optimality in computation time of only a few minutes. We were successfully able to solve instances containing up to 120 vessels and 10 berths for the dynamic discrete berth allocation model considered in this paper. Interestingly the optimality gap at the end of the proposed dynamic column generation method is zero or almost zero for all instances. Thus the algorithm returns the optimal or near-optimal integer solution, and no further branching is required. Furthermore the approach can be easily adapted to solve other variants of the BAP based on different spatial and temporal attributes.

In previous research (Umang et al. (2013), Buhrkal et al. (2011)), the set partitioning (SP) method was shown to outperform other state-of-the-art algorithms to solve the BAP. However the SP model runs out of memory when the number of feasible assignments is too large as determined by the problem size given by the number of vessels and berths, length of the planning horizon, redundancy of draft restrictions etc. For example, as shown in Umang et al. (2013), the SP model runs out of memory for $|N|=40$ vessels and $|M|=30$ berths for the dynamic hybrid berth allocation problem considered in their paper. Our approach addresses the limitations of the SP model in which the columns are generated apriori and fed as an input to the optimization model.

As part of future work, the proposed column generation framework should be tested on other more complex variants of the BAP based on different berthing layouts and other attributes. For problem instances where the computation time is large, acceleration techniques based on adding multiple negative reduced cost columns per iteration of the column generation, dynamic constraint aggregation, dual stabilization and heuristic pricing should be investigated.

References

Bierwirth, C. and Meisel, F. (2010). A survey of berth allocation and quay crane scheduling problems in container terminals, *European Journal of Operational Research* **202**(3): 615–627.

Bierwirth, C. and Meisel, F. (2015). A follow-up survey of berth allocation and quay crane scheduling problems in container terminals, *European Journal of Operational Research* **244**: 675–689.

Buhrkal, K., Zuglian, S., Ropke, S., Larsen, J. and Lusby, R. (2011). Models for the discrete berth allocation problem: A computational comparison, *Transportation Research Part E* **47**(4): 461 – 473.

Cordeau, J. F., Laporte, G., Legato, P. and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem, *Transportation Science* **39**(4): 526–538.

Imai, A., Nagaiwa, K. and Chan, W. T. (1997). Efficient planning of berth allocation for container terminals in Asia, *Journal of Advanced Transportation* **31**(1): 75–94.

Imai, A., Nishimura, E. and Papadimitriou, S. (2001). The dynamic berth allocation problem for a container port, *Transportation Research Part B* **35**(4): 401–417.

Imai, A., Nishimura, E. and Papadimitriou, S. (2003). Berth allocation with service priority, *Transportation Research Part B* **37**(5): 437–457.

Imai, A., Sun, X., Nishimura, E. and Papadimitriou, S. (2005). Berth allocation in a container port: using a continuous location space approach, *Transportation Research Part B* **39**(3): 199–221.

Kim, K. H. and Moon, K. C. (2003). Berth scheduling by simulated annealing, *Transportation Research Part B* **37**(6): 541–560.

Monaco, M. F. and Sammarra, M. (2007). The berth allocation problem: a strong formulation solved by a lagrangean approach, *Transportation Science* **41**(2): 265–280.

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

Stahlbock, R. and Voss, S. (2008). Operations research at container terminals: a literature update, *OR Spectrum* **30**(1): 1–52.

Steenken, D., Voss, S. and Stahlbock, R. (2004). Container terminal operation and operations research - a classification and literature review, *OR Spectrum* **26**(1): 3–49.

Turkogullari, Y., Taskin, Z., Aras, N. and Altinel, I. (2013). Optimal berth allocation and time-invariant quay crane assignment in container terminals, *European Journal of Operational Research* **235**: 88–101.

Umang, N., Bierlaire, M. and Vacca, I. (2013). Exact and heuristic methods to solve the berth allocation problem in bulk ports, *Transportation Research Part E* **54**: 14–31.

Vacca, I., Salani, M. and Bierlaire, M. (2013). An exact algorithm for the integrated planning of berth allocation and quay crane assignment, *Transportation Science* **47**: 148–161.