A Note on Formulations of Static and Dynamic Berth Allocation Problems

Pierre Hansen
Ceyda Oğuz

G–2003–30
May 2003
Revised: November 2003

Les textes publiés dans la série des rapports de recherche HEC n’engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d’une subvention du Fonds québécois de la recherche sur la nature et les technologies.
A Note on Formulations of Static and Dynamic Berth Allocation Problems

Pierre Hansen
GERAD and Department of Quantitative Methods in Management
HEC Montréal, Canada

Ceyda Oğuz
Department of Logistics
The Hong Kong Polytechnic University
Kowloon, Hong Kong SAR

May, 2003
Revised: September, 2003

Les Cahiers du GERAD
G-2003-30

Copyright © 2003 GERAD
Abstract

The berth allocation problem is to allocate berths (i.e., sections of the quayside) to ships arriving in a container port in order to minimize the sum of their waiting and cargo handling times. In the static case, ships are assumed to arrive before the berths become available; in the dynamic case they can arrive before or after. We discuss recent models for both problems, correct one formulation, and provide a compact reformulation for the dynamic case. The computational results for problems that are moderate in size but realistic, with up to 10 berths and 50 ships, using both formulations and CPLEX, are reported.

Keywords: berth allocation, berthing, total completion time, unrelated parallel machines, release date.

Résumé

Le problème d’ordonnancement de l’accostage consiste à allouer des sections de quais aux bateaux arrivants en un port de conteneurs de façon à minimiser la somme des temps d’attente, de déchargement et de chargement. Pour le cas statique, les bateaux sont supposés arriver avant que les sections de quais ne deviennent disponibles; dans le cas dynamique ils peuvent arriver avant ou après. On discute des modèles récents pour les deux problèmes, corrige une formulation et présente une reformulation compacte pour le cas dynamique. Des résultats de calculs sont présentés pour des instances de taille modérée mais néanmoins réalisable, avec jusqu’à 10 sections de quais et 50 bateaux; on utilise les deux formulations et CPLEX.

Mots clé : ordonnancement, accostage, temps total d’exécution, machines parallèles non corréllées, date de début.
1 Introduction

The berth allocation problem is to allocate berths (i.e., sections of the quayside) to ships arriving in a container port in order to minimize the sum of their waiting and cargo handling times. As container transportation grows rapidly, and ships spend a large amount of their time in port at the cost of several thousand US dollars per hour (Peterkovsky and Daganzo, 1990; Ward, 2002), this problem is of major importance in port operations (Meersmans and Dekker, 2001). It has recently been studied using mathematical programming and genetic search by Imai et al. (1997, 2001), Lim (1998), Nishimura et al. (2001), and Park and Kim (2002). With the exception of the subgradient method of this last paper, applied to small problems with up to seven ships, only heuristics are available for the dynamic case in which ships arrive before or after berths become available.

The paper is organized as follows. The model for the static berth allocation problem of Imai et al. (1997, 2001), where all ships are assumed to have reached the port before the berths become available, is recalled in the next section. A correction is then made in the expression of the objective function of Imai et al. (2001). An extension of the dynamic berth allocation problem (Imai et al., 2001) is discussed in Section 3. A compact reformulation is proposed in Section 4, together with some further extensions: taking into account safety constraints due to a ship’s length or draft, the due dates for the departures of some or all ships and a bounded planning horizon. Computational results for problems that are moderate in size but realistic, with up to 10 berths and 50 ships, using both formulations of the dynamic berth allocation problem and CPLEX, are presented in Section 5. Brief conclusions are given in the last section.

2 The static berth allocation problem

Let us assume that:

(a) A set $B$ of berths, indexed by $i = 1, 2, \ldots, I$, is available in a container port for receiving ships, from time $S_i$, $i = 1, 2, \ldots, I$, respectively onward;
(b) A set $V$ of ships, indexed by $j = 1, 2, \ldots, T$, gets to the port at arrival times $A_j$, $j = 1, 2, \ldots, T$, respectively, and may have to wait before being handled;
(c) Each berth can handle one ship at a time or remain idle for some time;
(d) Any ship $j$ can be handled at any berth $i$, with a given handling time $c_{ij}$, $i = 1, 2, \ldots, I$, $j = 1, 2, \ldots, T$, depending on both the ship and the berth;
(e) The total completion time, i.e., waiting and handling time for all ships is to be minimized;
(f) All ships arrive before any berth becomes available, i.e.,

$$\max_j A_j \leq \min_i S_i.$$  \hspace{1cm} (1)

Due to this last assumption, all berths will handle ships consecutively; i.e., without idle time between ships. If this assumption does not hold, the possible idle time must be taken into account and the dynamic berth allocation problem is obtained. Observe that
the model is short-term and deterministic. Moreover, fractional assignments are excluded; i.e., the uncommon practice of changing a ship’s berth while unloading or loading it is prohibited.

Consider, then, the reverse order in which ships \( j \) are handled at berth \( i \). To that effect, introduce binary variable \( x_{ijk} \), such that

\[
x_{ijk} = \begin{cases} 
1 & \text{if ship } j \text{ is the } k \text{th last to be handled at berth } i, \\
0 & \text{otherwise.}
\end{cases}
\]

At each berth, let us assign the reverse order at which ships will be handled with an index \( k \) such that \( k \in O = \{1, 2, \ldots, T\} \). Notice that this reverse order is unknown. It is then easy to see that the total completion time of all ships handled at berth \( i \) will be

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

where the first term corresponds to the waiting time for all ships handled there before berth \( i \) becomes available, and the second term is the sum for all such ships of the handling time and waiting time of the \( k - 1 \) ships still to be handled while the \( k \)th ship is being attended to. This is illustrated in Figure 1, where \( W_j \) denotes the waiting period and \( H_j \) the handling period for ship \( j \), \( j = 1, 2, \ldots, T \), and \( TCT \) denotes total completion times. Note that ships may be indexed in the non-decreasing order of their arrival times, but this is not mandatory.

Figure 1: Example of ships handled at berth \( i \) for SBAP: \( A_1 = -1, A_2 = -2, A_3 = -1, A_4 = -3, W_1 = 1, H_1 = 4, W_2 = 6, H_2 = 6, W_3 = 11, H_3 = 6, W_4 = 19, H_4 = 2, TCT=55. \)

The static berth allocation problem can be expressed as follows, which is done as part of a more general model in Imai et al. (1997).

\[
\text{(SBAP)} \quad \text{Minimize } \sum_{i \in B} \sum_{j \in V} \sum_{k \in O} (kc_{ij} + S_i - A_j)x_{ijk}
\]
subject to

\[ \sum_{i \in B} \sum_{k \in O} x_{ijk} = 1 \quad \forall j \in V, \quad (4) \]

\[ \sum_{j \in V} x_{ijk} \leq 1 \quad \forall i \in B, k \in O, \quad (5) \]

\[ x_{ijk} \in \{0, 1\} \quad \forall i \in B, j \in V, k \in O. \quad (6) \]

Observe that as the coefficients of the variables \( x_{ijk} \) in the objective function (3) decrease when \( k \) increases, the ships will be assigned to consecutive positions at each berth. Note also that ships will be processed at each berth in increasing order of the \( c_{ij} \) in the optimal solution. Replacing the pairs of indices \( i \) and \( k \) by the new indices \( n = 1, 2, \ldots, I \times T \), noting by \( N \) the resulting index set and setting

\[ D_{jn} = kc_{ij} + S_i - A_j \quad \forall j \in V, n = (i - 1)T + k \in N \quad \forall i \in B, k \in O \quad (7) \]

gives the new expression

\[ \text{(SBAP')} \quad \text{Minimize} \sum_{j \in V} \sum_{n \in N} D_{jn} x_{jn} \quad (8) \]

subject to

\[ \sum_{n \in N} x_{jn} = 1 \quad \forall j \in V, \quad (9) \]

\[ \sum_{j \in V} x_{jn} \leq 1 \quad \forall n \in N, \quad (10) \]

\[ x_{jn} \in \{0, 1\} \quad \forall j \in V, n \in N, \quad (11) \]

which is a two-dimensional assignment problem and easily solved by for example, the polynomial algorithm of Jonker and Volgenant (1987) or the “auction” algorithm of Bertsekas (1992).

As already noted by Imai et al. (1997), SBAP is a particular case of a known scheduling problem; i.e., minimizing the mean completion time for independent tasks on unrelated parallel machines. A solution similar to the one given above is presented in Bruno et al. (1974).

In Imai et al. (2001) a variant of the SBAP model is proposed: instead of interpreting variables \( x_{ijk} \) equal to 1 as indicating that the ship \( j \) is handled at berth \( i \) in the \( k \)th last position, they are considered to indicate that this ship is handled as the \( k \)th ship at berth \( i \). We will use \( x_{ijk}' \) in this case.

The following model is presented:

\[ \text{Minimize} \sum_{i \in B} \sum_{j \in V} \sum_{k \in O} \{(T - k + 1)c_{ij} + S_i - A_j\} x_{ijk}' \quad (12) \]
subject to constraints (4) to (6) with \( x'_{ijk} \) instead of \( x_{ijk} \). This return to the order in time instead of the reverse order of handling ships at berths presents two difficulties. First, the number \( T_i \) of ships handled at berth \( i \) should be used in (12) instead of \( T \) for all \( i \), and is not known beforehand. It can be determined by using further equations, i.e.,

\[
T_i = \sum_{j \in V} \sum_{k \in O} x'_{ijk} \quad \forall i \in B,
\]

but then the model becomes non-linear in objective.

Second, while in model (3)–(6) the coefficients of variables \( x_{ijk} \) increase with \( k \) for a fixed \( i \) and \( j \), which ensures that a ship cannot be handled at a given berth in the \( k \)th last position unless another ship is handled in the \((k-1)\)st position at that berth, in model (12), (4)–(6) the coefficients of variables \( x'_{ijk} \) decrease when \( k \) increases. This implies that constraints (4) to (6) are not sufficient to ensure that a ship will not be handled at a given berth in the \( k \)th first position while no ship is handled in the \((k-1)\)st first position at that berth in the solution. Additional constraints are then needed and can be written as

\[
\sum_{j \in V} x'_{ijk} \leq \sum_{j \in V} x'_{ij,k-1} \quad \forall i \in B, k \in O \setminus \{1\}.
\]

Then if no ship is handled in the \((k-1)\)st position at berth \( i \), the right-hand side is equal to zero, which implies the left-hand side is also null, and that no ship is handled in the \( k \)th position at that berth. Again, it is possible to reduce the problem to a two-dimensional one. Let \( n \) be defined as above. Then, taking

\[
D'_{jk} = (T - k + 1)c_{ij} + S_i - A_j \quad \forall j \in V, n = (i - 1)T + k \in N
\]

one gets

\[
(\text{SBAP}''') \quad \text{Minimize} \sum_{j \in V} \sum_{n \in N} D'_{jn} x'_{jn}
\]

subject to

\[
\sum_{n \in N} x'_{jn} = 1 \quad \forall j \in V,
\]

\[
\sum_{j \in V} x'_{jn} \leq 1 \quad \forall n \in N,
\]

\[
\sum_{j \in V} x'_{jn} \leq \sum_{j \in V} x'_{jn-1} \quad \forall n \in N, n \mod (T) \notin \{1\},
\]

\[
x'_{jn} \in \{0, 1\} \quad \forall j \in V, n \in N.
\]

where \( n \mod (T) \) denotes the remainder of \( n \) after division by \( T \).

However, the first difficulty remains. It is not innocuous, as it may affect the optimal solution and not only its value. To illustrate, consider the following three ships and two
Table 1: Handling times

<table>
<thead>
<tr>
<th></th>
<th>Ship</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berth</td>
<td>1 2 3</td>
<td>10 10 10</td>
<td>11 11 11</td>
<td></td>
</tr>
</tbody>
</table>

berths problem, with $S_i = 0$ for $i = 1, 2$ and $A_j = 0$ for $j = 1, 2, 3$. Handling times are given in Table 1.

Coefficients $D_{jn}$ of model (8)–(11) are given in Table 2. The optimal solution will be $x_{11}^* = x_{12}^* = x_{31}^* = 1$, and the other $x_{ijk}^* = 0$ (or equivalently $x_{11}^* = x_{22}^* = x_{34}^* = 1$, and the other $x_{jn}^* = 0$), with a total completion time of $10 + 2 \times 10 + 11 = 41$.

Table 2: Coefficients $D_{jn}$ of variables $x_{jn}$ in model (8)–(11)

<table>
<thead>
<tr>
<th>Berth</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ship</td>
<td>1</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
</tbody>
</table>

Coefficients $D'_{jn}$ of model (16)–(20) are given in Table 3. With this model, the optimal solution will be $x_{11}^{*'} = x_{12}^{*'} = x_{33}^{*'} = 1$, and the other $x_{ijk}^{*'} = 0$ (or equivalently $x_{11}^{*'} = x_{22}^{*'} = x_{33}^{*'} = 1$, and the other $x_{jn}^{*'} = 0$). The value of this solution is $3 \times 10 + 2 \times 10 + 10 = 60$, which is not optimal.

Table 3: Coefficients $D'_{jn}$ of variables $x_{jn}'$ in model (16)–(20)

<table>
<thead>
<tr>
<th>Berth</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ship</td>
<td>1</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>33</td>
<td>22</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>33</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>33</td>
<td>22</td>
<td>11</td>
</tr>
</tbody>
</table>

3 The dynamic berth allocation problem

If assumption (f) of the SBAP is relaxed, i.e., some ships may arrive after some or all berths become available, the dynamic berth allocation problem (DBAP) arises. This problem is NP-hard even if there is a single berth. Indeed, it then reduces to minimizing the total
completion time with release dates on a single machine. This last problem has been shown to be NP-hard by Lenstra et al. (1977). As a consequence, the first come, first served rule, which may be adopted for reasons of fairness, i.e., to avoid overpassing between ships, is not optimal for the criterion of total completion time (contrary to what was stated in a recent survey on the transshipment of containers, by Vis and de Koster, 2003, p3). This is easily seen with the following one berth, two ships example: Let $A_1 = 0$, $A_2 = 1$, $c_11 = 10$, $c_{12} = 1$ and $S_1 = 0$. If ship 1 is handled first, from $s_{11} = 0$ onwards, ship 2 is handled from $s_{12} = 10$, and the total completion time is $10 + 9 + 1 = 20$. If the second ship is handled first at $s_{11} = 1$, and the first one at $s_{12} = 2$, the total completion time is $1 + 2 + 10 = 13$ (see Figure 2). This example also shows another counter-intuitive fact: even if a ship has already arrived at the port, it may be better to keep the berth idle for some time in order to minimize the total completion time, than to handle it immediately (when a small ship will arrive soon).

![Figure 2: An example of DBAP, where overpassing is beneficial and the berth may be kept idle even if a ship is waiting (which is noted by I).](image)

If $A_j > S_i$ for some ships and berths, the possibility of idle periods at some berths must be taken into account. This is done in Imai et al. (2001) through the introduction of additional continuous variables $y_{ijk}$ representing the length of the idle period at berth $i$ before the arrival of ship $j$, which will be handled in $k$th position. Constraints for giving the right values to these variables are also added.

As the same error as in the SBAP model is made in the DBAP model of Imai et al. (2001) (and also in a recent paper by the same authors, i.e., Imai et al., 2003), we present this model after due corrections have been made. Therefore, we again use the reverse order for the position of the ships at a berth in contrast to Imai et al. (2001), who attempted to use the direct order. The continuous variables $y_{ijk}$ then represent the length of the idle period at berth $i$ before the arrival of ship $j$, which will be handled in the $k$th last position.
The corrected DBAP model is the following:

\[
(DBAP) \quad \text{Minimize} \quad \sum_{i \in B} \sum_{j \in V} \sum_{k \in O} (k_{ij} + S_i - A_j)x_{ijk} + \sum_{i \in B} \sum_{j \in V} \sum_{k \in O} ky_{ijk} \quad (21)
\]

subject to

\[
\sum_{i \in B} \sum_{k \in O} x_{ijk} = 1 \quad \forall j \in V, \quad (22)
\]

\[
\sum_{j \in V} x_{ijk} \leq 1 \quad \forall i \in B, k \in O, \quad (23)
\]

\[
\sum_{l \in V} \sum_{m \in P_k} (c_{il} x_{ilm} + y_{ilm}) + y_{ijk} - (A_j - S_i) x_{ijk} \geq 0 \quad \forall i \in B, j \in W_i, k \in O, \quad (24)
\]

\[
x_{ijk} \in \{0, 1\} \quad \forall i \in B, j \in V, k \in O, \quad (25)
\]

\[
y_{ijk} \geq 0 \quad \forall i \in B, j \in V, k \in O. \quad (26)
\]

where \( P_k = \{m \in O : m > k\} \) and \( W_i = \{j \in V : A_j > S_i\} \).

The total completion time of all ships handled at berth \( i \) is

\[
\sum_{j \in V} \sum_{k \in O} ((k_{ij} + S_i - A_j)x_{ijk} + ky_{ijk}). \quad (27)
\]

This is illustrated in Figure 3.

![Diagram of ship handling at berth](image)

Figure 3: Example of ship handling at berth \( i \) for DBAP: \( A_1 = -1, A_2 = 2, A_3 = 12, A_4 = 16, W_1 = 1, H_1 = 4, W_2 = 2, H_2 = 6, W_3 = 0, H_3 = 6, W_4 = 2, H_4 = 2, \) TCT=23.

When a ship arrives at this berth, it may either (i) have to wait, which is noted by \( W_j \), as \( A_j < S_i \) (e.g., for \( j = 1 \) in Figure 3) or as the berth is occupied (e.g., for \( j = 2 \) in Figure 3) or because a ship with a short handling time will be arriving soon, or (ii) be
processed immediately as the berth is unoccupied. In this last case, an idle period for the berth (which is noted by \( I \) and may be equal to 0) immediately precedes the handling of the ship (e.g., for \( j = 3 \) in Figure 3). The total completion time for the ships handled at berth \( i \), represented by the shaded area in Figure 3, is given by (27) and may be explained as follows:

Let
\[
j^*_i(k) = j \in V | x_{ijk} = 1
\]  

(28)
de note the index of the ship handled in the \( k \)th last position at berth \( i \) for \( k = 1, 2, \ldots, T_i \). Moreover, let \( O_i = \{1, 2, \ldots, T_i\} \) for all \( i \in B \). Consider first the sum for all such ships that arrive before \( S_i \) of their waiting time until berth \( i \) becomes available, where \( \bar{W}_i = \{ j \in V : A_j > S_i \} \):

\[
\sum_{k \in O_i} \sum_{j^*_i(k) \in V \setminus \bar{W}_i} (S_i - A_{j^*_i(k)}).
\]  

(29)
This sum is represented by the areas of all shaded rectangles to the left of the vertical line passing through \( S_i \) in Figure 3. Then, focus on the sum for all ships handled at \( i \) of the difference between their departure time and \( S_i \) (which includes waiting, handling, idle and those periods before a ship arrives):

\[
\sum_{k \in O_i} \sum_{j^*_i(k) \in V} (d_{j^*_i(k)} - S_i)
\]  

(30)
where the departure time \( d_{j^*_i(k)} \) corresponds to the abscissa of the end of a horizontal segment in the upper staircase line of Figure 3, and (30) corresponds to the area below that line and to the right of \( S_i \) in that figure. Partitioning this area into rectangles bounded by the abscissae axis, the upper staircase line and the vertical lines at the abscissae corresponding to the successive arrival and departure times shows that (30) is equal to

\[
\sum_{k \in O_i} k(y_{ij^*_i(k)k} + c_{ij^*_i(k)}).
\]  

(31)

Summing (29) and (31) gives an overestimate of the total completion time, as ships with \( j \in \bar{W}_i \) arrive after \( S_i \). A correction of

\[
\sum_{k \in O_i} (A_{j^*_i(k)} - S_i)
\]  

(32)
must therefore be subtracted from this sum. It corresponds to the area bounded by the vertical line through \( S_i \), the abscissae axis and a staircase line immediately below and to the right of each of the shaded rectangles corresponding to handling a ship at berth \( i \) in Figure 3.
Summing then gives
\[
\sum_{k \in O_i} \left\{ \sum_{j \in V} (S_i - A_j^*(k)) + k(y_{ij}^*(k) + c_{ij}^*(k)) \right\}
\]
which is the same as (27) after the decision variables are fixed.

The constraints of the DBAP model will be the same as those of the SBAP model, except that we have to define idle periods at berth \( i \) as a constraint in DBAP. The constraints (24) that define idle periods at berth \( i \) are easier to explain. When the \( k \)th last ship to be handled arrives, the time during which berth \( i \) has been available is
\[
A_j^*(k) - S_i
\]
and it has been used or idle until the departure of the ship handled at the \((k + 1)\)st last position, for a time of 0 if \( k = T_i \) and
\[
\sum_{m \in O_i, m > k} (y_{ij}^*(m)m + c_{ij}^*(m))
\]
otherwise.

Subtracting (35) from (34) corresponds to the constraints (24) on the length \( y_{ijk} \) of the idle period before handling the \( k \)th last ship, after the decision variables have been fixed.

Unfortunately, the size of problem (21)–(26) rapidly increases with the number of berths and ships considered. There are \( IJT = |B||V||O| \) binary variables \( x_{ijk} \) and as many continuous variables \( y_{ijk} \), \( J \) constraints (22), \( IT \) constraints (23) and \( IJT \) constraints (24) (at most, as some \( j \in V \setminus W_i \) for some \( i \) and then no idle time \( y_{ijk} \) occurs). For a realistic example with \( I = 10 \) and \( J = 50 \), as used in Imai et al. (2001), there are 25,000 0-1 variables, 25,000 continuous ones, 50 constraints (22), 500 constraints (23) and 25,000 constraints (24).

Such numbers are large for a mixed-integer program, even if constraints (22) and (23) are multiple-choice ones, which can be efficiently exploited in algorithms for mixed-integer programming with special ordered sets (Forrest et al., 1974) as implemented in most standard packages. In the next section we will present an equivalent, more compact, formulation. In order to take advantage of the assignment-type structure of constraints (22) and (23), as in the SBAP, Imai et al. (2001) moved constraints (24) into the objective function using Lagrangian relaxation. Then, for the \( y_{ijk} \) only the lower bounds of 0 remained in the constraint set. This implies that these variables will be equal to 0 at the optimum if their coefficient in the Lagrangian relaxation is positive or to \( \infty \) if it is negative (indicating infeasibility). Imai et al. (2001) set all \( y_{ijk} \) at 0. The resulting bounds, obtained with subgradient optimization, are therefore not tight unless there is no idle period at any berth. This happens in rare cases, in which it is assumed that the vast majority of ships arrive at the port before some or all of the berths become available. Most of the constraints (24) then disappear.
Some further modifications can be brought to DBAP in order to take additional realistic restrictions into account.

1. **Ship size constraints:** Some berths \( i \) may not be able to handle some ships \( j \) because of the excessive length or draft of the ships; it suffices to set \( x_{ijk} = 0 \) for such pairs \((i, j)\) and all \( k \in O \) to express that. Alternately, one could use very large \( c_{ij} \) to impose \( x_{ijk} = 0 \).

2. **Due dates:** It may be unwise to have some ships wait for long periods of time while others that arrive later are processed, even if this is required by the schedule to minimize the total completion time. Due dates may also be imposed by contracts. This can be taken into account by introducing due date constraints, i.e., times \( L_j \) before or at which the ships \( j \) must have left or leave the port. Such constraints take the following form:

\[
S_i + \sum_{l \in V} \sum_{m \in P_k} (c_{il}x_{ilm} + y_{ilm}) + y_{ijk} + c_{ij}x_{ijk} \leq L_j \quad \forall i \in B, j \in W_i, k \in O.
\]  

Indeed, the time at berth \( i \) is the time \( S_i \) when it becomes available, plus the times for handling ships there, with an index larger than or equal to \( k \) in the reverse handling order, plus the waiting times before handling them, and it should not be larger than \( L_j \). When it is the case, (36) imposes \( x_{ijk} = 0 \).

3. **Horizon:** As new information about arriving ships becomes available over time and there are some uncertainties, it appears to be realistic not to plan too much in advance, i.e., to impose a horizon \( H \) before or at which all ships considered will have been handled. It then suffices to take \( L'_j = \min\{L_j, H\} \) instead of \( L_j \) in (36). Note that imposing a fairly close horizon will also spread the work load among berths more equally, which could be a desirable secondary objective. By the same token, one might wish to roughly equalize the number of ships handled at each berth; e.g., limit the maximum number of ships handled at one berth to two or three times the average. When this is the case, large values of \( k \) need not be considered anymore, and the size of the model is further reduced, both in terms of variables and of equations.

### 4 A compact reformulation of DBAP

A close look at constraints (24) and Figure 3 suggests a possible way of obtaining an equivalent formulation of DBAP more compact than (21)–(26). This requires new variables \( s_{ik} \), defined as the starting time for handling the \( k \)th ship at berth \( i \), for \( i \in B, k \in O \); note that here we use the order of handling ships in time and not the reverse order. As above, we use variables \( x'_{ijk} \) instead of \( x_{ijk} \) to indicate this. Binary variables \( z_{ik} \) are also needed for \( i \in B, k \in O \) to indicate whether a \( k \)th ship will be handled at berth \( i \) (\( z_{ik} = 0 \)) or not (\( z_{ik} = 1 \)). Finally, large constants \( M_{ik} \) for \( i \in B, k \in O \) are introduced to allow \( s_{ik} \) to be equal to 0 if no \( k \)th ship is handled at the \( i \)th berth. The model is as follows:

\[
(DBAP') \quad \text{Minimize} \quad \sum_{i \in B} \sum_{j \in V} \sum_{k \in O} (c_{ij} - A_j)x'_{ijk} + \sum_{i \in B} \sum_{k \in O} s_{ik}
\]  

(37)
subject to

\[ \sum_{i \in B} \sum_{k \in O} x'_{ijk} = 1 \quad \forall j \in V, \quad (38) \]

\[ \sum_{j \in V} x'_{ijk} + z_{ik} = 1 \quad \forall i \in B, k \in O, \quad (39) \]

\[ z_{ik} \geq z_{i,k-1} \quad \forall i \in B, k \in O \setminus \{1\}, \quad (40) \]

\[ s_{ik} \geq \sum_{j \in V} A_j x'_{ijk} \quad \forall i \in B, k \in O, \quad (41) \]

\[ s_{ik} \geq s_{i,k-1} + \sum_{j \in V} c_{ij} x'_{i,j,k-1} - M_{ik} z_{ik} \quad \forall i \in B, k \in O \setminus \{1\}, \quad (42) \]

\[ s_{i1} \geq S_i - S_i z_{i1} \quad \forall i \in B, \quad (43) \]

\[ s_{ik} \geq 0 \quad \forall i \in B, k \in O \setminus \{1\}, \quad (44) \]

\[ x'_{ijk} \in \{0, 1\} \quad \forall i \in B, j \in V, k \in O, \quad (45) \]

\[ z_{ik} \in \{0, 1\} \quad \forall i \in B, k \in O, \quad (46) \]

where \( M_{ik} \) is equal to the sum of the \((k - 1)\) largest \( c_{ij} \), plus the largest of the maximum arrival time of the ships and the time berth \( i \) becomes available. Problem DBAP' has \( IJT \) binary variables \( x'_{ijk} \) as DBAP, but only \( IT \) continuous variables \( s_{ik} \) instead of \( IJT \) variables \( y_{ijk} \), and \( IT \) additional binary variables \( z_{ik} \). Constraints (38) and (39) of DBAP' are the same as (22) and (23) of DBAP, except that variables \( x'_{ijk} \) and \( x_{ijk} \) refer to the direct and reverse handling order of ships at berths, respectively. Constraints (40) express that one cannot handle a ship in the \( k \)th position for \( k \geq 2 \) at a berth \( i \) unless there is a ship handled in the \((k-1)\)st position there. They are equivalent to constraints (19) of SBAP'', but more compact.

There are \( 2I(T-1) \) constraints (41) and (42) in DBAP', which define variables \( s_{ik} \), instead of the much more numerous constraints (24) of DBAP, which define variables \( y_{ijk} \). Constraints (41) express that a ship cannot be handled, in the \( k \)th position, before it arrives at the port. Constraints (42) specify that if a \( k \)th ship is handled at berth \( i \), this cannot begin at \( s_{ik} \), before the time \( s_{i,k-1} \) at which the handling of the previous ship began, plus the time \( c_{ij} \) required by that operation. For \( I = 10, |V| = 50 \), as considered in Imai et al. (2001) and above, DBAP' has 25,500 binary variables \( x'_{ijk} \) and \( z_{ik} \), 500 continuous variables \( s_{ik} \) and 50+500+980=1,530 constraints, so it is much more compact than DBAP in terms of constraints. Unfortunately, constraints (42) contain “big M” coefficients which are known to slow down solutions.

As in the DBAP model, additional constraints may be added to express further requirements. Ship size constraints are treated in the same way as above, setting some variables \( x'_{ijk} = 0 \). Alternately, one could use very large \( c_{ij} \) to impose \( x_{ijk} = 0 \) (but then one
should avoid using such $c_{ij}$ in the computation of the $M_{ik}$’s). Due date constraints can be expressed in a more compact way than in DBAP. They take the following form:

$$\sum_{i \in B} \sum_{k \in O} (s_{ik} + c_{ij})x'_{ijk} \leq L_j \quad \forall j \in V,$$

(47)

and there are $J$ of them, which is small. Due to constraints (38), only one variable $x'_{ijk}$ in (47) will be equal to 1 in the optimal solution. This equation then ensures that the corresponding departure date $s_{ik} + c_{ik}$ does not exceed $L_j$. Finally, the horizon constraints are treated again in DBAP’ as in DBAP.

5 Computational experience

The systematic testing of models DBAP and DBAP’ has been conducted using CPLEX-MIP on a 600MHz PC computer. First, problems have been generated as in Imai et al. (2001). They consist of three series with the following characteristics:

1. Number of berths, $|B| = 5, 10$.
2. Number of ships, $|V| = 10, 15, 20, 25, 30$.
3. Arrival times of the ships, $A_j$, are from a uniform distribution in the range of $[1, (7000/60)(|V|/|B|)]$.
4. Handling times, $c_{ij} = (2 \times u_{ij} + 1.5) \times 2000/60$, where $u_{ij}$ is a random number from the uniform distribution between 0 and 1.
5. Availability times of the berths, $S_i = 1/2, 3/5, 5/8, 7/8$, of the time interval between the arrivals of the first and last ships.

We generated three problem instances for each combination of the data above and set, for the larger instances, a maximum of 1 hour of CPU time to stop the algorithm when solving both DBAP and DBAP’ during our computational experiments. The results are presented in Tables 4-7. If the 1 hour CPU time limit was exhausted for all three problem instances, we denoted that with the entry $>3600$ in the tables. Furthermore, if only one or two out of three problem instances could be solved within the 1 hour CPU limit, we denoted it with (1) and (2), respectively, next to the CPU times.

It appears that DBAP is capable of solving all moderate-sized problems, with up to 10 berths and 30 ships. Furthermore, DBAP is able to solve most of the larger problems, with up to 10 berths and 50 ships, within reasonable CPU times. We can observe from Tables 4 and 5 that DBAP obtains the optimum solution in a shorter time when the number of ships waiting before a berth becomes available increases. We note that this becomes more apparent for larger problem instances; in a 5-berth case, when the number of ships is greater than or equal to 40, and in a 10-berth case when the number of ships is greater than or equal to 35.
Imai et al. (2001) reported heuristic solutions for similar problems with times of up to about 1500 seconds on a SUN S4/2000E workstation. For 10 berths and 50 ships, the gap of these solutions defined as (feasible solution value – lower bound) / lower bound is in the range of 0 to 2% for $S_i = 7/8$, but 50 to 200% for $S_i = 1/2$.

Table 4: Minimum, average and maximum CPU times of DBAP for different numbers of berths and ships.

<table>
<thead>
<tr>
<th>Berth</th>
<th>Ship</th>
<th>$S_i$</th>
<th>Minimum (sec)</th>
<th>Average (sec)</th>
<th>Maximum (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>1/2</td>
<td>0.8900</td>
<td>2.4067</td>
<td>5.3200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>0.8900</td>
<td>4.5767</td>
<td>11.1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>0.1900</td>
<td>2.2633</td>
<td>6.3000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>0.0700</td>
<td>0.0867</td>
<td>0.1200</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>1/2</td>
<td>2.3700</td>
<td>7.9433</td>
<td>12.3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>1.0000</td>
<td>1.9300</td>
<td>2.6200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>1.1100</td>
<td>1.5800</td>
<td>1.8900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>0.5100</td>
<td>0.5200</td>
<td>0.5300</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>1/2</td>
<td>14.6700</td>
<td>56.1600</td>
<td>128.1800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>4.2100</td>
<td>14.0200</td>
<td>19.7999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>3.8500</td>
<td>10.5933</td>
<td>17.1500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>1.1800</td>
<td>1.9500</td>
<td>2.3400</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>1/2</td>
<td>109.3900</td>
<td>2330.1501</td>
<td>6474.1800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>14.5200</td>
<td>98.2700</td>
<td>224.8999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>11.1600</td>
<td>162.8533</td>
<td>451.4400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>5.3400</td>
<td>8.8500</td>
<td>13.1500</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>1/2</td>
<td>4123.6201</td>
<td>25445.6963</td>
<td>60419.391</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>157.3800</td>
<td>427.2233</td>
<td>799.8599</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>41.3800</td>
<td>49.5633</td>
<td>55.4199</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>103.7699</td>
<td>145.4533</td>
<td>215.4600</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>1/2</td>
<td>778.5300</td>
<td>2690.6033</td>
<td>3657.5400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>180.4500</td>
<td>1197.8600</td>
<td>3042.2300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>77.9500</td>
<td>520.3333</td>
<td>1324.1200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>28.5600</td>
<td>33.9633</td>
<td>37.7400</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>1/2</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>2820.6799 (1)</td>
<td>2820.6799 (1)</td>
<td>2820.6799 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>1201.2300 (2)</td>
<td>1357.8749 (2)</td>
<td>1514.5500 (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>31.7200</td>
<td>68.5167</td>
<td>136.1400</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>1/2</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>1047.3900 (1)</td>
<td>1047.3900 (1)</td>
<td>1047.3900 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>883.6500 (2)</td>
<td>2146.7700 (2)</td>
<td>3469.8899 (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>57.5900</td>
<td>117.3500</td>
<td>225.0300</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>1/2</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>1633.5100 (1)</td>
<td>1633.5100 (1)</td>
<td>1633.5100 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>182.0000</td>
<td>241.2667</td>
<td>302.3600</td>
</tr>
</tbody>
</table>
Table 5: Minimum, average and maximum CPU times of DBAP for different numbers of berths and ships.

<table>
<thead>
<tr>
<th>Berth</th>
<th>Ship</th>
<th>$S_i$</th>
<th>CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1/2</td>
<td>0.8400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>0.1700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>0.1800</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>0.1000</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>1/2</td>
<td>1.2300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>0.7400</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>0.7600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>0.4400</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>1/2</td>
<td>38.3900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>11.8900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>8.4100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>2.4700</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>1/2</td>
<td>874.3000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>91.7700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>74.1900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>10.4100</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>1/2</td>
<td>2517.5901</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>172.7100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>153.9300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>17.4700</td>
</tr>
<tr>
<td>35</td>
<td>10</td>
<td>1/2</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>1068.8800 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>572.6600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>35.9300</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>1/2</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>587.7000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>441.7700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>154.8900</td>
</tr>
<tr>
<td>45</td>
<td>10</td>
<td>1/2</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>927.6300 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>1007.6300 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>400.9500</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>1/2</td>
<td>&gt;3600</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>2206.7500 (2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>1339.5700 (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>492.4900</td>
</tr>
</tbody>
</table>
We see from Tables 6 and 7 that DBAP\(^{'}\) obtains the optimum solution within the 1 hour CPU time limit only for small problems. However, we notice that if there are large waiting times for the ships, then DBAP\(^{'}\) becomes more powerful in obtaining the optimal solution in a shorter time compared to DBAP.

Table 6: Minimum, average and maximum CPU times of DBAP\(^{'}\) for different numbers of berths and ships.

<table>
<thead>
<tr>
<th>Berth</th>
<th>Ship</th>
<th>S(_i)</th>
<th>Minimum</th>
<th>Average</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>1/2</td>
<td>77.3000</td>
<td>366.9133</td>
<td>682.6500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/5</td>
<td>41.3000</td>
<td>98.4367</td>
<td>190.1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/8</td>
<td>36.5800</td>
<td>77.0367</td>
<td>129.1500</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7/8</td>
<td>19.9300</td>
<td>142.5300</td>
<td>341.3800</td>
</tr>
</tbody>
</table>

Then, larger problems were tested, assuming due dates \(L_j\) given by \(\max\{A_j, S_i\} + r_j\), where \(r_j\) is a random number from a uniform distribution in the range of \([C(1 - RDD/2), C(1 + RDD/2)]\), with \(C\) being the sum of all handling times \((c_{ij})\) divided by the square of the number of berths and RDD standing for the relative range of due dates. The values

Table 7: Minimum, average and maximum CPU times of DBAP\(^{'}\) for different numbers of berths and ships.

<table>
<thead>
<tr>
<th>Berth</th>
<th>Ship</th>
<th>S(_i)</th>
<th>CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>1/2</td>
<td>Minimum, Average, Maximum</td>
</tr>
<tr>
<td>3/5</td>
<td>2.8000</td>
<td>3.5500</td>
<td>4.3400</td>
</tr>
<tr>
<td>5/8</td>
<td>2.3900</td>
<td>2.5100</td>
<td>2.7000</td>
</tr>
<tr>
<td>7/8</td>
<td>0.6900</td>
<td>0.7333</td>
<td>0.7900</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>1/2</td>
<td>&gt;3600, &gt;3600, &gt;3600</td>
</tr>
<tr>
<td>3/5</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
<tr>
<td>5/8</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
<tr>
<td>7/8</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
<td>&gt;3600</td>
</tr>
</tbody>
</table>
of $RDD$ are taken as 0.2, 0.4, 0.6, 0.8, and 1.0. Preprocessing has first been made to avoid excessively large values of $k$ in variables and equations. From the computational experiments, it appears that the due dates do not change computing times. The reason for this is probably that CPLEX can detect some variables that must be equal to 0 and fix them in a preprocessing phase.

6 Conclusions

The dynamic berth allocation problem is an important one in operations of container ports. Imai et al. (2001) recently introduced a model for it. After correcting it, a more compact equivalent model has been proposed, and extended to accommodate constraints on size, the due dates of ships and a bounded planning horizon.

Computational experiments with CPLEX on both models show that the first model, DBAP, is able to solve problems with up to 10 berths and 50 ships. However, the second, more compact model, DBAP', is not efficient due to the big M’s in the constraints. Although it appears that this paper is the first to report the exact solution of berth allocation problems of a realistic size, much work remains to be done, in several directions:

(i) enrich the DBAP model, to take costs into account, instead of times; in particular, to express differences in cost for waiting and handling, as well as for operations at different berths;

(ii) develop new heuristics to quickly solve large instances to near optimality, using recent metaheuristics such as Variable Neighborhood Search (Mladenović and Hansen, 1997), and compare them with previously proposed ones;

(iii) explore an exact solution for DBAP and extensions using advanced mixed integer programming techniques such as stabilized column generation (du Merle et al., 1999) combined with branch-and-bound and adapted to their particular structure.

Acknowledgments This work was carried out during two visits of the first author at the Department of Management, the Hong Kong Polytechnic University, whose support is gratefully acknowledged. The work described in this paper was also partially supported by a grant from The Hong Kong Polytechnic University (Project No. G-T247). The authors are grateful to Dragan Urošević for running the computational experiments and to Nenad Mladenović for stimulating discussions.

References


   [http://www.jwgroup.com/images/ppp/ctmeasurement/]