An Effective Ship Berthing Algorithm

Andrew Lim
Department of Computer Science
National University of Singapore
Lower Kent Ridge Road
Singapore 119260

Abstract

Singapore has one of the busiest ports in the world. Ship berthing is one of the problems faced by the planners at the port. In this paper, we study the ship berthing problem. We first provide the problem formulation and study the complexity of the problem with different restrictions. In general, the ship berthing problem is NP-complete, although, some of its variants may be solved quickly. While a geometrical model is intuitive, the model cannot be easily extended to handle clearance constraints and berth restriction. Rather than solving the problem geometrically, we transform the problem ɩnto the problem of fixing directionsof edges ,n graph to form directed acyclic graph with minimal lonqest path. Since the problem is NP-complete, solving the problem exactly in polynomial time is highly unlikely. As a result, we devise a fast and effective greedy algorithm to can generate good solutions. The greedy method together with a tabu search like post optimization algorithm is able to return optimal or near optimal solutions.

1 Introduction

Situated at the crossroads of the world, the Port of Singapore is one of the world's busiest port. Every few minutes a ship arrives or departs the port. Every month the port handles more than one million transhipment containers.

When a ship arrives at the port, the planners must first decide where to berth the ship for the unloading and loading of containers. For the containers that are to be unloaded, the planners must decide where to place these containers in the yard. The wharf line of the port is divided into sections, and no ship can be berthed across any two sections. Which section to assign a ship to and exactly where to berth a ship within a section depend on factors like the locations of containers to be loaded and unloaded, the physical (i.e. depth of the berth) and resource limitations (i.e. suitably of quay crane) of each berth. A sketch of a port is given in Figure 1. The allocation of ships to sections and placement of containers in the yard is studied by Lim [Lim, 1998]. The approach used is a variant of graph partitioning problem. Allocation of vessels to sections were also studied by Brown [Brown *et* al., 1994; 1997].

One of the subproblems in [Lim, 1998] is the ship berthing problem. The ship berthing problem was studied by HenglHeng, Khoong, and Lim, 1996J using a mixed integer linear programming model. Their model assumed constant inter-ship clearance distance and constant end-berth clearance distance. While their model worked reasonably well on historical test data, it did quite badly on fully packed test cases. Their approach is also computationally intensive. As the ship berthing problem is only a subproblem which is called many times in the berth yard planning system, it needs to be computationally efficient. Heng's version of the problem is also very closely related to the offline version of the dynamic storage allocation problem [Wilson *et al,* 1995]. Because the dynamic storage allocation problem is a special case of the ship berthing problem, the ship berthing problem is NP-Complete [Garey and Johnson, 1979]

2 Problem Formulation

Ships come in different lengths and they arrive at the port at different times to be berthed. Every ship has an expected duration of stay which may be different from another ship. To berth a ship is to place the ship along the wharf line of a section. Once a ship is berthed, it will not be moved until its departure time. When two ships are berthed side by side, a certain minimum inter-shipdearance distance must be observed. Each ship has an inter-ship clearance distance which is dependent on the ship's length. The minimum inter-ship clearance distance of two ships berthed side by side is the larger of the two ships' inter-ship clearance distances. If a ship is berthed at the end of a section, a certain end-berth clearance distance must be observed. This end-berth clearance distance is not fixed and is dependent on the ship. A ship can also be given a fixed berthing location within a section. A ship may also be prohibited from

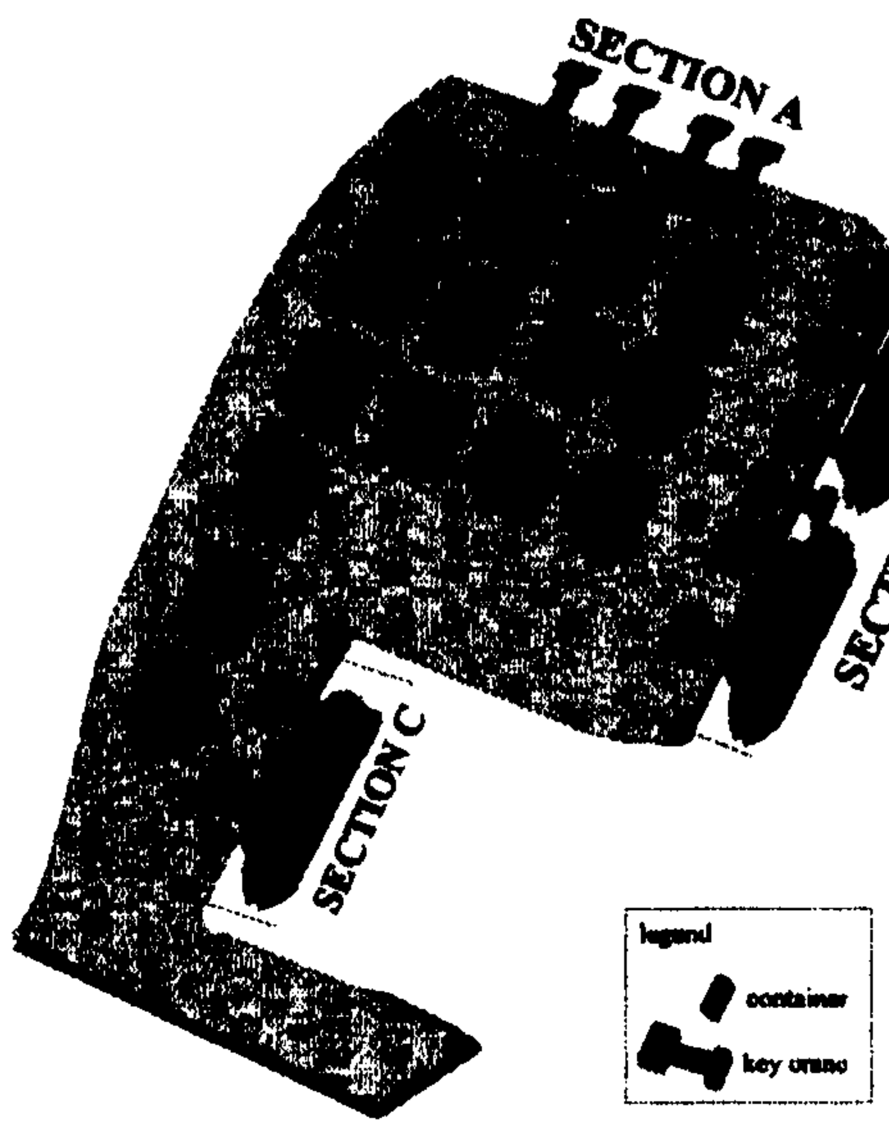

Figure 1: Sketch of a port

berthing at certain parts of a section. A berth plan for a set of ships in a section is the exact locations of ships within the section.

We can represent a *ship* geometrically by a rectangle such that the height of the rectangle is the length of the ship and the length of the rectangle is the duration of its stay. The left edge of the rectangle represents the arrival time of the ship. The right edge represents the departure time of the ship. A *section* can be represented geometrically by an infinitely long rectangle where its height represents the length of the section and the length represents the time axis. We can associate a coordinate with the left bottom corner of the geometric representation of each ship. The x-coordinate is fixed as it represents time of arrival, but the y-coordinate is not known unless the ship has a preassigned berth location. The Ship Berthing Problem is to decide the y-coordinates of the set of boxes in the long rectangle of the section such that all rectangles representing ships are non-overlapping and are within the rectangle of the section with all clearance distances are satisfied. Figure 2 clarifies the transformation.

Let us define the problem formally. Let S be the set of ships $\{S_1, S_2, \ldots, S_n\}$. Let $b_i, l_i, t_i, d_i, c_i^s, c_i^b$, and F_i be the start wharf mark of the berthing location, length, arrival time, duration of stay, inter-ship clearance distance, end-berth clearance distance, and the set of forbidden berth positions respectively of Ship 5,. When S_i is berthed at b_i, S_i will occupy wharf mark from b_i to $b_i + l_i$ from time t_i to time $t_i + d_i$. F_i is a set of intervals in the section in which ship i cannot be berthed. If S_i is berthed at 6,, the interval $[b_i, b_i + l_i]$ should not intersect with any interval in F_i. For a clearer picture of the above definitions, please refer to Figure 2.

The optimization version of the Ship Berthing Problem (SBP) is defined Figure 2, where *L* is the wharf length:

The decision version of the Ship Berthing Problem is similar to the SBP problem. All we need to do is to remove the objective function to Minimize *L* and assume that *L* is given.

2.1 Complexity

Lemma 1 *If* $\forall S_i \in \mathcal{S}, l_i = C_1, c_i^s = C_2, c_i^b = C_3$, *where* C1,C2 *and C3 are constants, the ship berthing problem can be solved in* O(nlogn) *time.*

Proof: The above problem can be transformed to the problem of coloring of interval graphs by first partitioning the section into $\lfloor \frac{L-2C_3+C_2}{C_1+C_2} \rfloor$ fixed berths and sorting the ships, in non-decreasing order based on the arrival times, followed by assigning the ships in the sorted order to the fixed berths using the criteria of most recently used available berth. This algorithm takes O(nlogn) time. □

Biro[Biro *et* al., 1992] showed that 1-precoloring of interval graph is polynomial time solvable and 2-precoloring is NP-Complete. Using his results, if $\forall S_i \in \mathcal{S}, l_i = C_1, c_i^s = C_2, c_i^b = C_3$, where C_1, C_2 and *C3* are constants, and if some of the ships are preassigned to a single berth, then the problem remains polynomial time solvable. If some of the ships are preassigned to 2 berths, then the problem becomes NP-complete even if the ships have the same length and clearance distances.

In the next section, we will show the that berth planning problem is NP-complete. Our reduction uses the PARTITION problem which is known to be NP-Complete. The definition of the partition problem is given below:

PARTITION Given a finite set $A = \{a_1, a_2, \ldots, a_x\}$ and a size $s(a) \in Z^+$ for each $a \in A$, is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A-A'} s(a)$?

Lemma 2 *The berth planning problem is NP-Complete.*

Proof: If we set all inter-ship clearances and end-berth clearances to be zero the berth planning problem is exactly the same as the dynamic storage allocation problem [Garey and Johnson, 1979]. The dynamic storage allocation problem has been shown to be NP-complete by Stockmeyer using the 3-PARTITION problem. However, this result is not published [Garey and Johnson, 1979]. In this section, we shall sketch a simple reduction using the PARTITION problem. Let $T = \sum_{a \in A} s(a)$. We use the triple (l_i, t_i, d_i) to represent the ship 5,, where l_i, t_i, d_i are the length, arrival time and the duration of stay of the ship repectively. Let $\mathcal{S} = \mathcal{S}_A \cup \mathcal{S}_B$. $\mathcal{S}_A = \{S_1, S_2, S_3, \ldots, S_9\}$, where

$$S_1 = (T + 2, 0, 1) \qquad S_2 = (1, 0, 3)$$
$$S_3 = (\frac{T}{2} + 1, 1, 1) \qquad S_4 = (1, 1, 3)$$
$$S_5 = (\frac{T}{2}, 1, 1) \qquad S_6 = (1, 2, 3)$$
$$S_7 = (\frac{T}{2}, 3, 1) \quad S_8 = (\frac{T}{2} + 1, 3, 1)$$

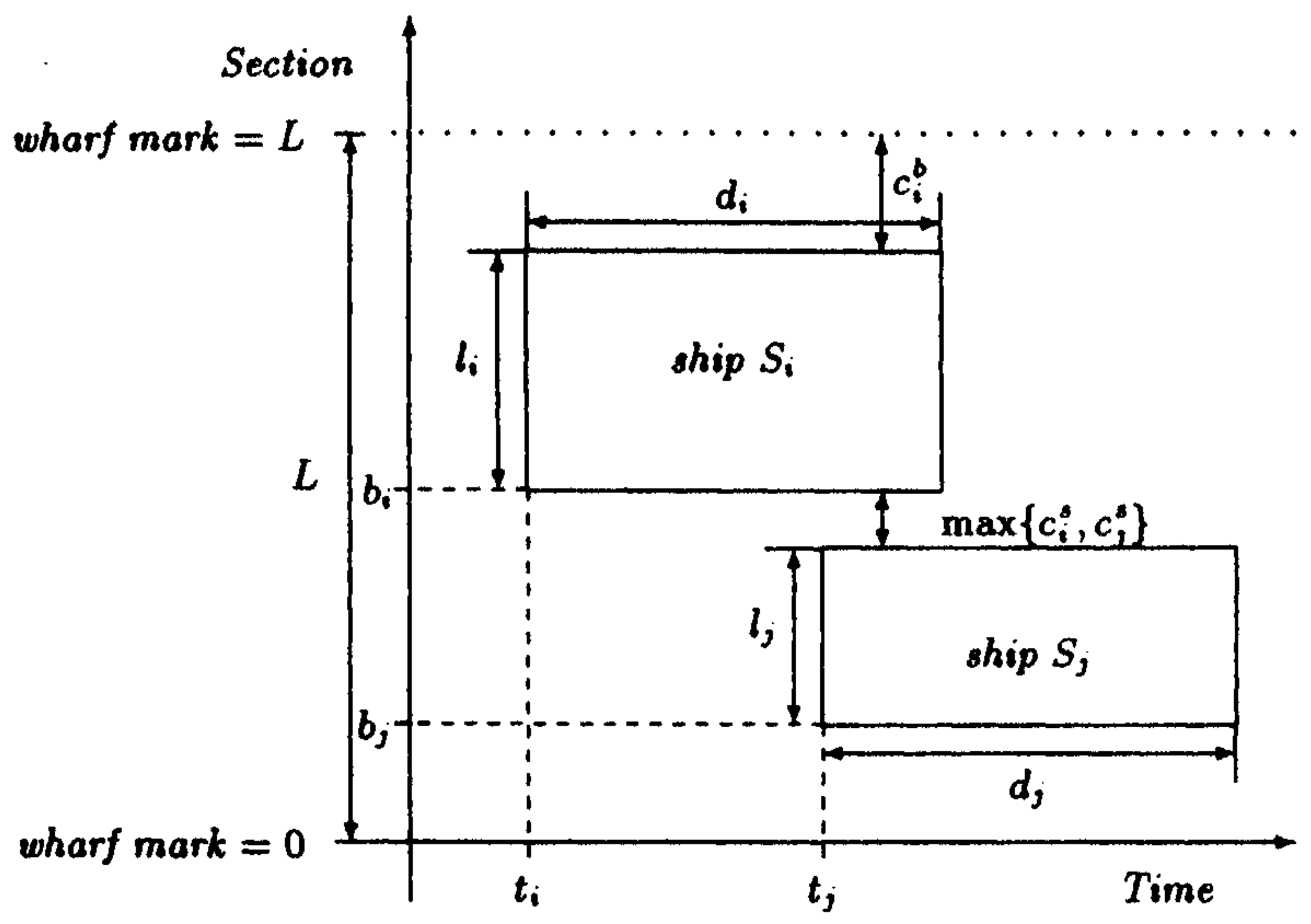

Figure 2: Geometric Representation to Graph Transformation

Minimize L
Subject to :

$$\begin{array}{rlll}
c_{ij} \geq c_i^s & \forall S_i, S_j \in \mathcal{S} & \text{s.t.} & t_i \leq t_j \leq t_i + d_i \quad \text{or} \quad t_j \leq t_i \leq t_j + d_j \\
c_{ij} \geq c_j^s & \forall S_i, S_j \in \mathcal{S} & \text{s.t.} & t_i \leq t_j \leq t_i + d_i \quad \text{or} \quad t_j \leq t_i \leq t_j + d_j \\
b_i + l_i + c_{ij} \leq b_j + (1 - x_{ij})M & \forall S_i, S_j \in \mathcal{S} & \text{s.t.} & t_i \leq t_j \leq t_i + d_i \quad \text{or} \quad t_j \leq t_i \leq t_j + d_j \\
b_j + l_j + c_{ij} \leq b_i + x_{ij}M & \forall S_i, S_j \in \mathcal{S} & \text{s.t.} & t_i \leq t_j \leq t_i + d_i \quad \text{or} \quad t_j \leq t_i \leq t_j + d_j \\
x_{ij} \in \{0, 1\} & \forall S_i, S_j \in \mathcal{S} & \text{s.t.} & t_i \leq t_j \leq t_i + d_i \quad \text{or} \quad t_j \leq t_i \leq t_j + d_j \\
c_i^b \leq b_i \leq L - l_i - c_i^b & \multicolumn{3}{c}{\forall S_i \in \mathcal{S}} \\
b_i + l_i \leq p + (1 - x_{if_j})M & \multicolumn{3}{c}{\forall S_i \in \mathcal{S} \quad \forall f_j = [p, q] \in F_i} \\
q \leq b_i + x_{if_j}M & \multicolumn{3}{c}{\forall S_i \in \mathcal{S} \quad \forall f_j = [p, q] \in F_i} \\
x_{if_j} \subseteq \{0, 1\} & \multicolumn{3}{c}{\forall S_i \in \mathcal{S} \quad \forall f_j \in F_i} \\
M & \multicolumn{3}{c}{\text{is a constant } > \sum_{\forall i} l_i + \max\{c_i^b, c_i^s\}}
\end{array}$$

Figure 3: Mathematical Model of the Problem

$$S_9 = (T+2, 4, 1)$$

For the set S_A, if $L \leq T+3$, there are only 2 berth plans (in fact one is a reflection of the other). In both of these plans, there are two unoccupied regions of equal size. One is from wharf mark 1 to $\frac{T}{2}+1$, from time unit 2 to time unit 3. The other is from wharf mark $\frac{T}{2}+2$ to T-f 2 from time unit 2 to unit 3. Let $SB = \{S_{10}, S_{11}, \ldots, S_{9+|A|}\}$ where $S_i = (s(a_{i-9}), 2, 1)$. Clearly, if there is a berth plan where $L \leq T+3$ then the set SB must be partitioned into two disjoint subsets such that the sum of lengths of ships in the two subsets are the same. It is clear that the transformation will only take polynomial time. It is also clear that the problem is in NP. Hence, the berth planning problem is NP-complete. □

Since the berth planning problem is NP-Complete, it is highly unlikely that a fast algorithm can be devised to solve the problem optimally.

3 Graph Model

While the geometrical model is visually attractive, it cannot be extended to handle clearance constraints and berth restriction constraints easily. In this section, we shall transform our geometrical representation of the Ship Berthing Problem to a graph, $\mathcal{G}$. For each ship $S_i \in \mathcal{S}$, we have a vertex v_i in $\mathcal{G}$. The weight of vertex v_i is set to l_i which is the length of the ship S_i. If two ships $S_i, S_j \in \mathcal{S}$ have time overlap, then there is an edge (v_i, v_j) linking the 2 vertices v_i and v_j. The weight of the edge (v_i, v_j) is the larger of the two ships inter-ship clearance distances, i.e. $\max\{c_i^s, c_j^s\}$. For each ship S_i, there are 2 additional vertices v_i^{et} and v_i^{eb}. These vertices have weights 0. There is a directed edge $< v_i^{eb}, v_i >$ and another directed edge $< v_i, v_i^{et} >$. The weight of these directed edges is the end-berth clearance c_i^e.

If a ship S_j is required to be fixed at a particular berth location k (i.e. b_j = k and S_j will occupy wharf mark from k to $k+l_j$) within the section, we will add two vertices u\ and u*i and two edges $< u_1, v_j >$ and $< v_j, u_2 >$. u1 and u_2 will have vertex weight of 0. $< u_1, v_j >$ and $< v_j, u_2 >$ will have edge weight of k and $L-(k+l_j)$. Similarly, a ship S_j may not be permitted to berth from wharf mark p to wharf mark q. We can handle this situation by creating a fictitious ship which is fixed at location p with length $q-p-c_j^s$ which has a hypothetical time overlap with only ship j. Using such a transformation, berth restrictions for ships can be handled consistently.

At the end of the transformation, we have a graph consisting of directed and undirected edges. Let us pick an undirected edge $e_{ij} = (v_i, v_j)$. e_{ij} exists because ship S_i and ship S_j have a time overlap. This implies that both ship S_i and ship S_j cannot share any part of the section. If ship S_i is berthed at a lower wharf mark, (a wharf mark is a particular position in a section) than ship S_j, we will set the edge (v_i, v_j) to become $< v_i, v_j >$. Similarly, if ship S_j is berthed at a lower wharf mark than ship S_i, we will set the edge e_{ij} to go from vertex v_j to vertex v_i, i.e. $< v_j, v_i >$. The ship berthing problem has been transformed to a problem of setting the directions of undirected edges in the graph such that the graph becomes directed acyclic and the longest path in the graph is minimized. The length of a path in our graph is the sum of all vertex weights and edge weights of all vertices and edges in the path.

The directed acyclic condition of the graph is important as the directions of the edges represent relative berth locations, therefore it is impossible to have a situation such that ship S_i is berthed at a lower location than ship S_j, and ship S_j is berthed at a lower location than ship S_k, and ship S_k is berthed at a lower location than ship S_i. Therefore, when we set the direction of the undirected edges we must not create any cycle. The length of the longest path in the directed acyclic graph created is the minimum length required in the section to berth the set of ships S.

Lemma 3 *The Ship Berthing Problem can be transformed to a problem of fixing the directions of some edges in a graph such that the graph becomes directed acyclic and the longest path in the graph is no more than the section length.*

Proof: Obvious from the above discussion.□

It is clear that the optimization version of the Ship Berthing Problem can be transformed to a problem of fixing the directions of edges in a graph such that the graph becomes directed acyclic and the longest path in the graph is minimized. In the worst case, only *0(n)* of the decision versions of the problem need to be solved through the use of binary search on L.

4 An Effective Greedy Algorithm

In this section, we shall discuss our heuristic for fixing the edge directions in our graph representation to create a DAG with the minimum longest path. Before describing the algorithm, let us define the following. Let e_{ij} be the edge between vertices v_i and v_j. If e_{ij} is undirected, it can be set in two directions, namely from v_i to v_j or from v_j to v_i.

Let β_{ij} and β_{ji} be defined as follows :

$$\beta_{ij} = L_{In}(v_i) + w(v_i) + w(e_{ij}) + w(v_j) + L_{Out}(v_j)$$

$$\beta_{ji} = L_{In}(v_j) + w(v_j) + w(e_{ij}) + w(v_i) + L_{Out}(v_i)$$

$L_{In}(v_i)$ is the longest incoming path of vertex v_i. $L_{Out}(v_i)$ is the longest outgoing path from vertex v_i. $w(v_i)$ and $w(e_{ij})$ are the weights of verte v_i and edge (i,j) respectively.

For every undirected edge, e_{ij}, the potential of edge, $\Phi(e_{ij})$ is given by $\max\{\beta_{ij}, \beta_{ji}\}$. The algorithm of our heuristic is given in Figure 4. The algorithm first computes the potential of every undirected edge. Next the algorithm selects the undirected edge with the highest potential. In the event of a tie, a second criterion is used. This criterion is $|\beta_{ij} - \beta_{ji}|$, the larger the better. Once the edge is selected, the direction of the edge is set