Chapter 2

The Berth Allocation Problem

In this chapter, we present a complete description of the berth allocation problem (BAP) and define it formally. We then explain the directions we took in our literature survey and summarise our findings.

2.1 Problem Description

In a container transshipment port, the *berth* is a stretch of land at the ocean front along which incoming vessels are berthed. The berth is logically divided into *sections*, which are straight segments of land based on the geographical demarcation of the land mass. Each section typically operates independently of other sections, managing the vessels that are assigned to be berthed within it. A section is generally large enough to accommodate several vessels at any one time. Sensible port

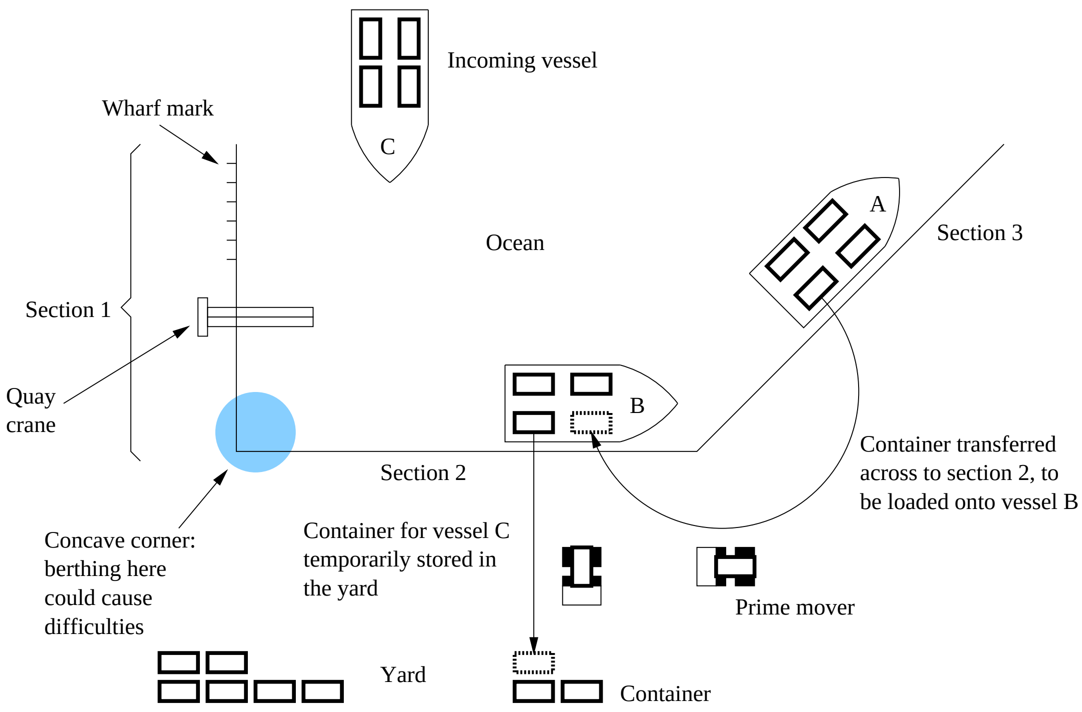

Figure 2.1: Elements of a Typical Port

planning results in sections that are multiples of 100m in length, so a section with a weird length of 342m would not be encountered. Such planning allows the section to be utilised more effectively.

Within a section, there are numerous wharf marks. A *wharf mark* indicates the start of the actual location at which a vessel is to be berthed. For vessel safety, wharf marks cannot be too close to each other and they are typically 10m apart. Hence, a 200m section has 20 wharf marks. Each section also has a number of quay cranes along the sea front for loading/unloading containers and a yard area for storing containers ([RAS96]).

Figure 2.1 shows the elements of a typical port. It also illustrates the activities that take place within the port, mainly that of managing the berthing of vessels and the transfer of containers. After an incoming container vessel has been berthed within a section, the quay cranes along the section will unload containers from it. These containers are then loaded onto prime movers which transport them for temporary storage in the yard while they await transshipment onto another vessel. When the recipient vessel arrives later, the reverse process occurs, namely the prime movers will transport the containers from the yard to the quay cranes, which will then load them onto the recipient vessel. If the recipient vessel is already in the port, the containers can be transferred directly onto it, bypassing the yard.

In Figure 2.1, vessels A and B are berthed in sections 3 and 2 respectively and a container from vessel A needs to be loaded onto vessel B. Since they are berthed in different sections of the port, a prime mover has to transport the container from section 3 across to section 2. As a port normally occupies several kilometres of land, transportation of containers across sections should be minimised as it slows down the overall transshipment process. In our example, if vessel B were berthed in section 3 instead, then the transshipment could be speeded up as the container would not have to be transported across the sections.

Vessel B has a container to be transshipped to vessel C. Since vessel C has not arrived at the port yet, the container has to be temporarily stored in the yard of section 2. When vessel C arrives at the port, it should preferably be berthed in section 2 (which hopefully has enough room to accommodate it), for it would then be nearest to the container. Otherwise, if vessel C were berthed in section 1 or 3, the container would have to be transported across to that section, delaying the transshipment process.

The management of the port would typically have a schedule of incoming vessels for the next 84 hours ($3\frac{1}{2}$ days) and also a transshipment chart of the containers. The BAP then involves planning an optimal assignment of vessels to sections, and then eventually to a specific wharf mark for berthing, so as to minimise the overall container traffic that is being moved across the different sections. Doing so will lead to speedier transshipment process which will decrease the vessel turnaround time, and allow the port to increase its business by servicing more vessels.

The real world problem is actually more complicated than what we just described. There are many other considerations and physical constraints which come into play. We will mention a few examples here and when we define the BAP formally in the next section, we will state the assumptions made in our model.

For example, there is a concave corner between section 1 and section 2 in Figure 2.1. Berthing vessels in this corner would cause difficulties for a vessel in section 1 might collide with a vessel in section 2. In addition, sections may not be entirely straight; it is possible that the land may meander, introducing physical difficulties to berthing. Secondly, vessels come in different sizes, the larger long-haul vessels have a deeper draft (vertical height) than the smaller feeder vessels and they require a section with deeper water for berthing. It is possible that some sections are too shallow to accommodate the larger vessels, thus adding more constraints to the problem. Thirdly, the time taken for the overall transshipment process is also affected by the actual allocation of quay cranes and prime movers to a particular vessel. By allocating more of such resources to one vessel, the turnaround time for that vessel can be reduced. However, this will take away resources from other vessels and increase their turnaround times. Thus, the real world BAP involves a delicate balance in resource allocation.

2.2 Formal Problem Definition

As we can see from the preceding discussion, the BAP involves many constraints and is very difficult to solve optimally. From the study of Loh and Leong ([LeLo96]), the BAP, like the quadratic assignment problem, is an NP-hard problem. Before defining the problem formally, we make the following assumptions in our BAP model:

1. All sections are straight lines and the entire section length is usable for berthing.

2. All sections are deep enough to berth even the largest vessels.

3. Any vessel can be assigned to any section, and to any wharf mark within that

section.

4. The distance between two sections is approximated by taking the Euclidean distance between the mid-points of both sections.

5. The quay crane and prime mover resources are infinite, ie. we assume a vessel will always be allocated enough resources to support the loading/unloading of containers. This assumption allows us to ignore the effect of quay cranes and prime movers in our model, as these resources can be readily increased when required, unlike the berth.

6. The vessel arrival and departure times are fixed. Each vessel is berthed immediately upon arrival and its entire transshipment process can be completed within its duration of stay at the port, which is = (departure-time − arrival-time).

7. The direction of all transshipment flows are always towards vessels that are either already berthed or will be berthed later. In other words, no vessel can transship containers to vessels which have already left the port.

The constraints for the BAP are:

1. Each vessel can be assigned to exactly one section.

2. Each vessel can be assigned to exactly one wharf mark within the allocated section.

3. At any instance of time, the total length of all vessels assigned to a section must not exceed the length of that section.

4. No two vessels can occupy the same space, or overlap in space, at any point in time.

5. No vessels can be straddled across section boundaries.

We now define the BAP as follows. Let:

v denote a vessel

s denote a section

f_{ij} denote the number of containers moved from vessel i to vessel j

d_{kl} denote the inter-section distance between section k and section l

Given:

1. Set of sections $S = \{s_1, s_2, \ldots, s_m\}$ where the section length of s_k is given by $length(s_k)$

2. Set of vessels $V = \{v_1, v_2, \ldots, v_n\}$ where the vessel length of v_i is given by $length(v_i)$, the time at which v_i arrives at the port is given by $arrival\text{-}time(v_i)$ and the time at which v_i leaves the port is given by $departure\text{-}time(v_i)$

3. Table of container flows between vessels $F = [f_{ij}]_{n \times n}$

4. Table of inter-section distances $D = [d_{kl}]_{m \times m}$

5. Time window of T hours

We want to find a mapping of vessels to wharf marks within sections:

$$v_i \rightarrow (s_k, w_{ik})$$

where w_{ik} is the wharf mark assigned to v_i in s_k.

Our *objective* is to minimise the total container traffic, which is also known as the total transshipment cost, across sections:

$$\text{Min } \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{m} f_{ij} d_{kl} x_{ik} x_{jl} \tag{2.1}$$

$$\text{where } x_{ik} = \begin{cases} 1 \quad \text{if} \quad v_i \quad \text{is assigned to} \quad s_k, \\ 0 \qquad\qquad\quad \text{otherwise.} \end{cases}$$

The total transshipment cost is calculated as the product of the number of containers moved f_{ij} and the distances across which each container is moved d_{kl}.

The BAP is subjected to the following constraints:

1. Each vessel can be assigned to exactly one section.

$$\sum_{k=1}^{m} x_{ik} = 1 \qquad \forall i = 1, \ldots, n$$

2. Each vessel can be assigned to exactly one wharf mark within the allocated section.

$$\sum_{k=1}^{m} \sum_{w=0}^{length(s_k)-1} z_{ikw} = 1 \qquad \forall i = 1, \ldots, n$$

$$\text{where } z_{ikw} = \begin{cases} 1 \quad \text{if} \quad v_i \quad \text{is assigned to} \quad s_k \quad \text{at wharf mark} \quad w, \\ 0 \qquad\qquad \text{otherwise.} \end{cases}$$

3. At any instance of time, the total length of all the vessels assigned to a particular section cannot exceed the length of that section.

$$\sum_{k=1}^{m} \sum_{i=1}^{n} x_{ik} y_{it} length(v_i) \leq length(s_k) \qquad \forall t = 1, \ldots, T$$

$$\text{where } y_{it} = \begin{cases} 1 \quad \text{if} \quad t \in \quad [arrival\text{-}time(v_i), departure\text{-}time(v_i)], \\ 0 \qquad\qquad \text{otherwise.} \end{cases}$$

4. No two vessels can occupy the same space simultaneously.

$$\forall i \forall j, i \neq j \text{ if } x_{ik} x_{jk} = 1$$

then either one of the following must be true:

(a) $departure\text{-}time(v_i) < arrival\text{-}time(v_j)$ — v_i left of v_j

(b) $arrival\text{-}time(v_i) > departure\text{-}time(v_j)$ — v_i right of v_j

(c) $\forall w, k, w'$ s.t. $z_{ikw} = 1$ and $z_{jkw'} = 1$, $w + length(v_i) < w'$ — v_i bottom of v_j

(d) $\forall w, k, w'$ s.t. $z_{ikw} = 1$ and $z_{jkw'} = 1$, $w > w' + length(v_j)$ — v_i top of v_j

5. A vessel cannot be straddled across section boundaries.

$$\forall z_{ikw} = 1, (w + length(v_i)) x_{ik} \leq length(s_k) \qquad \forall i = 1, \ldots, n$$

2.3 Literature Survey

This section summarises the literature survey that was done on the BAP and on the graph partitioning problem. The reason graph partitioning was chosen is we believe that a graph can adequately model the BAP partitioning stage. Intuitively, the nodes in a graph represent the vessels and the edges represent the transshipment flows between the vessels. The BAP partitioning can then be viewed as a graph partitioning problem in which we partition the nodes into a number of parts, where each part corresponds to a section. Graph partitioning algorithms attempt to minimise the sum of the edges interconnecting the different parts and this translates into minimising the transshipment traffic across sections in the BAP partitioning. With this in mind, we directed our literature search into the area of VLSI which is a very matured research area where graph partitioning algorithms are used extensively.

2.3.1 BAP Literature

There are relatively few research papers published on the BAP. From our literature search, we found research works done by Leong and Hum ([LeHu92a], [LeHu92b]), Loh and Leong ([LoLe96]) and Heng, Khoong and Lim ([HKL96]). We will briefly mention their works.

Leong and Hum published their research on the BAP in 1992 ([LeHu92a], [LeHu92b]). They modelled the BAP as a network flow problem and solved it using a network flow algorithm. Their model only considered the container flows between sections, but did not take into account the inter-section distances. Thus there is no way for their algorithm to capture the cost of moving a container from one end of the port to the other end. We do not see an easy way of extending their model to incorporate inter-section distances.

Loh and Leong ([LoLe96]) proposed a 0-1 integer programming formulation for the BAP and modelled it as a generalization of the quadratic assignment problem with two-dimensional interval packing constraints. Their model viewed each section as a two-dimensional bin with the horizontal dimension denoting time in discretised units and the vertical dimension denoting berth space. A port with many sections was modelled as a set of two-dimensional bins. A vessel occupied a certain berthing space over a period of time and its resource requirement was represented by a rectangle (see Figure 2.2). Solving the BAP involved finding a packing for the set of rectangles. Their research focused on solving the BAP partitioning problem, minimising the objective function given in equation (2.1), and leaving the packing of vessels within a section to be solved as a separate problem. Their solution made use of advanced search techniques: a genetic algorithm (GA) heuristic and a hybrid GA combined with tabu search.

Heng, Khoong and Lim ([HKL96]) also modelled the BAP as a two-dimensional packing problem similar to Figure 2.2. Their work focused on finding an allocation

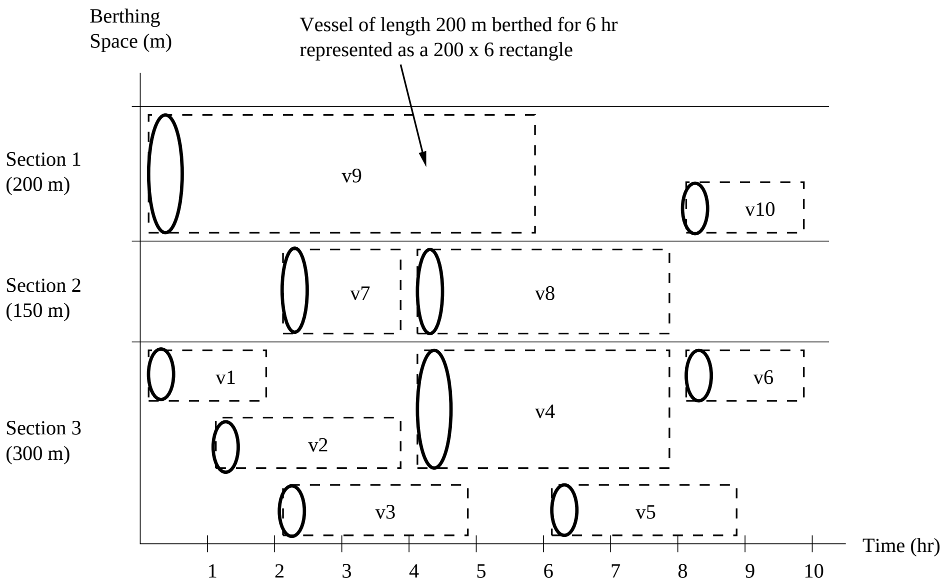

Figure 2.2: 2-Dimensional Bin Model

of wharf marks to a set of vessels which had been assigned to a section. The initial linear programming formulation they used became intractable as the number of vessels increased. Their final solution was a heuristic control strategy that iteratively rolled across the time horizon and solved each sub-horizon using LP. The vessels at the edge of each sub-horizon were added back to the problem set to be reassigned whenever the sub-horizon was moved forward.

We also mention the other BAP research done by the RAS Group, namely [ChLe99], [SiLe99], [XuLe99]. [XuLe99] solved the BAP partitioning problem using exhaustive search with branch-and-bound. Their contribution allows the RAS Group to generate the optimum solutions for small data sets to serve as benchmarks for the other BAP algorithms (see Chapter 5 for more details). [ChLe99], [LiLe99] and [SiLe99] studied the BAP packing problem in detail — whereby vessels were assumed to be already assigned to sections and that at any instance of time, the total length of all the vessels present in a section did not exceed the section length. [ChLe99] and [SiLe99] solved this problem using rectangle packing heuristics while [LiLe99] used mathematical programming methods.

2.3.2 Graph Partitioning Survey

Most VLSI partitioning algorithms perform a 2-way partition of the given graph, separating the nodes into 2 parts A and B, such that the total cost of the *cut-size* (external edges connecting nodes in A to the nodes in B) is minimal. Each part is bounded by a maximum part size to prevent a zero-size counterpart, or the trivial solution of moving all nodes into only one part. These algorithms also generalise from a graph into a hyper-graph, as edges connecting 2 or more nodes are mapped into the net-lists in VLSI circuits ([ScKe72]). A k-way ($k > 2$) or multi-way partition of a graph is often obtained by recursively applying the partitioning algorithm to the resultant parts A and B ([KeLi70]). There are also algorithms which specialise in multi-way partitioning per se ([Sanc89]).

Kernighan and Lin ([KeLi70]) are the pioneers in the area of graph partitioning. Their KL heuristic bi-partitions a graph by performing pairwise exchanges of nodes. Starting with a randomly generated initial solution, the KL heuristic goes through a number of moves whereby it selects a node from A and a node from B such that swapping them produces the greatest decrease (or the smallest increase if no decrement is possible) in the cut-size. The decrease in cut-size is termed the *gain*. The moved nodes are then "locked" to prevent consideration again in future moves, to ensure termination of the algorithm. All the nodes will eventually be locked, and the algorithm then computes the cumulative partial sum of all pairwise exchanges performed, and selects the solution obtained from a subset of these exchanges such that the overall gain is maximal. This ends one *pass* of the algorithm. Further passes can be performed by using the solution obtained as the new initial solution, until no further improvement can be found at the end of a pass.

The selection of the node pair for exchange is done using a gain function associated with each node. The gain function captures the change (decrease or increase) in the cut-size by moving that node to the opposite part. After a node has been moved, the gain of all its neighbouring nodes have to be updated to reflect the effect of moving this node. Incremental gain updates can be performed in $O(n)$ time, while selection of the node pair for exchange typically takes $O(n^2)$. There are n nodes to be processed in each pass, so the KL heuristic requires $O(n^3)$ time per pass.

Fiduccia and Mattheyses ([FiMa82]) produced the next major breakthrough with their FM heuristic that reduces the running time of one pass to $O(p)$ where p is the total number of *pins* (a pin connects an edge to a node in VLSI circuits).

Instead of performing a pairwise exchange of nodes at each move, the FM heuristic only moves the highest gain node to the opposite part. Doing so increases the number of moves per pass, but eliminates the $O(n^2)$ time associated with the selection of the node pair. In addition, the FM heuristic uses a carefully implemented bucket data structure which allows the selection of the highest gain node to be done in $O(1)$ time and the total gain updates to be done in $O(p)$ time amortised across one pass. The FM heuristic is thus able to achieve a significant improvement in running time over the KL heuristic with very little loss in solution quality.

Running time plays in important part in VLSI partitioning because the number of nodes range from hundreds of thousands to several millions. Therefore, a running time of $O(p)$ vs $O(n^3)$ makes a lot of difference. p is $O(n^2)$ for a complete graph, but a complete graph scenario is impractical in the VLSI context; we note that this observation is also true for the BAP. As such, over the past decade, the spirit of the FM heuristic forms the major backbone behind many later, newer partitioning algorithms. We will briefly survey a few such algorithms, which extend and improve the solution quality of the FM heuristic.

Krishnamurthy ([Kris84]) attempted to enhance the FM heuristic with a "look-ahead" gain computation. One drawback in the FM heuristic is that when there is more than one node with the same highest gain, the candidate node is selected randomly. The look-ahead gain tries to provide an "intelligent" tie-breaker in such situations. Krishnamurthy's algorithm allows the number of look-ahead steps to be selected — more look-ahead produces a better solution but with a corresponding increase in the running time. Sanchis ([Sanc89]) also modified the FM heuristic with look-ahead capabilities. In addition, her model also extends the original bipartition of the FM to handle multi-way partition directly.

In the KL/FM-based algorithms, the part sizes must be pre-defined or a ratio of the sizes of both parts must be given. To meet this part size constraint, the algorithms may end up with "unnatural" clusters of nodes, as there is normally no way to know the optimal part sizes before partitioning. Based on this motivation, Wei and Cheng ([WeCh89], [WeCh90]) defined a ratio cut metric to be used to replace the standard gain function. The ratio of a cut is defined by them to be the ratio of the cut-size over the product of the two part sizes, and the ratio cut is defined as the cut generating the minimum ratio. The idea behind the ratio cut is to define a metric which balances the cut-size and the part sizes to produce a natural clustering of nodes. Wei and Cheng's algorithm uses the ratio cut metric to find an initial solution with the minimum ratio, and then apply the FM algorithm with the gain re-defined as the change in the ratio. Results obtained with ratio cut are comparable to, and at times better than, the FM heuristic.

Wei and Cheng extended their ratio cut idea to a two-phase FM algorithm ([WeCh90]). In phase I, a clustering of nodes is constructed and the FM algorithm is run on this graph of clusters by treating the clusters as hyper-nodes. After the clusters have been optimally partitioned, they are then "flattened" back into the

Authors	**Node Size**	**Edge Cost**	**Part Size**	**Method**
Kernighan, Lin 1970	equal	integer	fixed	pairwise exchange, $O(n^3)$ runtime
Fiduccia, Matt-heyses 1982	variable	1	given ratio	single node move, $O(p)$ runtime
Krishnamurthy 1984	variable	1	given ratio	extends FM with look-ahead
Sanchis 1989	variable	1	given ratio	extends FM to multi-way with look-ahead
Wei, Cheng 1989	variable	integer	determined by ratio cut	ratio cut, finds "natural" clusters
Wei, Cheng 1990	variable	integer	determined by ratio cut	2-phase FM, refine-ment of Wei, Cheng 89
Karypis, Kumar 1995	variable	integer	flexible	multi-level partitioning

Table 2.1: Summary of VLSI Partitioning Algorithms

original nodes and used as the initial solution for Phase II, where the FM algorithm is run on the set of original nodes. The 2-phase FM algorithm produces results up to 30% better than single-phase FM.

An extension of the 2-phase FM idea to a multilevel graph partitioning algorithm was done by Karypis and Kumar ([KK95], [KK98])[1]. Instead of just two levels, their algorithm successively coarsens the initial graph across multiple levels, where the number of nodes at each successive level is less than its predecessor. Partitioning is performed at the coarsest level of a few hundred nodes and the resulting graph is then uncoarsened. At each step of the graph uncoarsening, the partition is further refined.

[1]Thanks to Asst. Prof. Lau H C for this reference.

Table 2.1 summarises the above discussion. For the BAP, we adopted the spirit of the FM heuristic in our solution for its good running time and solution quality. The KL heuristic has a slow running time while the ratio cut idea to achieve natural clustering with natural part sizes is not very important to the BAP as the part sizes (which are given by the section lengths) are all known beforehand; the real number ratio cut metric also introduces complications in implementation vs using the integer gain computation in FM.