A Mathematical Model and a Heuristic Algorithm for Berth Planning

Brain Korea 21 Logistics Team
Industrial Engineering / Pusan National University

Author :
K.C. Moon
(Logistics Systems Lab., Industrial Engineering, Pusan National University)

Thesis supervisor :
Prof. K.H. Kim
(Logistics Systems Lab., Industrial Engineering, Pusan National University)

Abstract

The berth-planning problem is the problem of determining berthing times and positions of containerships in port container terminals. Each vessel requires a specific amount of the space on the berth during a predetermined length of time for unloading and loading containers. A mixed integer linear programming model is formulated for the berth-planning problem. A heuristic procedure is suggested for searching a near optimal solution. Experimental results show that the heuristic algorithm obtains solutions similar to the optimal solutions by the optimization model.

Keywords: Berth planning; mixed integer programming; heuristics

Contents

1 Introduction

Competition among ports continues to be more serious as the differentiation of ports into hub ports and feeder ports progresses. Managers in many container terminals try to attract carriers by automating handling equipment, speeding up various services, and providing most current information on the flow of containers. At the same time, they try to reduce costs by utilizing resources efficiently, including human resources, berths, container yards, container cranes, and various yard equipment. In this paper, we try to maximize the utilization of the berth and satisfy various constraints for berthing container vessels.

When planning a berth usage, berthing time and position of each vessel are usually determined. In the process, several variables must be considered, which include the length and the arrival time of each vessel, the number of containers for discharging and loading, and the storage location of outbound containers to be loaded onto the corresponding vessel. Fig. 1 illustrates a real example of the berth plan of a port container terminal.

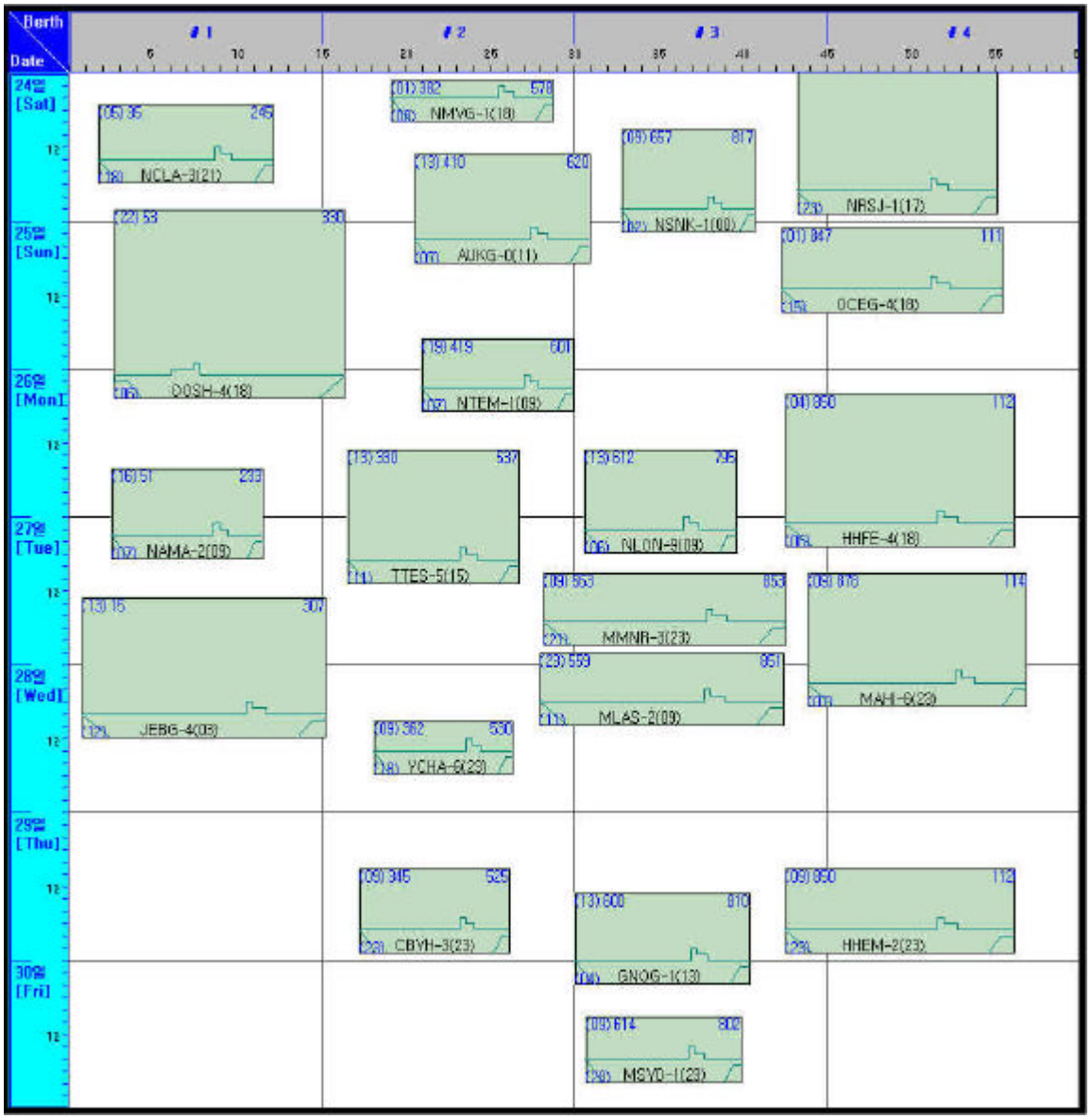


Fig. 1 A real example of a berth plan

Lai and Shih [1992] compared various berth allocation policies for container vessels in a simulation experiment. Brown et al. [1994] formulated an integer programming model for assigning one possible berthing location to a vessel considering various practical constraints. However, they considered a berth as a collection of discrete berthing locations. Lim [1998] addressed the berth-planning problem by keeping the berthing time fixed and trying to decide the berthing locations. The berth was considered to be a continuous space rather than a collection of discrete locations. He proposed a heuristic method for determining berthing locations of vessels by utilizing a graphical representation for the problem.

In the next section, the berth-planning problem is mathematically formulated. In section 3, we introduce properties of the optimal solution. In section 4, we explain to maintain the stability of clusters. In section 5, we propose an effective heuristic for the berth-planning problem. In section 6, the result of a numerical experiment will be provided. Finally, in section 7, concluding remarks will be given.

2 Problem formulation

In the berth planning, we try to minimize the penalty cost resulting from delayed departures of vessels and the additional handling cost resulting from deviation of the berthing position from the best location on the berth. Carriers usually inform the expected arrival time and the requested departure time of vessels to the terminal operator. Based on the information, a terminal operator tries to satisfy the requested departure time of each vessel. However, when the arrival rate of vessels is high or unexpected arrivals occur, it may not be possible to service all of the vessels before the service completion time that they requested. Thus, departures of some vessels may be delayed over the requested due time. Note that the terminal operator has different priorities for different types of vessels. The priorities can be converted into weights (c_{2i}) on the penalty cost of vessels in the objective function.

The berthing position is also an important decision variable. It comes from that containers for a vessel may have already arrived at the yard and so it will reduce the handling cost if the vessel is berthed at the location near to the storage location of the corresponding containers. And a certain berthing location is preferable for a vessel to other locations because of a long-term contract about the usage of the berth with carriers, the minimum draft required for a vessel, or different levels of waves according to the location on the berth, etc..

The following notations are used:

N = The total number of vessels

L = The length of the berth

p_i = The best berthing location of vessel i. This location is represented by the x-coordinate of the leftmost end of the vessel and determined considering the distribution of containers already arrived or a designated location for a specific vessel. The reference point for x-coordinate is the leftmost boundary of the berth

x_i = The berthing position of vessel i (a decision variable)

y_i = The berthing time of vessel i (a decision variable)

a_i = The estimated arrival time of vessel i

d_i = The requested departure time of vessel i

b_i = The requested time for the ship operation for vessel i. This value includes the requested allowance between departure of a vessel and berthing of another vessel

c_{1i} = The additional travel cost per unit distance for delivering containers of vessel i resulting from deviation of berthing location from the best position

c_{2i} = The penalty cost per unit time of vessel i resulting from a delayed departure over the requested due time

l_i = The length of vessel i. This value includes the requested gap between adjacent vessels

$$x_{ij} = \begin{cases} 1: \text{if vessel } i \text{ is located in the left-hand side of vessel } j \\ 0: \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1: \text{if vessel } i \text{ is located lower side of vessel } j \\ 0: \text{otherwise} \end{cases}$$

Fig. 2 illustrates the berth-time space and some notations.

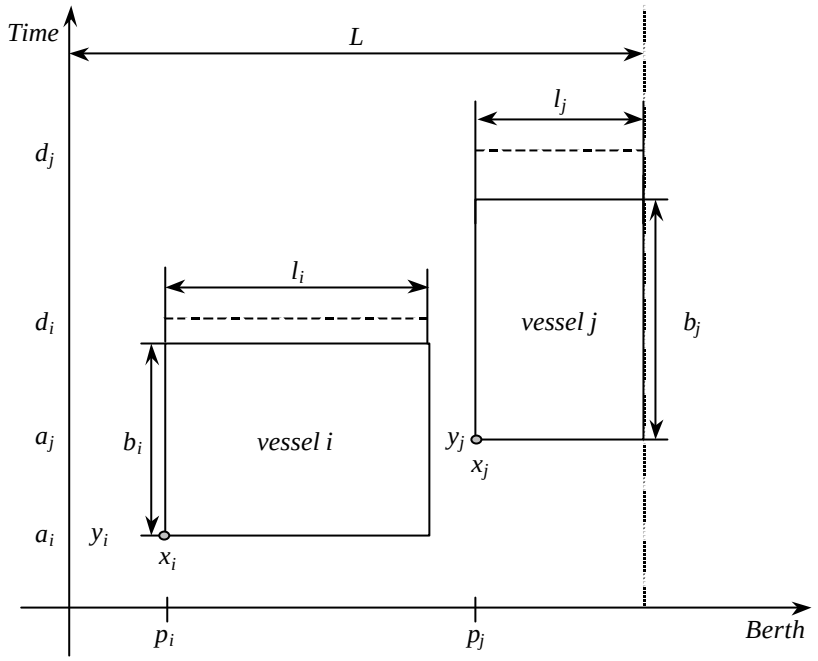


Fig. 2 An illustrative example of a berth-time space

The objective function of the berth-planning problem can be written as follows:

$$Min \ \sum_{i=1}^{N} \left\{ c_{1i} \left| x_i - p_i \right| + c_{2i} \left(y_i + b_i - d_i \right)^+ \right\} \tag{2.1}$$

where $x^+ = \max\{0, x\}$.

The first term of the objective function comes from the deviation of the berthing position from the best location and the second term is related to the penalty cost from the delay of the departure of vessels behind the requested departure time.

Let $\left| x_i - p_i \right|$ be a_i^+ when $x_i - p_i \geq 0$ and a_i^- when $x_i - p_i < 0$. And let $\left(y_i + b_i - d_i \right)$ be b_i^+ when $y_i + b_i - d_i \geq 0$, and b_i^- otherwise. Then, the berth-planning problem can be formulated as follows:

$$Min \ \sum_{i=1}^{N} \left\{ c_{1i} \left(a_i^+ + a_i^- \right) + c_{2i} b_i^+ \right\} \tag{2.2}$$

subject to

$$x_i - p_i = a_i^+ - a_i^- \quad \text{for all } i \tag{2.3}$$

$$y_i + b_i - d_i = b_i^+ - b_i^- \quad \text{for all } i \tag{2.4}$$

$$x_i + l_i \leq L \quad \text{for all } i \tag{2.5}$$

$$x_i + l_i \leq x_j + M\left(1 - z_{ij}^x\right) \quad \text{for all } i \text{ and } j,\ i \neq j \tag{2.6}$$

$$y_i + b_i \leq y_j + M\left(1 - z_{ij}^y\right) \quad \text{for all } i \text{ and } j,\ i \neq j \tag{2.7}$$

$$z_{ij}^x + z_{ji}^x + z_{ij}^y + z_{ji}^y \geq 1 \quad \text{for all } i \text{ and } j,\ i \neq j \tag{2.8}$$

$$y_i \geq a_i \quad \text{for all } i \tag{2.9}$$

$$a_i^+, a_i^-, b_i^+, b_i^-, x_i \geq 0 \quad \text{for all } i \tag{2.10}$$

$$z_{ij}^x, z_{ij}^y \quad 0/1 \text{ integer for all } i \text{ and } j,\ i \neq j \tag{2.11}$$

Constraint (2.3) and (2.4) is related to the definition of a_i^+, a_i^-, b_i^+, and b_i^-. Constraint (2.5) implies that the position of the rightmost end of vessel i is restricted by the length of the berth. Constraint (2.6) or (2.7) is effective only when z_{ij}^x or z_{ij}^y equals one. Constraint (2.8) excludes the case that two vessels are in conflict with each other with respect to the berthing time and the berthing position. That is, constraint (2.8) excludes the case $z_{ij}^x + z_{ji}^x + z_{ij}^y + z_{ji}^y = 0$ in which case the rectangles representing schedules for vessel i and j overlap with each other. Constraint (2.9) implies that a vessel cannot berth before she arrives.

A simple version of the model in this study, in which berthing times of all the vessels are fixed, is known to be NP-hard [Lim, 1998]. Thus, due to the excessive computational time, it is impractical to solve the formulation (2.2) – (2.11) by using a commercial software for integer linear programming problems.

3 Properties of the optimal solution

In this section, several useful properties of the optimal solution will be investigated. The several definitions are also provided. Note that a schedule of a vessel is represented by a rectangle as shown in Fig. 2.

Definition 1: A *x*-cluster is defined as a set of rectangles (vessels) whose the vertical sides are in contact with each other. Also, a *y*-cluster is defined as a set of rectangles (vessels) whose the horizontal sides are in contact with each other.

We assume that, for any two disjoint subset of vessels (V_1 and V_2), the sum of l_i, for all $i \in V_1$, is not equal to the sum of l_j, for all $j \in V_2$. Also, we assume that, for any two disjoint subset of vessels (V_1 and V_2), the sum of b_i, for all $i \in V_1$, is not equal to that, for all $j \in V_2$. We call this assumption as the "non-modularity assumption". Then, a cluster can be represented by a graph that can be constructed as follows. The words, "node", "vessel", and "rectangle", will be used interchangeably with each other.

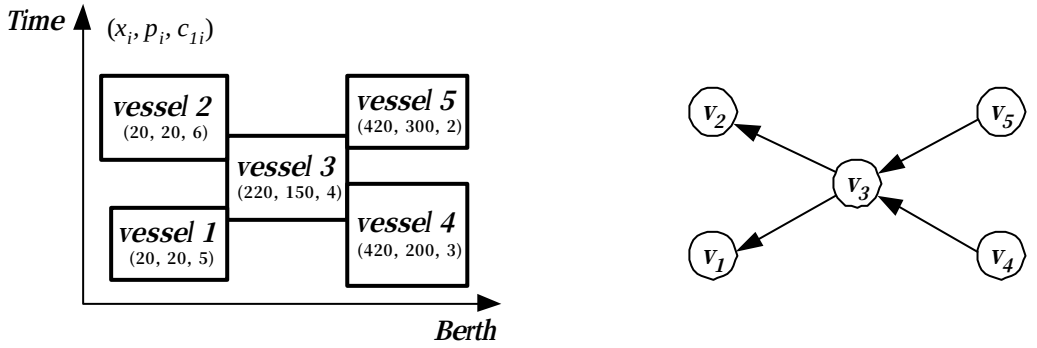
Step 0: Make a node for each vessel in the cluster. Let $S=\varnothing$ and T be the set of all the rectangles in the cluster.

Step 1: Select such rectangles from T that have no rectangle on the right-hand side. Include the selected rectangles into the set S.

Step 2: Select a rectangle from S, whose left side is located in the rightmost location among all rectangles in S. Let the selected rectangle be a. Identify rectangles in T, whose right sides are in contact with the left side of the rectangle a. Connect directed arcs from node a to the corresponding nodes of the identified rectangles.

Step 3: Eliminate rectangle a from set S and T. Include rectangles corresponding to newly connected nodes into set S. Go to *step 2* until no rectangle remains in set S.

The graph for a *y*-cluster can be constructed in the same way. Fig. 3 illustrates a *x*-cluster and the resulting cluster-graph.



(i) A berth schedule (ii) Graphic representation of the berth schedule

Fig. 3 An illustration of a berth schedule

Property 1: A cluster graph is a tree.

Proof) By the non-modularity assumption, no two branches that sprouted from a node can merge into the same node. Thus, the conclusion holds. ■

Definition 2: A cluster is said to be "stable", if the total cost for vessels in the cluster cannot be reduced by moving it in the positive or negative direction.

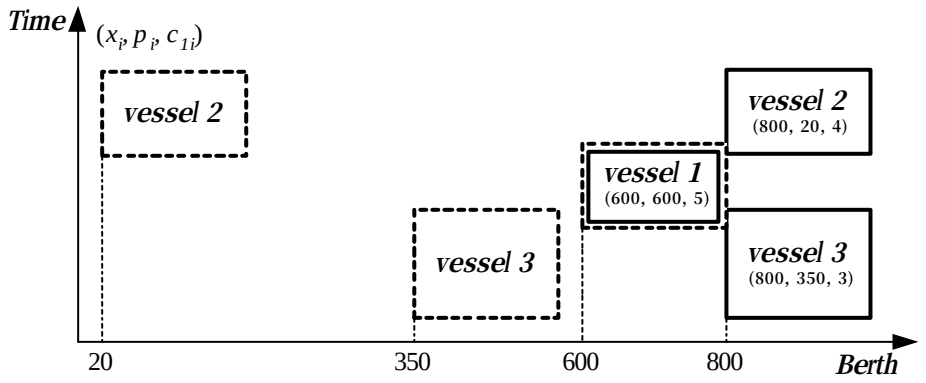
Property 2: When a x-cluster is stable, one or more vessel in the x-cluster is located at its location of the lowest cost on x-axis or in contact with either of two borderlines of the berth.

Proof) Let x be the central location of the leftmost rectangle in a cluster, which we call a "reference rectangle" and let r_i be the relative location of the center of rectangle i from the center of the reference rectangle of the associated cluster. Then, the cost of the i^{th} rectangle can be expressed as $c_{1i}|x+r_i-p_i|$. Thus, the total cost can be expressed by $\Sigma_{i \in Ck}\, c_{1i}|x+r_i-p_i|$, where c_k is a set of the k^{th} x-cluster.

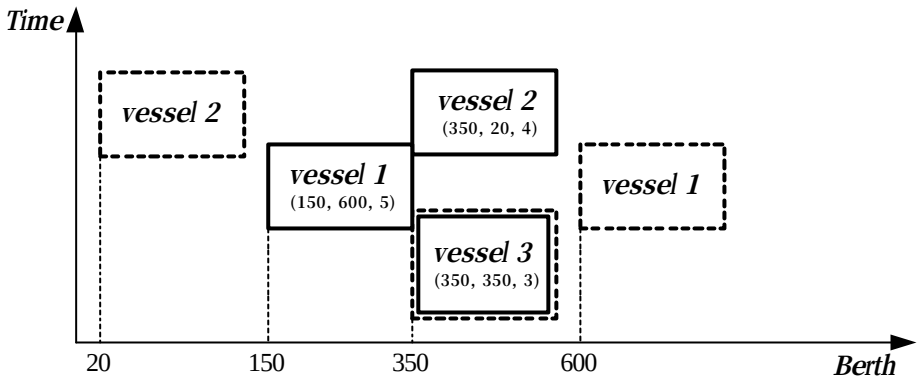
This problem is equivalent to the minisum problem of a single facility with rectilinear distance measure. In this case, (p_i-r_i) can be considered to be the location of the i^{th} existing facility. That is, x^* can be found by applying the property of the median condition [Francis, 1992]. Thus, the conclusion holds. ■

In Fig. 4, stable and unstable clusters are illustrated. Rectangles of solid line indicate the current schedule of vessels. The rectangle of dotted line represent the minimum cost location of each corresponding vessel. Note that the rectangles with both solid and dotted line imply that they are currently on the minimum cost position. Fig. 4-(i) illustrates an unstable cluster. The unstable cluster in Fig. 4-(i) can be made stable by moving it in the left-hand direction until vessel 3 is positioned at its minimum cost location.

Notice that the cost of the cluster in Fig. 4-(ii) cannot be reduced by moving it in either of right or left-hand direction.



(i) An unstable x-cluster

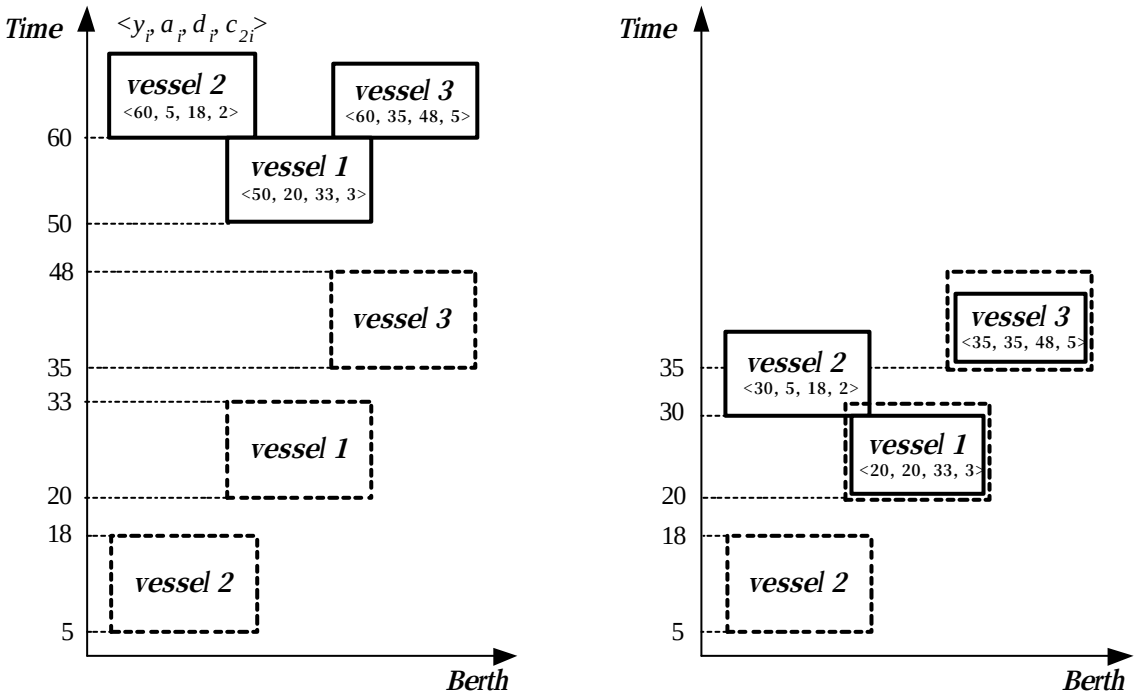


(ii) A stable *x*-cluster

Fig. 4 Illustrations of stable and unstable *x*-clusters

Property 3: When a *y*-cluster is stable, the berthing times of all the vessels in the *y*-cluster are located in the range of $[a_i, d_i-b_i]$, or the berthing time of one or more vessel in the cluster is at its arrival time (a_i).

Proof) Note that the cost of vessel i is the minimum at $y_i \in [a_i, d_i-b_i]$. If the berthing time of a vessel in the cluster is greater than d_i-b_i, then the total cost can be reduced by decreasing all the berthing times in the cluster until one of the berthing time becomes in contact with the arrival time of the vessel. Thus, the conclusion holds. ■



(i) An unstable *y*-cluster (ii) Stable *y*-clusters

Fig. 5 Illustrations of stable and unstable *y*-clusters

Fig. 5 illustrates unstable and stable *y*-clusters. In Fig. 5-(i), all the rectangles are located at positions higher than their minimum cost locations that are indicated by rectangles of dotted line.

Thus, the cluster in Fig. 5-(i) is unstable. However, the cluster can be made stable by moving it downward until all the rectangles are located in the ranges of y-axis that incur no cost or a rectangle reaches its arrival time as shown in Fig. 5-(ii).

4 Maintaining the stability of clusters

In this section, it will be discussed how to make clusters stable. For given relative positions of rectangles, the total cost will be reduced by making clusters stable. Before a procedure for making clusters stable is provided, procedures for checking stabilities of clusters will be introduced.

4.1 Procedures to check the stability of clusters

In this sub-section, procedures for checking the stability of clusters are suggested. Because checking the stability of *y*-cluster is simple, procedures for *x*-cluster will be mainly explained. In order to confirm the stability of a *x*-cluster, it must be confirmed that the movement of the *x*-cluster in either direction cannot reduce the total cost of rectangles in the cluster. In the following, procedures are suggested for checking the cost change of the movement of a *x*-cluster in either direction.

A procedure to check the stability of a x-cluster against the movement in the left-hand direction (Check-stab-x-left)

Step 0: (Initialize) Draw the cluster graph and write the weight with the plus sign on nodes whose minimum cost location is the current position or in the right-hand side of the current position. Write the weight with the minus sign on nodes whose minimum cost location is in the left-hand side of the current position. The minus sign of the weight on a node implies that the movement of the cluster in the left-hand direction will reduce the cost of the corresponding rectangle. The plus sign implies that the corresponding rectangles resist the movement of the cluster in the left-hand direction.

Step 1: If the cluster-graph consists of a single node, go to step 2. Otherwise, select a tip from the cluster-graph. If the selected tip is connected to the tail of an arc and has a positive weight, detach the selected tip and nodes included into the selected tip from the rest of the cluster to make a new cluster. If the selected tip is connected to the head of an arc and has a negative weight, detach the selected tip and nodes included into the selected tip from the rest of the cluster to make a new cluster. Go to the beginning of this step. Otherwise, add the weight of the tip to the weight of the next imminent node. We say that the selected tip is included into the next imminent node on the cluster-graph (tree). Delete the tip and the arc connected to the tip from the graph. Go to the beginning of this step.

Step 2: If the weight of the last node is positive, the *x*-cluster is stable against the movement in the left-hand direction. If it is negative, the *x*-cluster is said to be unstable against the movement in the left-hand direction. That is, it means that moving the cluster in the left-hand direction will reduce the cost of the cluster. If it is zero, the total cost will not change until a rectangle in the cluster arrives at its minimum cost position.

A procedure to check the stability of a x-cluster against the movement in the right-hand direction (Check-stab-x-right)

In this procedure, steps 0 and 2 must be modified as follows, while step 1 is the same as the procedure of *Check-stab-x-left*:

Step 0: (Initialize) Draw the cluster graph and write the weight with the plus sign on nodes whose minimum cost location is in the right-hand side of the current position. Write the weight with the minus sign on nodes whose minimum cost location is the current position or in the left-hand side of the current position.

Step 2: If the weight of the last node is negative, the *x*-cluster is stable against the movement in the left-hand direction. If it is positive, the *x*-cluster is said to be unstable against the movement in the left-hand direction. That is, it means that moving the cluster in the right-hand direction will reduce the cost of the cluster. If it is zero, the total cost will not change until a rectangle in the cluster arrives at its minimum cost position.

Note that a *y*-cluster is always stable against the movement in the upward direction, because the movement in the upward direction cannot reduce the total cost.

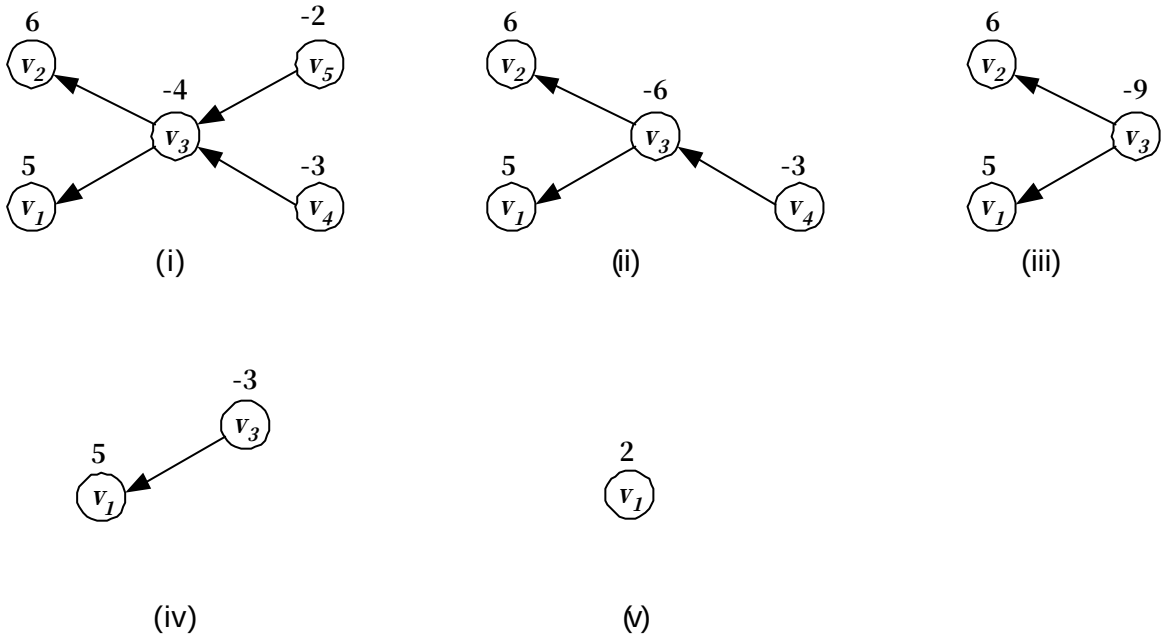


Fig. 6 An illustration of *Check-stab-left*

A procedure to check the stability of a y-cluster against the movement in the downward direction (Check-stab-y-down)

From property 3, if the berthing time of a vessel in the cluster is greater than d_i-b_i and no rectangle in the cluster is in contact with the arrival time of the corresponding vessel, then the total cost can be reduced by decreasing all the berthing times in the cluster until all the rectangles are at their minimum cost locations or at least one rectangle reaches the arrival time of the corresponding vessel.

Fig. 6 illustrates the process of *Check-stab-x-left* for the cluster in Fig. 3 step by step. In (i), tip v_5 is selected. Because tip v_5 has a negative weight, its weight is added to that of v_3. And then, v_4 is selected whose weight is again added to that of v_3. Next, v_2 is selected and finally v_1 is selected. Because the final node, v_1, has a positive weight, we can conclude that the cluster is stable against the movement in the left-hand direction.

4.2 Procedures to make clusters stable

In this sub-section, it will be discussed how to make clusters stable.

A procedure to make a x-cluster stable against the movement in the left-hand direction

The fact that a cluster is unstable implies that the total cost can be reduced by moving the cluster to either of two directions. In the following, a heuristic procedure is suggested to make the cluster stable.

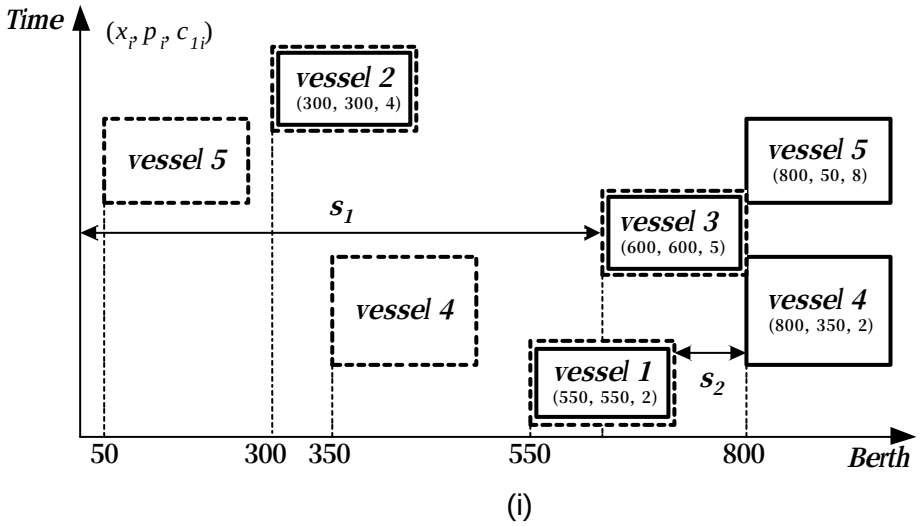
Step 1: Check if the cluster is stable against the movement in the left-hand direction by using *Check-stab-x-left.* If stable, stop. Otherwise, let s_0 be the minimum of the positive values among x-coordinates of the current locations deducted by the minimum cost x-coordinate of the corresponding rectangle in the cluster.

Step 2: Find the longest distance possible that the cluster can move to the left-hand direction without making a rectangle in the cluster be in contact with the boundary of the berth or a rectangle that is not a member of the cluster. Let it be s_1 in the first case and s_2 in the second case. $S=\{s_0, s_1, s_2\}$. Select the smallest value in S. Move all the rectangles in cluster to the left-hand direction by the selected value. If the selected element is s_1, stop. If the selected element is s_2, make a new cluster additionally including the contacted rectangle. Go to step 1.

Note that the member of the cluster may change after each iteration of step 2. The procedure to make a x-cluster stable against the movement in the right-hand direction is similar to step 1-2 above.

A procedure to make a y-cluster stable against the movement in the downward direction

If the berthing time of a vessel in the cluster is greater than d_i-b_i and no rectangle in the cluster is in contact with the arrival time of the corresponding vessel, the cluster is unstable. Thus, move all the rectangles in the cluster downward until the berthing times of all rectangles in the cluster is less than or equal to d_i-b_i or a rectangle in the cluster is in contact with the arrival time of the corresponding vessel.



(i)

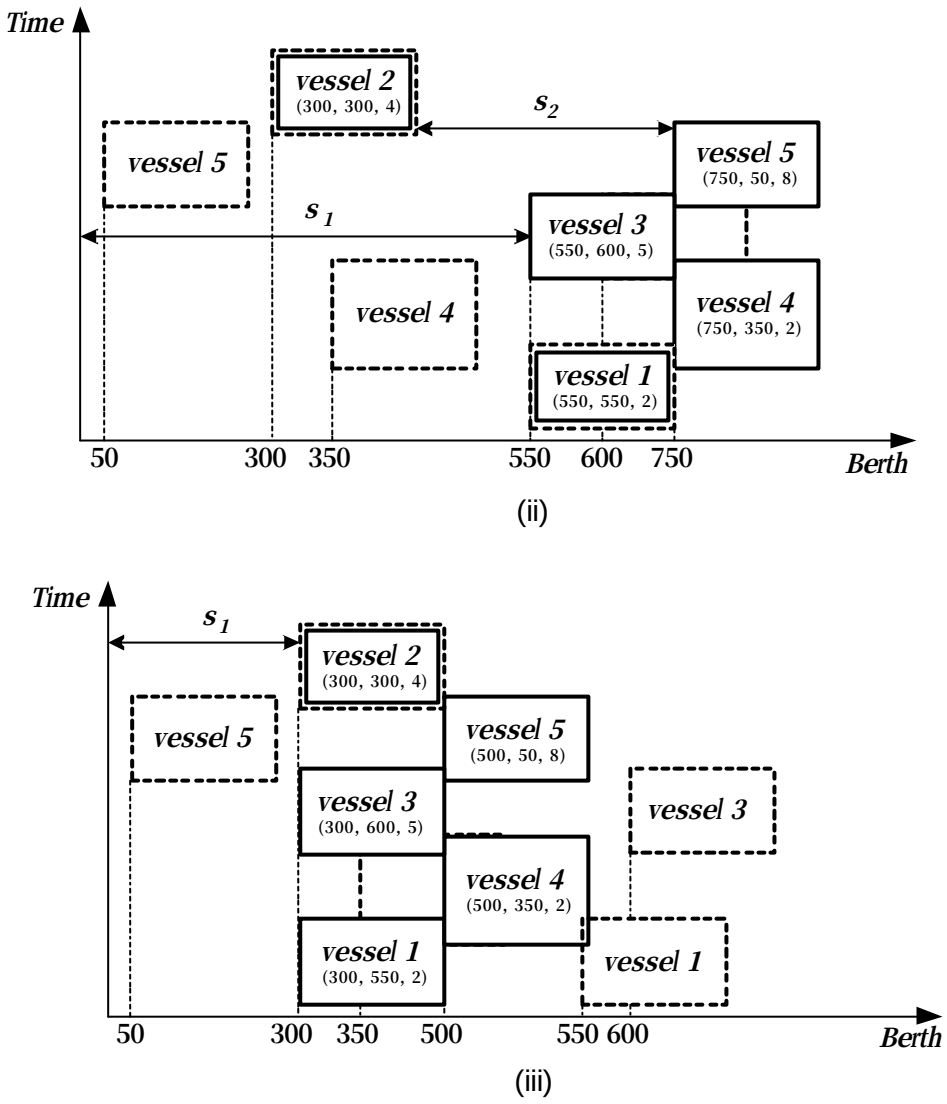


Fig. 7 An illustration of making a *x*-cluster stable

Fig. 7 illustrates the procedure to make a cluster stable. Consider the cluster consisting of vessels 3, 4, and 5 in Fig. 7-(i). It is easy to see that the cost of the cluster can be reduced by moving it in the left-hand direction. s_0=450, s_1=600, s_2=50, and so S={450, 600, 50}. The minimum element in S is 50. Thus, the cluster is moved in the left-hand direction by 50 as shown in Fig. 7-(ii). Note that a new member, vessel 1, is added into the cluster. The new cluster is still unstable against the movement in the left-hand direction. s_0=400, s_1=550, s_2=250, and S={400, 550, 250}. Since the minimum element in S is 250, the cluster is again moved in the left-hand direction by 250 as shown in Fig. 7-(iii). Now, the cluster becomes stable.

5 A heuristic solution algorithm for the berth planning

A heuristic algorithm is suggested in this section as follows:

Step 0: i=1. Sequence vessels in the increasing order of $ap_i / L + (1-a)d_i / \max_i\{d_i\}$, where $0 < a < 1$. Locate the first rectangle in the sequence at the minimum cost position.

Step 1: i=i+1. If i>N, stop. Otherwise, from the set of the right/upper corners of all the rectangles located so far, make the non-dominated set of right/upper corners of rectangles. A right/upper corner, (s_i, t_i) is said to be dominated if there exists another corner, (s_j, t_j), such that $s_i \le s_j$ and $t_i \le t_j$ and either of two inequalities strictly holds. Let the number of elements in the non-dominated corner set be m. Construct a stairway as shown in Fig. 8-(ii) whose profile is divided into m+1 segments by elements in the non-dominated corner set. Let j=0.

Step 2: j=j+1. If j>m+1, go to step 1. Otherwise, locate the i^{th} rectangle at the minimum cost position in the i^{th} profile segment. Identify x- and y-cluster that the i^{th} rectangle is associated with. Check the stability of the associated clusters. Include unstable clusters into the set of unstable clusters. Go to step 3.

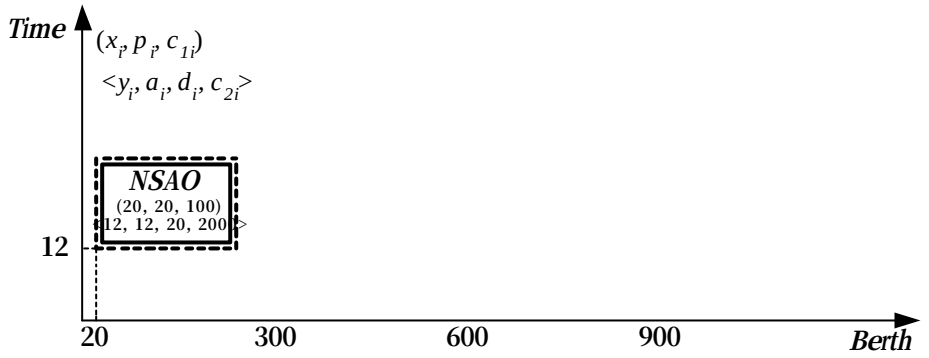
Step 3: Check if there remains an unstable x-cluster. If no, go to step 4. Otherwise, select a x-cluster that is unstable. Perform the procedure to make the x-cluster stable. In the process, one or more unstable y-cluster may be created. Include those unstable y-clusters into the set of unstable clusters. Go to the beginning of this step.

Step 4: Check if there remains an unstable y-cluster. If no, check if there remains an unstable x-cluster. If there remains no unstable x-cluster, go to step 2. If there remains an unstable x-cluster, go to step 3. If there still remains an unstable y-cluster, select an unstable y-cluster. Perform the procedure to make the y-cluster stable. In the process, one or more unstable x-cluster may be created. Include those x-clusters into the set of unstable clusters. Go to the beginning of this step.

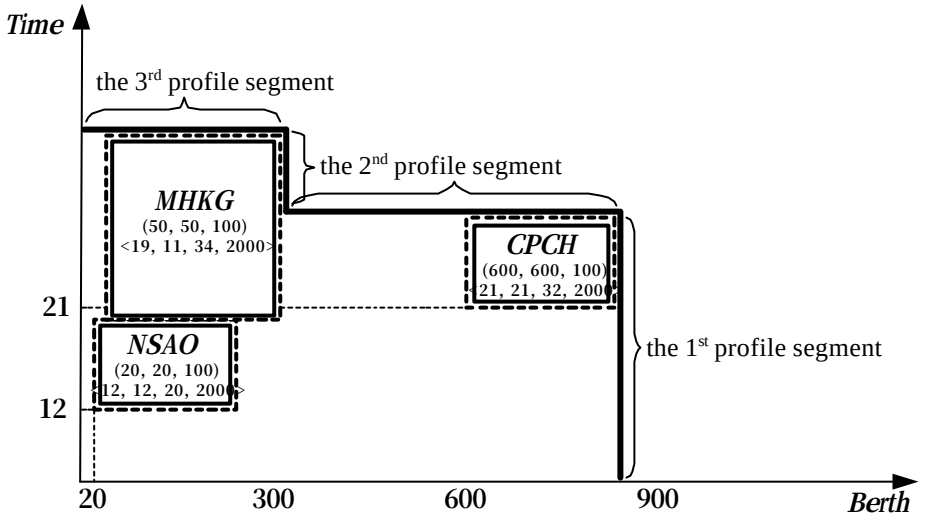
The heuristic algorithm is illustrated by using the data in Table 1. The penalty cost per hour was set to be \$4,000 for HHGL and \$2,000 for the other vessels, and the additional handling cost by the deviation of the berthing location from the most favorable position was set to be \$200 per meter for HHGL and \$100 for the other vessels. When α is set to be 0.5, the sequence of vessels in step 0 becomes NSAO, MHKG, CPCH, HHGL, MOKI, OECH, and ROYL.

Table 1. Input data of an example for the illustration

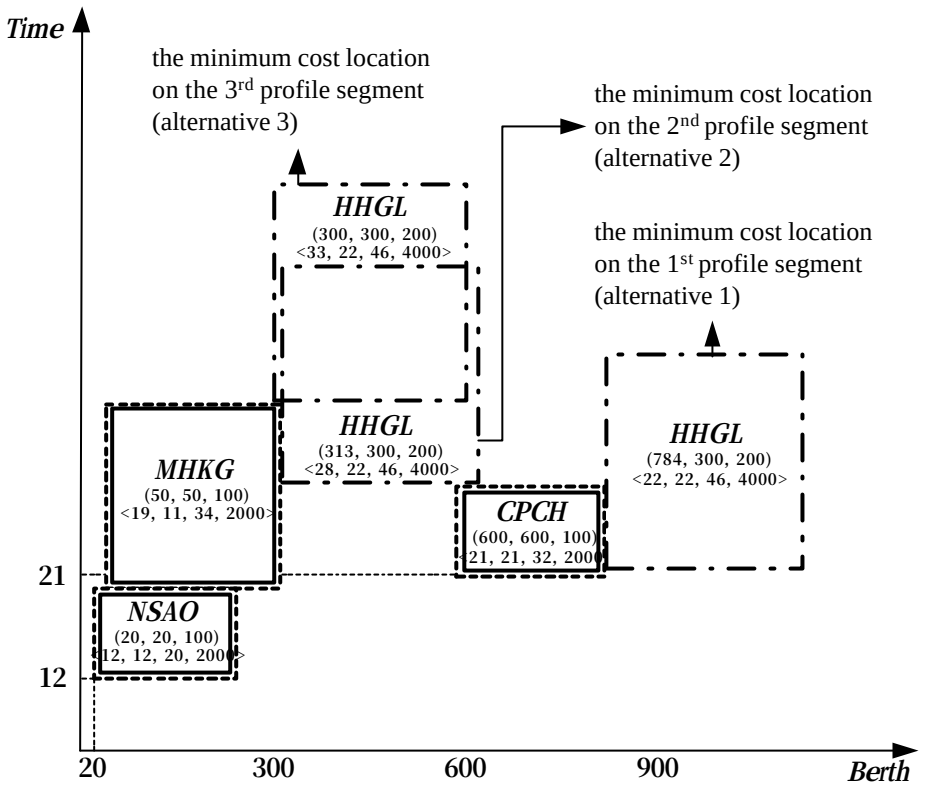
Vessels	l_i (m)	b_i (hours)	a_i	d_i	p_i (m)
NSAO	182	7	12	20	20
HHGL	295	22	22	46	300
ROYL	294	13	27	53	550
OECH	185	6	28	36	900
MHKG	263	14	11	34	50
CPCH	184	7	21	32	600
MOKI	262	18	5	30	900



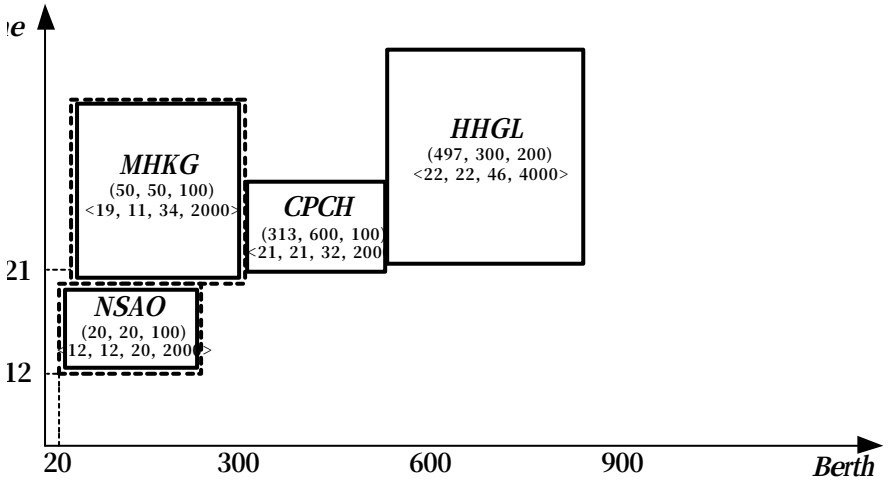
(i) The first iteration



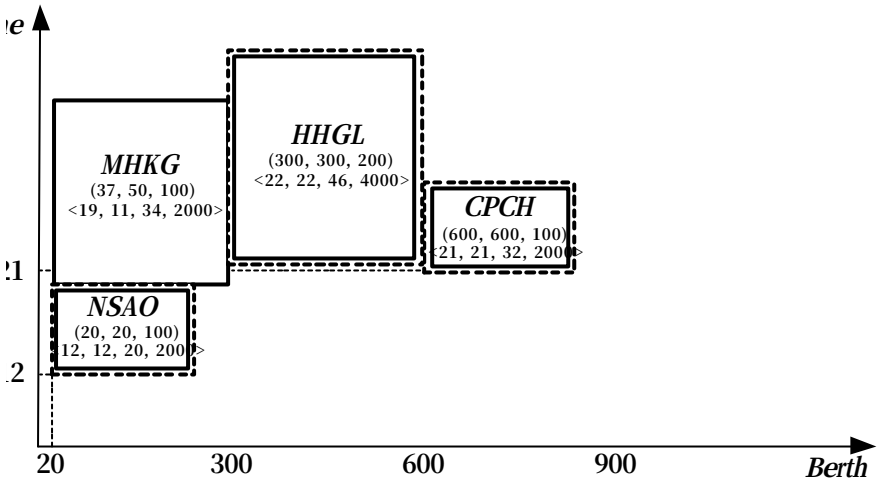
(ii) An illustration of profile segments



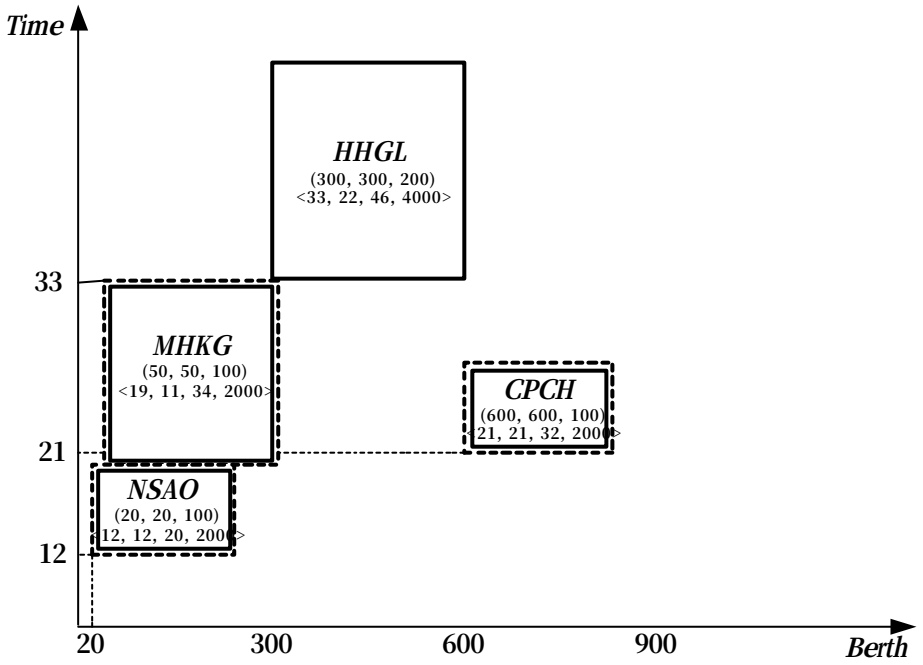
(iii) The minimum cost location of HHGL on each profile segment



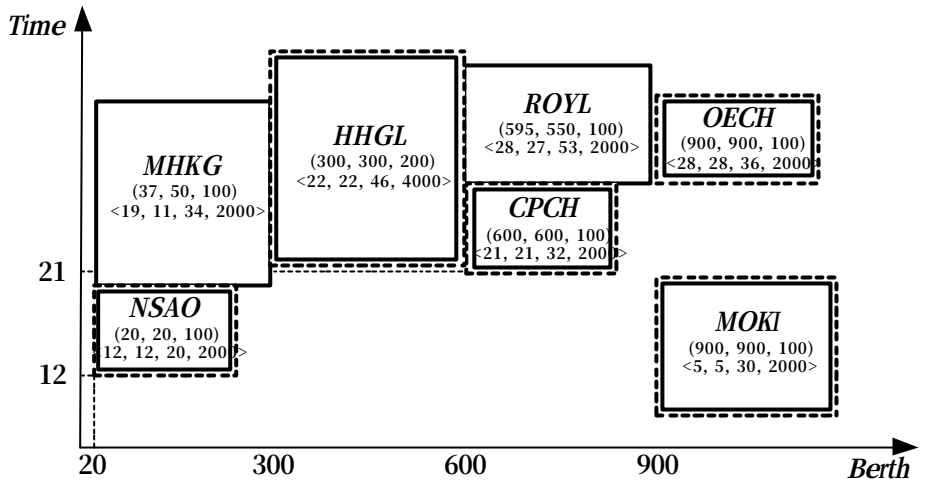
(iv) Alternative 1 after stabilization



(v) Alternative 2 after stabilization



(vi) Alternative 3 after stabilization



(vii) The final solution

Fig. 8 An illustration of the heuristic algorithm

Fig. 8 shows an illustration of the heuristic algorithm. The first vessel, NSAQ, is located on its most favorable location as shown in Fig. 8-(i). Suppose that NSAO, MHKG, and CPCH are located. Then, the next vessel to be positioned is HHGL. The profile of the stairway can be partitioned into three segments as shown in Fig. 8-(ii). Fig. 8-(iii) shows the initial location of HHGL on each profile segment before the stabilization. Fig. 8-(iv), (v), and (vi) show alternatives obtained, after making clusters stable. The increases in the cost resulting from the stabilization for three alternative layouts were \$68,100, \$1,300, and \$36,000. Thus, the solution in Fig. 8-(v) will be chosen. After all the iterations, the solution in Fig. 8-(vii) will be chosen as the final solution.

6 A numerical experiment

A numerical experiment was carried out to evaluate the performance of the algorithm in this study. Real data for the berth and sizes of vessels were collected from Pusan East Container Terminal (PECT) that is one of two largest terminals in Korea. Data for vessels were collected from January to May in 1999. The length of the berth of PECT is 1200m. The safety allowance between vessels at the berth is 20m and at least two hours are needed for departing of a vessel and berthing of another vessel. Table 2 illustrates a set of input data for 7 days.

Table 2. An illustrative input data

Vessels	l_i (m)	b_i (hours)	a_i	d_i	p_i (m)
NAMB	240	12	143	158	580
SLPR	260	9	87	102	230
ASIN	240	9	50	63	50
NARI	180	12	97	121	480
MSCM	210	19	4	24	900
UNSG	170	12	58	73	210
KTRY	180	13	53	77	610
DVOT	290	22	130	158	40
SPCH	190	7	132	143	30
BPDG	280	21	10	39	580
AMEG	270	15	142	163	900
HHHG	280	23	61	95	890
RSLT	290	13	117	141	760
MHKG	240	16	83	111	90
AGRA	280	13	55	77	480
ARIA	240	13	127	143	210
CLBG	290	20	134	162	160
SLLB	260	9	14	26	80

The arrival time and the requested departure time were generated randomly but the difference between the arrival time and the requested departure time was maintained between 1.0-2.0 times of the ship operation time. And p_i was also generated randomly within the range of the berth. First, in order to compare the performance of the heuristic algorithm in this study with that of the optimizing model, forty problems were solved by LINDO® for the formulation (2.2)-(2.11) and by the heuristic algorithm in this study, respectively. The planning horizon was set to be 72 hours. The reason for the restriction on the problem size was that the computational time for LINDO® exceeded a reasonable limit when the number of time became larger than 7 and the planning horizon exceeded 72 hours.

Table 3 shows the result of the comparison. For problems 1-20, c_{2i} was set to be \$12,000, while it was \$2,000 for problems 21-40. c_{1i} was set to be \$10 for problems 1-20, while it was \$100 for problems 21-40. For problems 1-20, the negative effect from delaying the departure time was considered much more serious than the deviation of the berthing position from the best berth location was. Because the penalty cost for the delayed departure is too high, the difference between the objective values by LINDO® and the heuristic algorithm was unexpectedly jumped up for three problems. However, when c_{2i} was set to be \$2,000, the results by the heuristic

algorithm were stabilized and near to that by LINDO®. Note that the optimal solution was obtained for 21 problems among 40 example problems.

Table 3. A comparison of the objective values by LINDO® and the heuristic algorithm

Problem number	Obj. value by the heuristic algorithm (A)	Obj. value by LINDO® (B)	(A-B) *100/B	Computation time of LINDO® (sec.)	Problem number	Obj. value by the heuristic algorithm (A)	Obj. value by LINDO® (B)	(A-B) *100/B	Computation time of LINDO® (sec.)
1	2,280	2,280	0	32	21	174,000	94,000	85	40
2	50	50	0	18	22	5,000	5,000	0	12
3	200	200	0	32	23	20,000	20,000	0	9
4	1,840	1,840	0	14	24	184,000	184,000	0	83
5	421,720	3,380	12,377	34	25	211,000	78,000	171	215
6	2,620	2,620	0	33	26	262,000	262,000	0	41
7	120	80	50	22	27	12,000	8,000	50	17
8	1,300	960	35	35	28	130,000	96,000	35	17
9	372,330	8,560	4,250	734	29	460,000	308,000	49	2,372
10	1,230	1,230	0	45	30	123,000	123,000	0	22
11	11,860	8,200	45	1,270	31	492,000	396,000	24	201
12	2,200	310	610	15	32	186,000	31,000	500	16
13	3,610	1,940	86	152	33	120,000	120,000	0	40
14	270	270	0	22	34	27,000	27,000	0	12
15	213,690	9,450	2,161	312	35	263,000	263,000	0	145
16	3,880	1,650	135	24	36	200,000	165,000	21	28
17	0	0	0	2	37	0	0	0	5
18	1,320	1,320	0	141	38	132,000	132,000	0	34
19	0	0	0	21	39	0	0	0	11
20	5,980	5,900	1	143	40	341,000	306,000	11	104
Average	52,325	2,512	988	155	Average	167,100	130,900	47	171

7 Conclusions

It is discussed how to determine the berthing time and position of arriving containerships when the berth is a bottleneck resource of the flow of containers in port container terminals. It is tried to satisfy simultaneously carrier's request on timing of berthing and minimize the delivery efforts between the marshaling yard and the apron during the ship operation. An integer linear program is formulated for the problem and solved using LINDO® package. The computational time of LINDO® increased rapidly when the number of vessels became higher than 7 and the length of the planning horizon exceeded 72 hours.

Some properties of the optimal solution were investigated. Based on the properties, a heuristic algorithm for the berth-planning problem was suggested. The performance of the heuristic algorithm was compared with that of the optimizing technique.

In practices, many planners consider the schedule of quay cranes simultaneously during the berth planning process. That is, the berthing time may vary according to the number of cranes assigned to the corresponding vessel. Although the crane scheduling issue was not incorporated into the model, it may be a promising issue for the further study.

8 References

Brown, G. G., Lawphongpanich, S., and Thurman, K. P. (1994), *Optimizing ship berthing*, Naval Research Logistics, 41, 1-15.

Francis, R. L., McGinnis Jr., R. F., and White, J. A. (1992), Facility Layout and Location: An Analytic Approach, Prentice-Hall, 189-199.

Lai, K. K., and Shih, K. (1992), *A study of container berth allocation*, Journal of Advanced Transportation, 26(1), 45-60.

Lim, A. (1998), *The berth planning problem*, Operations Research Letters, 22, 105-110.