Ocean Systems Engineering, Vol. 2, No. 4 (2012) 271-287
DOI: http://dx.doi.org/10.12989/ose.2012.2.4.271

Block layout method in the block stockyard based on the genetic algorithm

Myung-Il Roh*

School of Naval Architecture and Ocean Engineering, University of Ulsan, Mugeo-Dong, Ulsan, 680-749, Korea

(Received October 27, 2012, Revised December 1, 2012, Accepted December 10, 2012)

Abstract. Due to its large size, a ship is first divided into scores of blocks and then each block is constructed through various shops, such as the assembly shop, the painting shop, and the outfitting shop. However, each block may not be directly moved to the next shop and may be temporarily laid at a block stockyard because the working time in each shop is different from each other. If blocks are laid at the block stockyard without any planning, the rearrangement of the blocks by a transporter is required because the blocks have the different in and out time. In this study, a block layout method based on the genetic algorithm was proposed in order to minimize the rearrangement of the blocks in the block stockyard. To evaluate the applicability of the proposed method, it was applied to simple layout problems of the block stockyard. The result shows that the proposed method can yield a block layout that minimizes the total relocation cost of moving obstacle blocks in the block stockyard.

Keywords: block layout; block stock yard; transporter; genetic algorithm; optimization; shipbuilding

1. Introduction

1.1 Background

Due to its large size, a ship is first divided into scores of blocks and then each block is constructed through various shops, such as the assembly shop, the painting shop, and the outfitting shop. However, each block may not be directly moved to the next shop and may be temporarily laid on a certain place called ‘block stockyard’ because the working time in each shop is different from each other. If blocks are laid at the block stockyard without any planning, the rearrangement of the blocks by a transporter is required because the blocks have the different in and out time according to their production schedules. For example, if there exist blocks, called ‘obstructive blocks’, when taking out a certain block, called ‘due block’ from the block stockyard, we should first move them away by using the transporter and release the block. If obstacle blocks are increased, the retrieving time of the block and traveling distance by the transporter become long. As a result, the occupation time of the transporter due to the obstacle blocks increases and this consequently has a big impact on the productivity of the shipyard. However, if the blocks at the block stockyard are efficiently arranged considering their production schedules, that is, in and out time, the rearrangement of the

*Corresponding author, Professor, E-mail: miroh@ulsan.ac.kr,

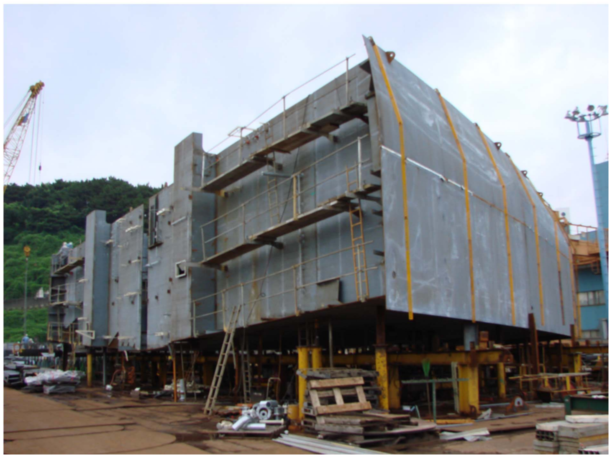
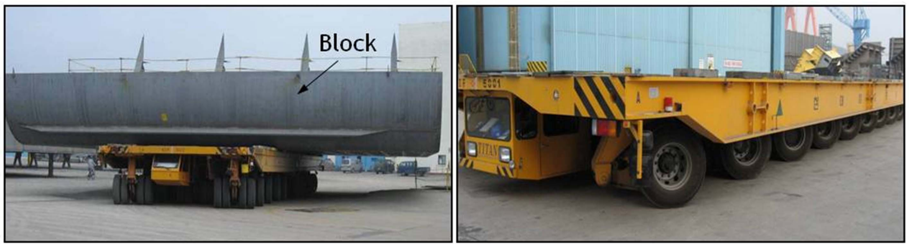


Fig. 1 Blocks at a blocks stockyard in the shipyard



(a) Transporter with loading (b) Transporter without loading

Specifications	• Length : 23.3 m • Breadth : 6.6 m • Height : Avg. 2.2 m (1.55 ~ 2.2 m, adjustable) • Lightweight : 126 ton • Speed : without loading 15 km/h, with loading 10 km/h • Number of wheels : 88
Purpose	Moving blocks, deck houses, main engines, large pipe equipments, etc.
Features	• Moving forward and backward, 360° at the current position • Two control rooms at the front and back • Two signalmen are required for ensuring against risks

Fig. 2 Example of a transporter in the shipyard

blocks can be minimized. Fig. 1 shows blocks at a block stock yard and Fig. 2 shows a transporter for moving the blocks in the shipyard.

1.2 Related works

Research related to the block layout method at the block stockyard in the shipbuilding industry is not prevalent, even though it is possible to increase the productivity of a shipyard and to reduce the building cost of a ship through efficient block stockyard management. Park and Seo (2006) proposed a heuristic method for block layout at the block stockyard. This method is to assign incoming blocks on empty cells in the blocks stockyard, reassign obstructive blocks on other empty cells, and retrieve a due block from the block stockyard with the objective of minimizing the number of obstructive block moves. Baek *et al.* (2006) developed a prototype system based on Park *et al.*'s method and applied it to a shipyard. The developed system embodied the visual function of monitoring the shipyard on the real-time and the interactive block layout function.

Related to this, some research on optimal crane operation in a container terminal has been conducted. Daganzo (1989, 1990) and Peterkofsky and Daganzo (1990) proposed a heuristic method for optimal management of a quay crane in a container terminal. Kim and Kim (1999) presented a method for optimal management of a transfer crane for a given working schedule and container loading condition. Similarly, Ng and Mak (2005) proposed a method for the optimal management of a transfer crane for loading and unloading work having different setup times. In addition, Kim and Park (1998) developed a method for determining the optimal storage space in order to minimize the number of relocations of containers when loading and unloading the containers. Kim and Kim (2007) proposed an optimal routing method to minimize the travel time of a transfer crane and the setup time at a yard bay in a container terminal. Lee *et al.* (2007) proposed a method based on the generic algorithm for determining the optimal storage space in order to minimize the number of operations of a transfer crane and the number of relocations of containers when loading and unloading the containers. Choi *et al.* (2004) and Kang *et al.* (2004) presented a heuristic method for the optimal layout of containers in a container terminal in order to minimize the number of relocations of containers.

However, there are limitations in applying these existing studies as they do not directly apply to the block layout at a block stockyard. In this study, a block layout method based on the genetic algorithm was proposed in order to minimize the rearrangement of the blocks at the block stockyard. An optimization problem for block layout in the block stockyard is mathematically formulated. To solve this problem, an optimization algorithm based on the genetic algorithm was implemented. Finally, to evaluate the applicability of the proposed method, it was applied to simple layout problems of the block stockyard.

2. Block layout problem in a block stockyard

2.1. Overview of the block layout problem

The goal of the block layout problem is to find an optimal block layout in the block stockyard of a shipyard. At this time, the purpose of optimal layout is to minimize the number of relocations and the traveling distance of blocks by the transporter. That is, when each block is arranged (come in or go out) in the block stockyard, the number of obstacle blocks and the traveling distance are minimized.

A block stockyard where working blocks are temporarily laid takes up much space in the shipyard. In general, the block stockyard is divided into some areas for the efficient management. In this study, it is assumed that the shape of the block stockyard is rectangular and the interior is divided into same sizes, as shown in Fig. 3 It is also assumed that the transporter moves in a straight line in order to move obstacle blocs in the block stockyard, as shown in Fig. 4. In fact, the transporter can only enter in a straight line considering existing blocks in the stockyard, the length and the radius of rotation of the transporter, and so on.

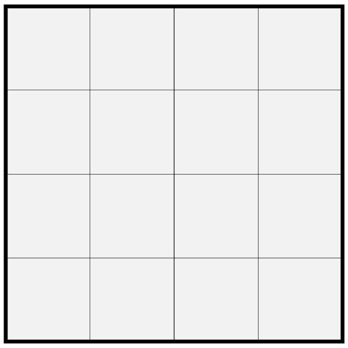


Fig. 3 Example of the block stockyard having 4 × 4 regions

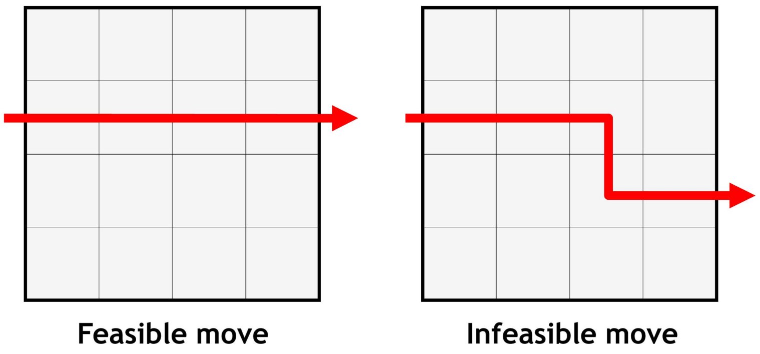


Fig. 4 Moving pattern of a transporter in the block stockyard in this study

The input data in this study given by a designer are indicated below for optimal block layout.
- Number of regions in the block stockyard
- Total number of blocks to be arranged in the block stockyard
- Period of block layout (start and finish time of block layout)
- Information on the blocks to be arranged in the block stockyard

Here, the detail information about the blocks is as follows.
- ID or name of blocks: It is assumed that the name is comprised of 5 letters (XXXXX). For example, S0101, D0102, E0203
- In and out time of blocks: It is assumed that the time is comprised of 12 numbers (YYYYMMDDHHMM). For example, '200902201030' means 10:30 AM at February 20, 2009.

Now, let's define some terminologies which are related to optimal block layout in this study. Any blocks are out of existence or some blocks (existing block) are already arranged in the block stockyard at the beginning. As time goes on, a new block, which is called 'incoming block', come in the block stockyard at a certain time, which is called 'in time' of the block. Then, a block, which is called 'release block', goes out at a certain time, which is called 'out time' of the block. In some cases, when a block comes in or goes out the block stockyard, one or more existing blocks, which are called 'obstacle blocks', should be temporarily moved. In this study, it is assumed that existing blocks except obstacle blocks are not moved.

Fig. 5 shows an example of the block stockyard layout in this study. As shown in Fig. 5, a block stockyard has 16 regions. Each two numbers (e.g., '1-9', '3-12', '16-25' etc.) written in the region mean not area lot numbers of the region but in and out priority of the block. The former number is the incoming order (the order of in time) and the latter number is the release order (order of out time) of the block considering in and out time of the blocks in the block stockyard. In other words, the small number is early inserted and released. For example, the block, which is allocated in the '1-9' region, came in the first and will go out the ninth. Of course, it is possible that the incoming order and release order are same. If the incoming and release order of the block are same, the in and out time can be different from each other. Thus, the block cannot come in and go out the block stockyard at the same time. That is, in this study, the block in the block stockyard is expressed not the ID or name but the in and out priority of the block.

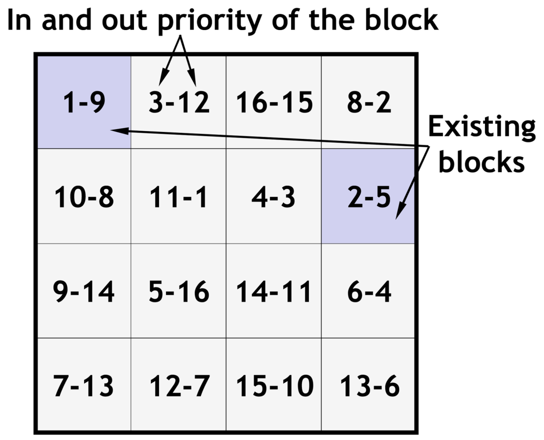


Fig. 5 Example of the block stockyard layout in this study

Let us return to Fig. 5. As shown in Fig. 5, the '11-1' block is inserted the eleventh and released the first. When this block is inserted in the block stockyard, the '3-12' block or the '10-8' block are obstacle blocks because they are already inserted. When this block is released from the block stockyard, they become obstacle blocks due to the same reason.

Like this, in the aspect of the '11-1' block, this block layout of Fig. 5 is not good because the obstacle blocks ('3-12' block or the '10-8' block) should be moved before incoming or releasing the '11-1' block. This will cause the cost and the purpose of this study is to reduce the cost, that is, to minimize the number of relocations and the traveling distance of blocks by the transporter in the block stockyard.

2.2 Mathematical formulation of the block layout problem

In this study, the block layout problem is mathematically formulated as an optimization problem to find an optimal block layout in the block stockyard. The following variables and symbols are used to formulate the problem.

- M: total cost of moving obstacle blocks in the block stockyard (relocation cost)
- I_{ij}: sum of traveling distance of obstacle blocks when the block whose position is the ith block layout in the block stockyard and the jth row is inserted in the given block layout
- s_{ij}: in time of the block whose position is the ith column and the jth row
- O_{ij}: sum of traveling distance of obstacle blocks when the block whose position is the ith column and the jth row is released from the given block layout
- e_{ij}: out time of the block whose position is the ith column and the jth row
- m, n: number of columns and rows in the block stockyard, respectively
- c: transportation cost per unit traveling distance
- S: start time of block layout
- E: finish time of block layout

Now, the block layout problem is mathematically formulated as follows.

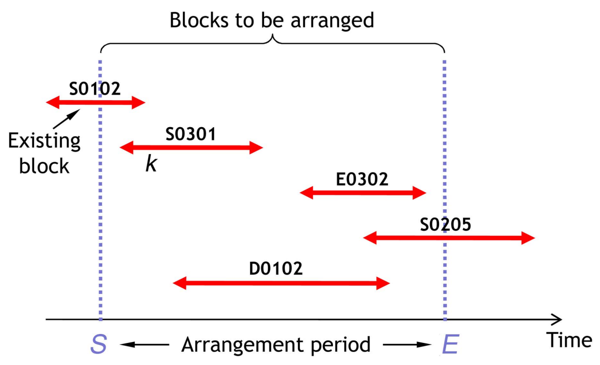


Fig. 6 Blocks having the different in and out time in the given period of block layout

Minimize $$M = \sum_{i=1}^{m} \sum_{i=1}^{n} c(I_{ij} + O_{ij}) \tag{1}$$

Subject to $$S \le s_{ij} \tag{2}$$

$$e_{ij} \le E \tag{3}$$

Here, Eq. (1) means that the relocation cost of moving obstacle blocks in the block stockyard should be minimized, and is referred as an objective function in the block layout problem.

Eqs. (2) and (3) means that the in time of each block should be after start time of block layout and the out time of each block should be before finish time, respectively, and are referred as constraints in the block layout problem. That is, they mean that all blocks are subject to be arranged in the given period of block layout.

Fig. 6 shows blocks having the different in and out time in the given period of block layout.

3. Block layout method in a block stockyard

3.1 Overview of the proposed method for block layout

In this study, the block layout method in the block stockyard is proposed based on the genetic algorithm. The genetic algorithm (GA) is classified as an evolutionary search and optimization technique that considers the design process to be evolutionary (Goldberg 1989, Davis 1991). The GA attempts to find the best solution by generating a collection ('population') of potential solutions ('individuals'). Through selection, crossover, and mutation operations, it is anticipated that more accurate solutions are generated from the current set of potential solutions. The selection operation is a process performed to select two individuals (parents), from the current population, for the next genetic operation. The crossover operation is a process performed to generate new individuals (children) from two individuals (parents) randomly selected by the selection operation. The mutation operation is a process performed to randomly change some part of chromosome. This iteration continues until the algorithm finds an acceptable solution. Details about the genetic algorithm can be found in many references, so it is omitted here. Fig. 7 shows a scheme of the proposed method

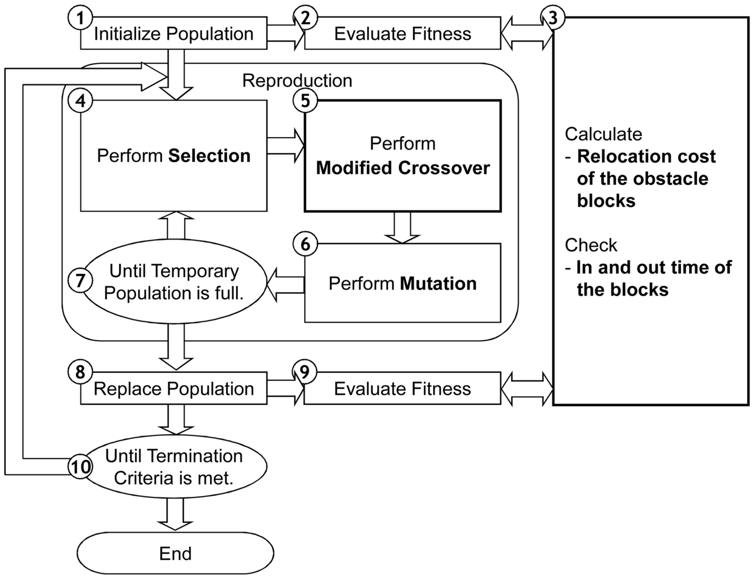

1 Initialize Population
2 Evaluate Fitness
3
Reproduction
4 Perform Selection
5 Perform Modified Crossover
6 Perform Mutation
7 Until Temporary Population is full.
Calculate
- Relocation cost of the obstacle blocks
Check
- In and out time of the blocks
8 Replace Population
9 Evaluate Fitness
10 Until Termination Criteria is met.
End

Fig. 7 Scheme of the proposed method for the block layout

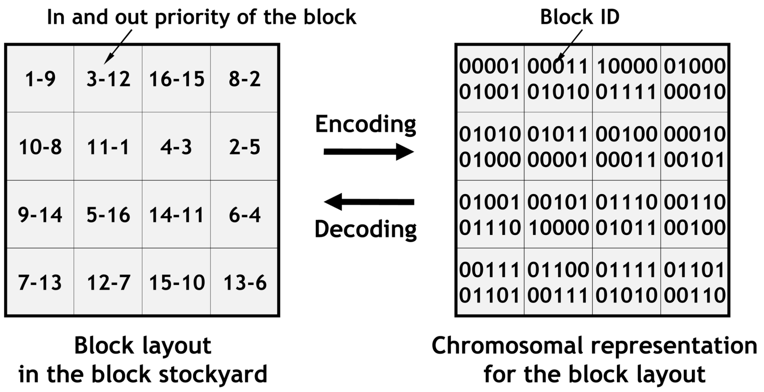


Fig. 8 Example of the block layout and the corresponding representation of the chromosome

for the block layout in this study.

3.2 Representation of the block layout in a block stockyard

The block layout in the block stockyard can be represented as a chromosome by an encoding process in the GA. The chromosome can likewise be represented as the block layout in the block stockyard by a decoding process. In this study, a method to model the block layout in the block stockyard in a chromosome of two-dimensional array type is used; the method is shown in Fig. 8.

3.3 Calculation of the rearrangement cost of block layout

As mentioned above, the objective function of the block layout problem is to minimize the

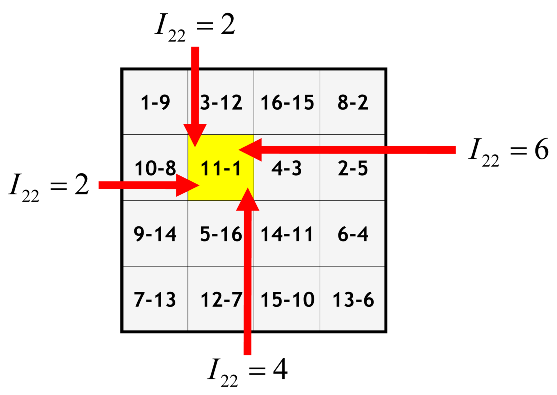


Fig. 9 Example of calculating the relocation cost for inserting '11-1' block in the block stockyard

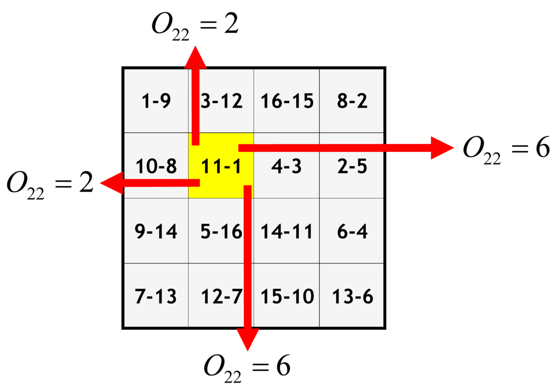


Fig. 10 Example of calculating the relocation cost for releasing the '11-1' block from the block stockyard

relocation cost of moving obstacle blocks when blocks are inserted in or released from the block stockyard. A method to calculate the relocation cost is represented as below.

Let us calculate the relocation cost when the '11-1' block is inserted in the block stockyard, as shown in Fig. 9. For convenience of explanation, it is assumed that the block starts to be released after all blocks are inserted.

When the '11-1' block is inserted in the block stockyard, a transporter can enter from all directions; north, south, east, and west. In case the '11-1' block is inserted from the north, the '3-12' block (obstacle block) is already laid in the block stockyard, therefore this block should be temporarily moved to outside of the stockyard. After that, the '11-1' block is inserted in its region, and then the '3-12' block should be relocated in the original position. Thus, when the '11-1' block is inserted, total traveling distance for the relocation of the '3-12' block is 2 because of in and out. Here, total traveling distance means a distance for moving the obstacle block ('3-12' block for the '11-1' block). In the same manner, in case the '11-1' block is inserted from the west and south, total traveling distance for the relocation of the '10-8' block and the '5-16' block is 2 and 4, respectively. When the '11-1' block is inserted from the east, total traveling distance is 6 because of 2 for the relocation of the '4-3' block and 4 for the relocation of the '2-5' block. For inserting the '11-1' block, therefore, the minimum traveling distance for the relocation is 2 from the north or west. The relocation cost of the obstacle block for inserting the '11-1' block whose position is the second column and the second row, I_{22}, amounts to 2 among total relocation cost (M of Eq. (1)).

Now, let us examine the case of releasing of the '11-1' block from the block stockyard. In case the '11-1' block is released from the north, the '3-12' block (obstacle block) is already laid in the block stockyard, therefore this block should be temporarily moved to outside of the stock yard. After that, the '11-1' block is released from its region, and the '3-12' block should be relocated in the original position. Thus, when the '11-1' block is released, total traveling distance for the relocation of the '3-12' block is 2 because of in and out. In the same manner, in case the '11-1' block is released from the west, total traveling distance for the relocation of the '10-8' block is 2. When the '11-1' block is released from the south, total traveling distance is 6 because of 4 for the relocation of the '5-16' block and 2 for the relocation of the '12-7' block. When the '11-1' block is released from the east, total traveling distance is 6 because of 4 for the relocation of the '4-3' block and 2 for the relocation of the '2-5' block. For releasing the '11-1' block, therefore, the minimum traveling distance for the relocation is 2 from the north or west. The relocation cost of the obstacle block for releasing the '11-1' block whose position is the second column and the second row, O_{22}, amounts to 2 among total relocation cost (M of Eq. (1)).

If above process is repeatedly applied to all blocks in the block stockyard, we can calculate the relocation cost of the given block layout. Below is a pseudo code for calculating the relocation cost of the given block layout.

```
SET $M$ equal to 0.
  FOR all locus of an individual (all blocks)
  CALCULATE obstacle blocks' distance for this block to the right side (mc$R$).
  CALCULATE obstacle blocks' distance for this block to the left side (mc$L$).
  CALCULATE obstacle blocks' distance for this block to the topmost (mc$T$).
  CALCULATE obstacle blocks' distance for this block to the bottommost (mc$B$).
  RETURN temporary relocation cost equal to min (mc$R$, mc$L$, mc$T$, mc$B$).
  CALCULATE total relocation cost $M = I + O$.
END FOR
```

3.4 Genetic operation for the optimal block layout

As mentioned above, the proposed method for block layout is based on the GA. In the GA, four genetic operations known as selection, crossover, inversion, and mutation are typically used to generate new individuals (children). In this study, these operations of the GA were improved to solve efficiently the block layout problem. These operations were incorporated into the proposed method.

3.4.1 Initialization of the population

The proposed method based on the GA is to find the best solution by generating a collection ("population") of potential solutions ("individuals"). It is expected that more optimal solutions will be generated from the current set of potential solutions. Each individual in the population represents block layout in the block stockyard. In this study, when the initial population is generated, each individual in the population is randomly generated. Below is to explain the process for generating an individual (block layout) when generating the initial population.

As shown in Fig. 11, it is assumed that the block stockyard has 4x4 regions and two blocks ('1-9' block and '2-5' block) are laid at (1, 1) (the first column and the first row), (2, 4) as existing blocks, respectively. Therefore, the maximum number of blocks to be arranged is 14. A new block, of course, cannot be inserted in the region where the existing block is already laid. For representing whether a new block can be inserted or not, each region has a property of 'true' or 'false'. For example, (1, 1) and (2, 4) regions where the existing blocks are already laid have a property of 'false'. Initially, the others of regions where a block is not laid have a property of 'true'. After inserting the block in that region, the property of the region is changed to 'false'. According to incoming order, all of 14 blocks are randomly inserted in regions having a property of 'true' then the block layout, that is, one individual of the population is completed. Fig. 10 shows an example of block layout by this process.

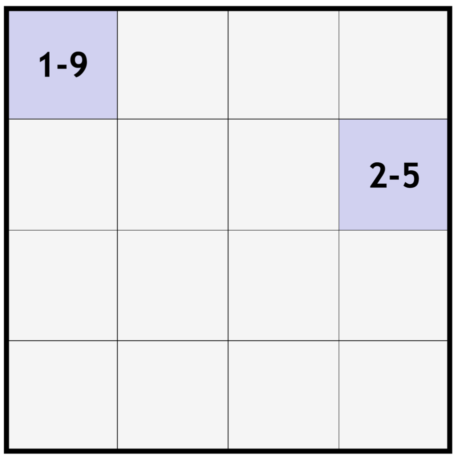


Fig. 11 Initial block layout having two existing blocks

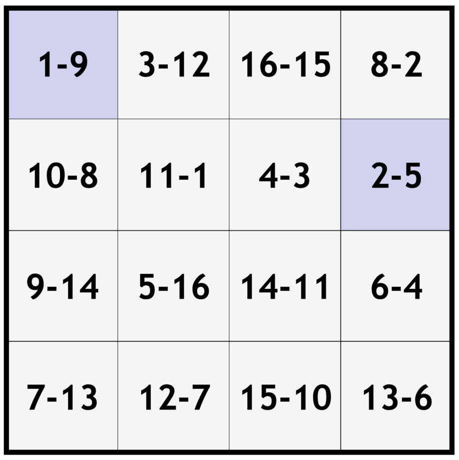


Fig. 12 Block layout having 16 blocks

3.4.2 Selection operation

The selection operation is a process performed to select two individuals (parents), from the current population, for the next genetic operation. A proportionate selection method is employed, as it is the most popular of the stochastic selection methods. The technique is sometimes called the roulette wheel selection method (Goldberg 1989, Jacobs 1996). Below is a pseudo code for selecting two individuals from the current population.

SUM all chromosome finesses in population (sum *S*).

GENERATE random number from interval [0, *S*] (*r*).

[LOOP] Go through the population and sum finesses from 0 (sum *s*). When the sum *s* is greater than r, stop and return the chromosome where you are.

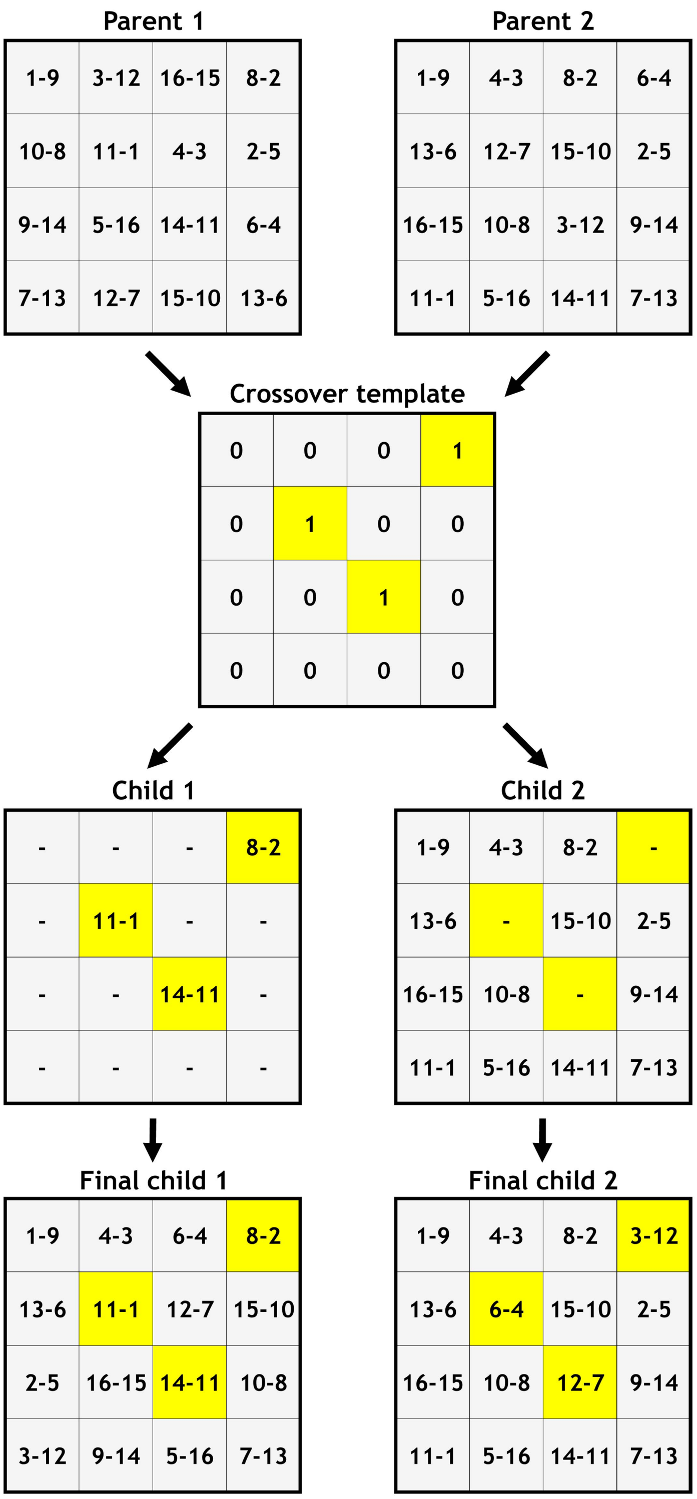
3.4.3 Crossover operation

The crossover operation is a process performed to generate new individuals ('children') from two individuals ('parents') selected by the selection operation. Two crossover operations, a modified crossover and a uniform order-based crossover, are used in this study.

(1) Uniform order-based crossover

The uniform order-based crossover operation, which was proposed by Davis (1991) and can be applied to a chromosome of two-dimensional array type, is used as the first crossover operation in this study. Let us see the uniform order-based crossover by using Fig. 13. Because genetic operations are applied to a chromosome in general, the chromosome of each individual should be represented as a binary meaning the block ID or name. However, the chromosome is represented here as the incoming and release order for convenience of explanation.

First, a chromosome ('crossover template') of two-dimensional array type having the same size of parents' chromosomes is generated and a property of each gene in the chromosome is randomly given 0 or 1. Then, the block of the first parent ('Parent 1') which corresponds to the gene having a property of '1' is copied to the corresponding position of the first child ('Child 1'). The remaining positions of 'Child 1' is copied with the blocks which are not arranged yet by referencing the



Parent 1

1-9	3-12	16-15	8-2
10-8	11-1	4-3	2-5
9-14	5-16	14-11	6-4
7-13	12-7	15-10	13-6

Parent 2

1-9	4-3	8-2	6-4
13-6	12-7	15-10	2-5
16-15	10-8	3-12	9-14
11-1	5-16	14-11	7-13

Crossover template

0	0	0	1
0	1	0	0
0	0	1	0
0	0	0	0

Child 1

-	-	-	8-2
-	11-1	-	-
-	-	14-11	-
-	-	-	-

Child 2

1-9	4-3	8-2	-
13-6	-	15-10	2-5
16-15	10-8	-	9-14
11-1	5-16	14-11	7-13

Final child 1

1-9	4-3	6-4	8-2
13-6	11-1	12-7	15-10
2-5	16-15	14-11	10-8
3-12	9-14	5-16	7-13

Final child 2

1-9	4-3	8-2	3-12
13-6	6-4	15-10	2-5
16-15	10-8	12-7	9-14
11-1	5-16	14-11	7-13

Fig. 13 Example of the uniform order-based crossover operation

chromosome (block layout) of the second parent ('Parent 2'). Through this step, we can generate 'Child 1' finally. Similarly, the block of the second parent ('Parent 2') which corresponds to the gene having a property of '0' is copied to the corresponding position of the second child ('Child 2'). The remaining positions of 'Child 2' is copied with the blocks which are not arranged yet by referencing the chromosome of 'Parent 1'. Through this step, we can generate 'Child 2' finally.

Below is a pseudo code for generating the first child by using the uniform order-based crossover operation.

```
GENERATE randomly a bit array having the same as parent.
COPY contents from 'Parent 1' to 'Child 1' wherever element contains '1'.
PUSH elements we get from previous step so that they appear in the same order they appear in 'Parent 2'.
COPY these permuted elements to gaps associated with a '0' on 'Child 1'.
```

(2) Modified crossover

In this study, it is assumed that existing blocks except obstacle blocks are not moved during optimization. However, if we perform the uniform order-based crossover operation, the existing blocks can be moved. To modify this problem, the modified crossover is used as the second crossover operation in this study. Of course, if there are no existing blocks, this operation is not necessary.

Fig. 14 shows an example of the modified crossover operation. Here, it is assumed that the '11-1' block is the existing block. As shown in Fig. 14, the position of this block is changed after performing the uniform order-based crossover operation, and it is modified by the modified

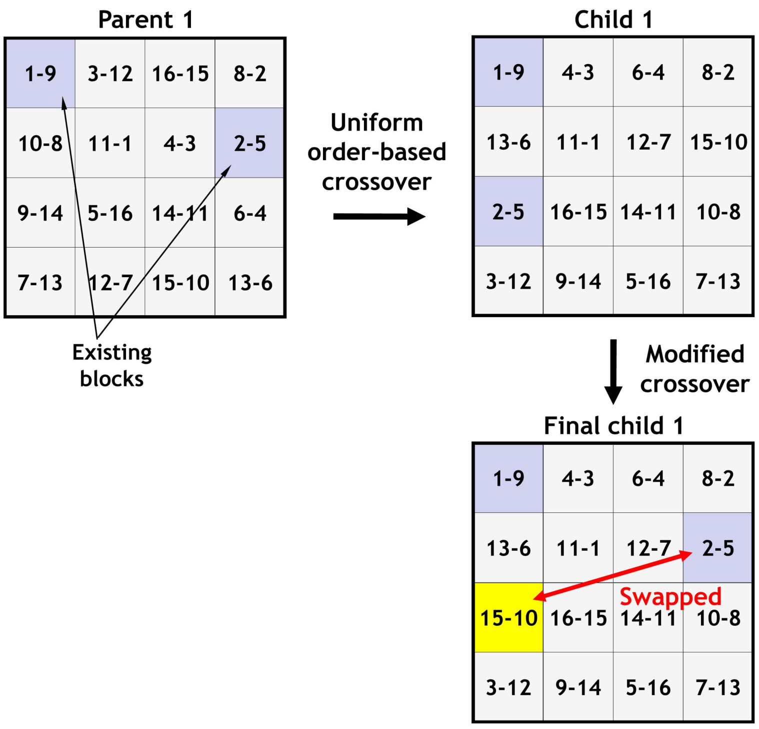


Fig. 14 Example of the modified crossover operation

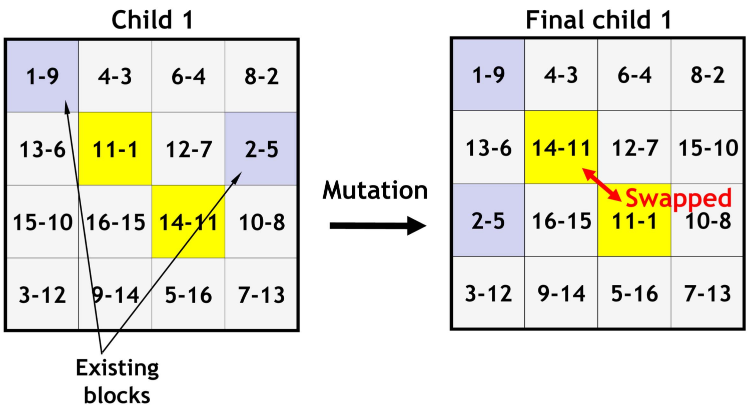


Fig. 15 Example of the mutation operation

crossover operation (position swap with the corresponding position of the block).

Below is a pseudo code for modifying the first child by using the modified crossover operation.

```
[LOOP] Scan all genes of chromosome of ‘Parent 1’.
 IF (Does this locus be getting fixed with an allele?)
   RETURN locus.
    IF (Is ‘Parent 1’ allele not equal to ‘Child 1’ allele at this locus?)
       LOOKP and RETURN locus on ‘Child 1’ that keep the same allele on ‘Parent 1’.

      CALL swap function.
      END IF
    END [LOOP]
```

3.4.4 Mutation operation

The mutation operation, which can be considered self-crossing, is used to increase population diversity. That is, the mutation operation provides the possibility of exploring portions of the design space that are not represented in the genetic makeup of the current population. The mutation operation is applied to each child generated from the crossover operation with a very low probability (typically $p_{mutation} = 0.01$ from Grefenstette’s study (Grefenstette 1986)). Two genes in each child are randomly selected and are exchanged with each other, as shown in Fig. 15. Of course, when performing this operation, the position of existing block should not be moved. That is, the existing blocks should not be selected for the mutation operation. Fig. 15 shows an example of the mutation operation applied to the first child.

Below is a pseudo code for mutating the first child by using the mutation operation.

```
DO
  CALL random function to get the first locus.
WHILE (locus is fixed.)
GET first locus.
```

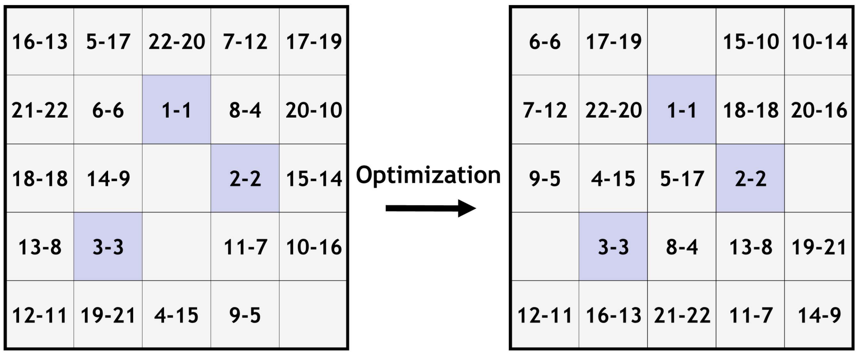
```
DO
  CALL random function to get the second locus.
WHILE (locus is fixed.)
GET second locus.
SWAP (first locus, second locus).
```

4. Application of the proposed method for block layout

To evaluate the efficiency and applicability of the proposed method for block layout, it was applied to a block layout problem having 22 blocks in the block stockyard whose size is 5x5. It is assumed that three blocks ('1-1', '2-2', and '3-3') among 22 blocks are existing blocks whose positions are not changed during optimization. Fig. 16 shows the block information for block layout

Block	Out time	Block	In time	Block	Out time	Block	In time
		1-1	2009-11/09-08:00			19-17	2009-11/15-10:00
		2-2	2009-11/09-09:00	6-6	2009-11/15-12:00		
		3-3	2009-11/09-10:00	7-11	2009-11/15-14:00	18-18	2009-11/15-14:00
		15-4	2009-11/09-11:00			21-19	2009-11/15-15:00
		17-5	2009-11/09-15:00	8-13	2009-11/15-15:00		
		6-6	2009-11/09-16:00	9-14	2009-11/15-17:00		
		12-7	2009-11/10-11:00	10-15	2009-11/16-11:00		
		4-8	2009-11/10-17:00			10-20	2009-11/16-12:00
		5-9	2009-11/11-10:00	11-12	2009-11/16-13:00		
		16-10	2009-11/11-11:00	12-7	2009-11/16-14:00	22-21	2009-11/16-14:00
		7-11	2009-11/11-17:00			20-22	2009-11/16-15:00
		11-12	2009-11/11-18:00	13-16	2009-11/17-10:00		
		8-13	2009-11/12-12:00	14-10	2009-11/17-12:00		
		9-14	2009-11/13-10:00	15-4	2009-11/17-14:00		
1-1	2009-11/13-12:00			16-20	2009-11/17-17:00		
		14-15	2009-11/13-13:00	17-5	2009-11/18-11:00		
2-2	2009-11/13-14:00			18-18	2009-11/19-10:00		
3-3	2009-11/13-16:00			19-17	2009-11/19-13:00		
4-8	2009-11/14-13:00			20-22	2009-11/19-16:00		
5-9	2009-11/14-14:00			21-19	2009-11/20-14:00		
		13-16	2009-11/14-15:00	22-21	2009-11/20-15:00		

Fig. 16 Block information for block layout



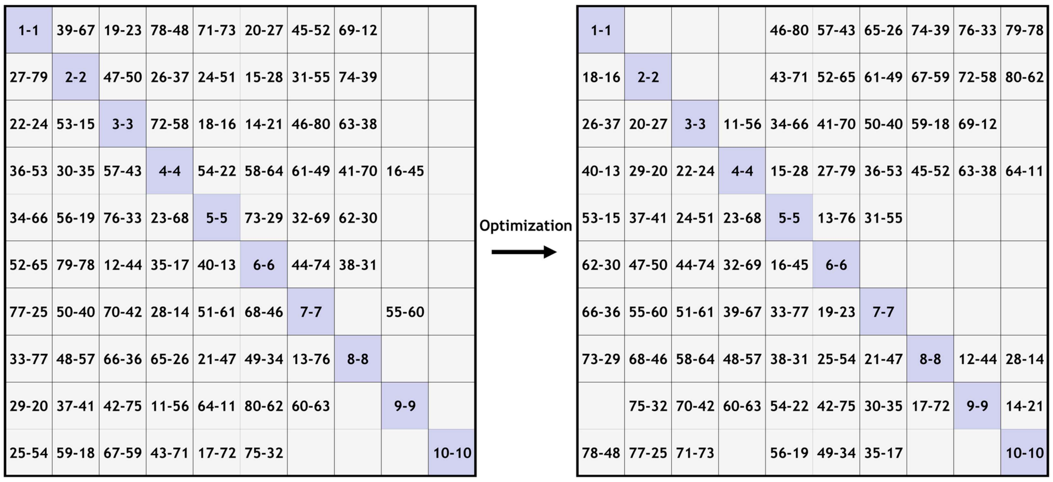
$M = \sum_{i=1}^{5}\sum_{j=1}^{5} c(I_{ij} + O_{ij}) = 2 + 6 = 8$

(a) Initial layout

$M = \sum_{i=1}^{5}\sum_{j=1}^{5} c(I_{ij} + O_{ij}) = 0 + 0 = 0$

(b) Final layout

Fig. 17 Block layout of the block stock yard of 5 × 5 before and after optimization using the proposed method


Optimization

1-1	39-67	19-23	78-48	71-73	20-27	45-52	69-12		
27-79	2-2	47-50	26-37	24-51	15-28	31-55	74-39		
22-24	53-15	3-3	72-58	18-16	14-21	46-80	63-38		
36-53	30-35	57-43	4-4	54-22	58-64	61-49	41-70	16-45	
34-66	56-19	76-33	23-68	5-5	73-29	32-69	62-30		
52-65	79-78	12-44	35-17	40-13	6-6	44-74	38-31		
77-25	50-40	70-42	28-14	51-61	68-46	7-7		55-60	
33-77	48-57	66-36	65-26	21-47	49-34	13-76	8-8		
29-20	37-41	42-75	11-56	64-11	80-62	60-63		9-9	
25-54	59-18	67-59	43-71	17-72	75-32				10-10

$$M=\sum_{i=1}^{10}\sum_{j=1}^{10}c(I_{ij}+O_{ij})=122+178=300$$

(a) Initial layout

1-1				46-80	57-43	65-26	74-39	76-33	79-78
18-16	2-2			43-71	52-65	61-49	67-59	72-58	80-62
26-37	20-27	3-3	11-56	34-66	41-70	50-40	59-18	69-12	
40-13	29-20	22-24	4-4	15-28	27-79	36-53	45-52	63-38	64-11
53-15	37-41	24-51	23-68	5-5	13-76	31-55			
62-30	47-50	44-74	32-69	16-45	6-6				
66-36	55-60	51-61	39-67	33-77	19-23	7-7			
73-29	68-46	58-64	48-57	38-31	25-54	21-47	8-8	12-44	28-14
	75-32	70-42	60-63	54-22	42-75	30-35	17-72	9-9	14-21
78-48	77-25	71-73		56-19	49-34	35-17			10-10

$$M=\sum_{i=1}^{10}\sum_{j=1}^{10}c(I_{ij}+O_{ij})=0+28=28$$

(b) Final layout

Fig. 18 Block layout of the block stock yard of 10×10 before and after optimization using the proposed method

of this problem.

Fig. 17 (a) shows an initial block layout before optimization. In the initial block layout, the relocation cost is 8 if we assume the transportation cost per unit traveling distance (*c* of Eq. (1)) is 1. However, as shown in Fig. 17 (b), we can see that the optimal block layout having the relocation cost of 0 was obtained after optimization.

Next, the proposed method was applied to a block layout problem having 80 blocks in the block stockyard whose size is 10x10. It is assumed that ten blocks ('1-1', '2-2', …, and '10-10') among 80 blocks are existing blocks whose positions are not changed during optimization. Fig. 18 (a) shows an initial block layout before optimization. In the initial block layout, the relocation cost is 300 if we assume the transportation cost per unit traveling distance (*c* of Eq. (1)) is 1. However, as shown in Fig. 18 (b), we can see that the optimal block layout having the relocation cost of 28 was obtained after optimization.

5. Conclusions

In this study, a block layout method based on the GA was proposed and implemented to minimize the relocation cost of moving obstacle blocks, that is, the rearrangement of the blocks in the block stockyard. First, an optimization problem for block layout in the block stockyard was mathematically formulated. To solve this problem, an optimization algorithm based on the GA was implemented. Finally, to evaluate the applicability of the proposed method, it was applied to simple layout problems of the block stockyard. The result shows that the proposed method can yield a block layout that minimizes the total relocation cost of moving obstacle blocks in the block stockyard.

In this study, it is assumed that the shape of the block stockyard is rectangular, the interior is divided into same sizes, and only one block is inserted in each region of the block stockyard. It is also assumed that the transporter moves in a straight line in order to move obstacle blocs in the block stockyard. Thus, in the future, it is necessary to study how to solve these constraints. In addition, to increase the efficiency and applicability of the proposed method, it should be improved through application to more complicated problems.

Acknowledgments

This work was supported by:
(a) Industrial Strategic Technology Development Program (10035331, Simulation-based Manufacturing Technology for Ships and Offshore Plants) funded by the Ministry of Knowledge Economy, Republic of Korea
(b) Global Leading Technology Program of the Office of Strategic R&D Planning (OSP) funded by the Ministry of Knowledge Economy, Republic of Korea (No. 10042556-2012-11)
(c) New & Renewable Energy of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) funded by the Ministry of Knowledge Economy, Republic of Korea (No. 20124030200110)

References

Baek, T.H., Kim, J.O., Lee, S.H., Min, S.G., Ha, S.J., Chung, M.Y., Lee, B.Y., Choi, T.H. and Park, C.K. (2006), "Assembly block operations management at shipyard: a case of hyundai heavy industries", *Proceedings of the 2nd PAAMES and AMEC 2006*, Cheju, Korea.

Choi, Y.J., Oh, M.S., Kang, J.H., Jun, S.M., Ryu, K.Y. and Kim, G.H. (2004), "Comparison of the performance of heuristics of the determination of the transformation position to minimize the retreatment of container", *Proceedings of the Annual Spring Meeting on the Korea Intelligent Information System Society*, Seoul, Korea.

Daganzo, C.F. (1989), "The crane scheduling problem", *Transport. Res. B - Mech.*, **23**(3), 159-175.

Daganzo, C.F. (1990), "Crane productivity and ship delay in ports", *Transport. Res. Rec.*, **1251**, 1-9.

Davis, L. (1991), *Handbook of genetic algorithms*, New York: Van Nostrand-Reinhold.

Goldberg, D.E. (1989), *Genetic algorithms in search*, Optimization and Machine Learning, Reading, MA: Addison Wesley.

Grefenstette, J. (1986), "Optimization of control parameters for genetic algorithms", *IEEE T. Syst. Man. Cy.*, **16**(1), 122-128.

Jacobs, S. (1996), "On genetic algorithms for the packing of polygons", *European J. Operat. Res.*, **88**(1), 165-181.

Jang, J.H., Ryu, K.Y. and Kim, G.H. (2004), "An efficient method for the relocation of container in a container terminal", *Proceedings of the Annual Autumn Meeting on the Korea Intelligent Information System Society*, Seoul, Korea.

Kim, H.G. and Kim, C.H. (2007), "A study on the optimal routing problem for a transfer crane", *Proceedings of the annual spring meeting on the Korean*, *Operations Research and Management Science Society/Korean Institute of Industrial Engineers*, Jinju, Korea.

Kim, K.H. and Kim, K.Y. (1999), "An optimal routing algorithm for a transfer crane in port container terminals", *Transport. Sci.*, **33**(1), 17-33.

Kim, K.H. and Park, K.T. (1998), "A dynamic space allocation method for outbound containers in carrier-direct system", *Proceedings of the 3rd annual international conference on industrial engineering theories Applications and Practice*, Hong Kong.

Lee, J.W., Hong, D.H. and Chung, T.C. (2003), "A slot assignment method for export containers in the yard using genetic algorithm", *Proceedings of the Annual Spring Meeting on the Korean Institute of Information*

Scientists and Engineers, Jeju, Korea.

Ng, W.C. and Mak, K.L. (2005), “An effective heuristic for scheduling a yard crane to handle jobs with different ready times”, *Eng. Optimiz*., **37**(8), 867-877.

Park, C.K. and Seo, J.Y. (2006), “A case study on assembly block operations management at shipyard”, *T. Korean Operat.Res. Man. Sci. Soc*., **23**(2), 175-185.

Peterkofsky, R.I. and Daganzo, C.F. (1990), “A branch and bound solution method for the crane scheduling problem”, *Transport. Res. B. – Mech*., **24**(3), 159-172.