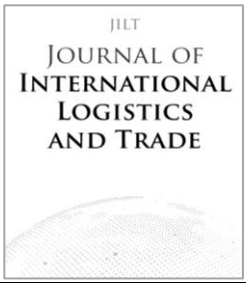
Journal of International Logistics and Trade, Vol. 15, No. 3, 2017, 83-90



Journal of
International Logistics and Trade

Heuristic Algorithms for Assigning Ship Assembly Blocks to Storage Locations at a Shipyard

Kwang-Tae Kim[a], Hwa-Joong Kim[b*], Seong-Hoon Choi[c]

[a] School of Industrial Management Engineering, Korea University, Seoul, Korea
[b] Asia Pacific School of Logistics, Inha University, Incheon, Korea
[c] Department of Management Engineering, Sangmyung University, Chungcheongnam-do, Korea

ARTICLE INFO

Article history:
Received 27 November 2017
Accepted 11 December 2017

Keywords:
Shipyard planning
Storage location assignment
Heuristics
Tabu search

ABSTRACT

We consider the problem of assigning ship assembly blocks to storage locations at a ship yard while allowing one-way movement of the blocks. The objective is to minimize the number of blocks obstructing other blocks' movements. We show that the integer program in a previous study contains errors in identifying obstructive blocks and therefore, suggest another integer program without the error. Also, we suggest construction heuristics and a tabu search algorithm using properties characterizing optimal solutions of subproblems. Computational experiment results indicate that our heuristics outperform the existing algorithm in terms of solution quality.

1. Introduction

The shipbuilding process consists of design, steel cutting, assembly, pre-outfitting, painting, erection, launching, outfitting, sea trial, and delivery, which is illustrated in Figure 1. In the design process, a large ship is split into divisions and the divisions are further split into blocks as a basic unit of the ship. In the assembly process, an assembly block is made by welding steel plates. The assembly block, as work-in-process inventory, goes through several operations such as pre-outfitting and painting process by transporters at the shipyard. Throughout these processes, assembly blocks are temporarily stored at a stockyard prior to the next process being performed, which is illustrated as bolded arrows in Figure 1. According to Park et al. (2007), Hyundai Heavy Industries, one of the biggest shipbuilders in the world, moves approximately 500 blocks daily, including an estimated 130 of these block transfers involve unproductive work. These unnecessary moves are generally incurred when an assembly block rests in the moving path of the transporter and the transporter must move the obstructive blocks to empty locations, in order to access other assembly blocks at the stockyard. This delays the subsequent block processes, and hence, increasing the total processing time and operational costs.

In this context, this paper considers the problem of assigning assembly blocks to storage locations at a stockyard over a planning horizon while minimizing the number of blocks obstructing the movement of other blocks., which occurs when a block is in the way of the other blocks. A series of papers have addressed this problem in the recent literature (Park et al. 2007, Park and Seo, 2009a, 2009b, 2010). Park et al. (2007) examined this problem and provided a detailed description of the Hyundai Heavy Industries case. They allowed storage and retrieval of the blocks in four directions and suggested a heuristic algorithm. Later, Park and Seo (2009a) suggested a non-linear program and linearized the non-linear constraint to convert the model into an integer program and revised their previous heuristic.

Park and Seo (2009b) considered the problem of one directional storage and retrieval operations, and suggested a genetic algorithm with a sophisticated crossover rule. More recently, Park and Seo (2010) reconsidered the four directional problem considered in Park et al. (2007) and Park and Seo (2009a). In their paper, they extended the genetic

* Corresponding author: Asia Pacific School of Logistics, Inha University, 100 Inha-ro, Nam-gu, Incheon 22212, Korea
Email: hwa-joong.kim@inha.ac.kr

algorithm in Park and Seo (2009b) to consider the four directional operations and compared the genetic algorithm with the heuristic in Park and Seo (2009a).

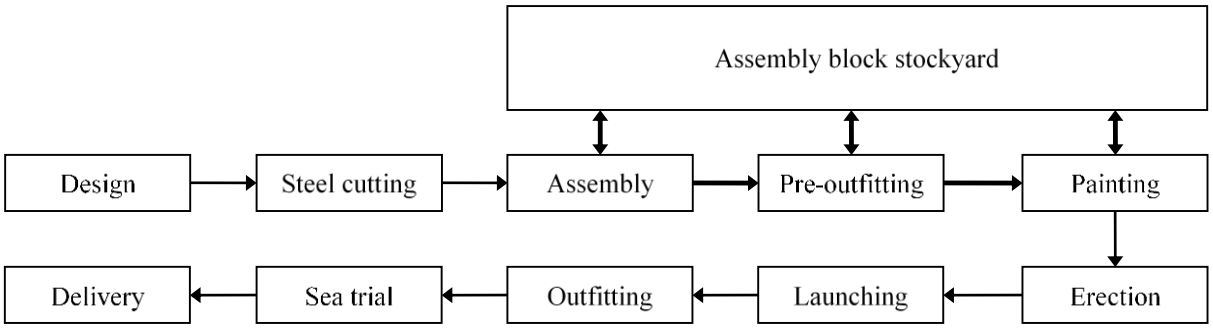


Figure 1. Shipbuilding process

Our study revisits the one-directional problem considered in Park and Seo (2009b) since we found that the integer program of Park and Seo (2009b) has an error in counting obstructive blocks, which is described in detail in the next section. Therefore, we suggest an alternative integer program by eliminating this error. In addition, we suggest two construction heuristics and a tabu search algorithm. The construction heuristic algorithms used to obtain initial solutions are developed using simple properties of the problem and the tabu search algorithm was suggested to improve the initial solutions.

This paper is organized as follows: Section 2 defines the problem, explains the counting error in the previous model and suggests an alternative integer program without this error. Section 3 describes the construction heuristics along with simple properties. Section 4 describes the tabu search algorithm used to improve the solutions of the construction heuristics. Section 5 summarizes computational test results concerning randomly generated test instances. The final section summarizes the research and proposes suggestions for future research.

2. Problem description, existing model's error and new integer program

2.1 Problem description

For a formal description of the problem, we begin with an illustration of the ship assembly block stockyard. Figure 2 shows the schematic drawing of an assembly block stockyard consisting of storage locations with 4 rows × 5 columns (= 20 storage locations), each of which can store one assembly block. Due to the enormous weight of the assembly blocks, ranging between 100 and 300 tons (Park et al. 2007), several blocks cannot be stored at the same storage location, i.e., placing one block on the top of another block. In Figure 2, arrows represent accessible moving paths for a transporter of assembly blocks. The storage and retrieval periods of the assembly blocks are preliminarily determined by the design and assembly schedules as in Park and Seo (2009b). However, the storage locations of the assembly blocks should be determined carefully in order to minimize unnecessary block moves. Unnecessary movement of blocks occur when blocks obstruct the movement of another block in the storage/retrieval operation, and hence, the obstructive blocks should be removed from their original locations.

Given storage and retrieval periods of the assembly blocks, the problem considered in this paper is determining the storage locations of the assembly blocks for minimizing the total number of obstructive blocks over a planning horizon. We assume that if blocks obstruct the movement of another block, the obstructive blocks should be returned at their original locations after finishing the block's move as in Park and Seo (2009b).

We summarize the notations used throughout this paper below.

Sets and parameters

I set of assembly blocks, $i \in I$
J set of storage locations, $j, k \in J$
T set of planning periods, $t \in T$
S_t set of blocks scheduled to be stored in period t
R_t set of blocks scheduled to be retrieved in period t
C set of columns, $c \in C$
D set of rows, $d \in D$
J_c set of storage locations in column c
P_j set of storage locations placed before location j out of the locations in the column location j is located

r_j retrieval period of block *i*
s_j storage period of block *i*
$|\bullet|$ cardinality of set •

Decision variables
X'_{ijt} 1 if block *i* is assigned to storage location *j* in period *t*, and 0 otherwise
X_{ij} 1 if block *i* is assigned to storage location *j*, and 0 otherwise
Y_{jt} 1 if a block exists in storage location *j* in period *t*, and 0 otherwise
Z_{ct} maximum number of obstructive blocks incurred in column *c* in period *t*

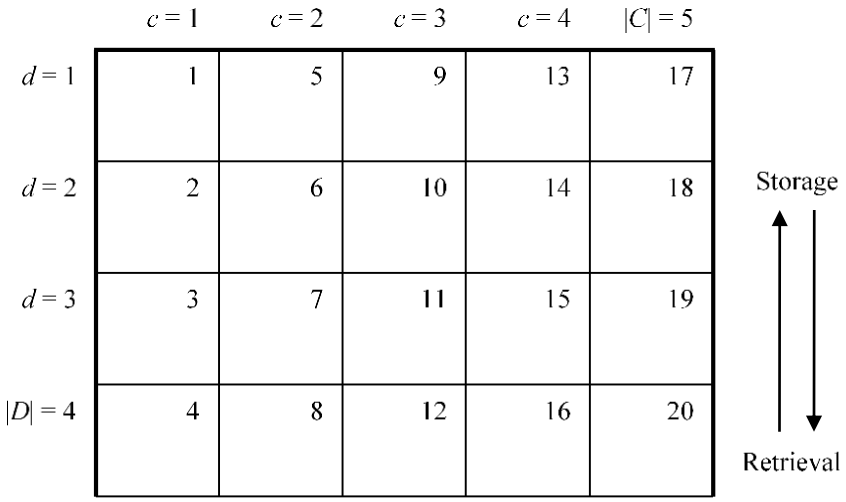


Figure 2. A schematic drawing of block stockyard

2.2 Existing model's error

This subsection illustrates the counting error in Park and Seo (2009b)'s model (called the PS model hereafter). The PS model calculates the number of obstructive blocks using the following non-linear function. In fact, the PS model used a different form of function but we modified the function as follows to use the notation above.

$$\sum_{t\in T}\sum_{i\in S_t\cup R_t}\sum_{j\in J} X'_{ijt}\sum_{k\in P_j} Y_{kt} \tag{1}$$

In the PS model, if block *i* is stored in location *j*, X'_{ijt} is set to 1 in the storage and retrieval periods, and 0 in the other periods. Y_{jt} is set to 1 in the periods from storage until retrieval and set to 0 in the other periods. For example, block *i* is scheduled to be stored in period 1 and retrieved in period 5 at location *j*. Then, decision variables are expressed as: $X'_{ij1}=1$, $X'_{ij5}=1$, and $X'_{ijt}=0$ for t = 2, 3, 4, 6, 7,…, *T*, $Y_{j2} = Y_{j3} = Y_{j4} = 1$, and $Y_{jt} = 0$ for *t* = 1, 5, 6,…, *T*.

For illustrating the counting error, an example is illustrated in Figure 3. Suppose that there are two blocks a and b to be stored in storage location 1 and 2 in period *t*, respectively, and block *c* exists in storage location 3, as shown in Figure 3. Hence, block *c* in location 3 should be moved to an empty location; i.e., block c is an obstructive block. The number of obstructive blocks in this example is 1 since the two blocks can be stored by moving block *c* only once. However, the equation (1) generates two obstructive blocks in period t as follows:

$$\sum_{i\in S_t}\sum_{j\in J} X'_{ijt}\sum_{k\in P_j} Y_{kt} = X'_{a1t}\cdot(Y_{2t}+Y_{3t}+Y_{4t}) + X'_{b2t}\cdot(Y_{3t}+Y_{4t}) = 2,$$

since in the above example,

S_t = {a, b}, J = {1, 2, 3, 4}, P_1 = {2, 3, 4}, P_2 = {3, 4}, P_3 = {4}, P_4 = {},
$X'_{a1t}=1$, $X'_{a2t}=X'_{a3t}=X'_{a4t}=0$, $X'_{b2t}=1$, $X'_{b1t}=X'_{b3t}=X'_{b4t}=0$, and
Y_{3t} = 1, $Y_{1t} = Y_{2t} = Y_{4t}$ = 0.

This contradictory case leads to the conclusion that the PS model cannot generate an optimal solution for the problem.

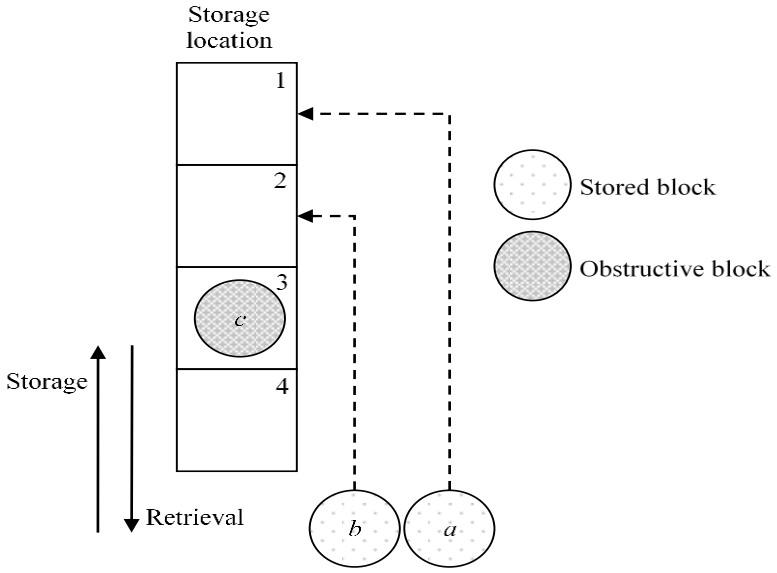


Figure 3. An example of the storages and blocks

2.3 A new integer program

To accurately determine the number of obstructive blocks, we use the maximum number of obstructive blocks in a column in a period. In the example above, each stored block requires the removal of block *c*, and the maximum number of obstructive blocks is 1 which is the same as the simple calculation. We use X_{ij}, instead of X'_{ijt} used in the PS model, in order to reduce the number of decision variables.

The new integer program is given below.

[P] Minimize $\sum_{c\in C}\sum_{t\in T} Z_{ct}$ (2)

subject to

$\sum_{j\in J} X_{ij} = 1$ for $i \in I$ (3)

$\sum_{i\in S_t} X_{ij} + Y_{jt} \le 1$ for $j \in J$ and $t \in T$ (4)

$\sum_{t=s_i+1}^{r_i-1} Y_{jt} \ge (r_i - s_i - 1)\cdot X_{ij}$ for $i \in I$ and $j \in J$ (5)

$\sum_{k\in P_j} Y_{kt} - |P_j|\cdot(1-\sum_{i\in S_t} X_{ij}) \le Z_{ct}$ for $c \in C$, $j \in J_c$, and $t \in T$ (6)

$\sum_{k\in P_j} Y_{kt} - |P_j|\cdot(1-\sum_{i\in R_t} X_{ij}) \le Z_{ct}$ for $c \in C$, $j \in J_c$, and $t \in T$ (7)

$X_{ij} \in \{0,1\}$ for $i \in I$ and $j \in J$ (8)

$Y_{jt} \in \{0,1\}$ for $j \in J$ and $t \in T$ (9)

$Z_{ct} \ge 0$ and integer for $c \in C$ and $t \in T$ (10)

The objective function (2) to be minimized implies the total number of obstructive blocks over the planning horizon. Constraint (3) requires that a block must be stored at exactly one storage location. Constraint (4) ensures that a block can only be assigned to an empty storage location; i.e., a block to be stored in a period cannot be assigned to a storage location occupied by another block. Constraint (5) guarantees that if a block to be stored is assigned to a storage location, the storage location must be occupied from its storage period to the period just before its retrieval period. Constraints (6) and (7) determine the maximum number of obstructive blocks in a column in a period when a block is stored or retrieved at the location. Finally, the remaining constraints (8), (9), and (10) represent restrictions on decision variables.

3. Construction heuristics

The survey was constructed during June to September 2016, interviews and questionnaires were completed by multiple respondents from forty-three of Thai and foreign pharmaceutical companies to ensure the data covered a range of the whole industry. This paper attempted to develop an empirically supported partnership model and strategic fit model in outsourcing pharmaceutical manufacturing industry. Likert-type questionnaire items adapted from previously validated studies were completed by senior managers from each company to assess the outsourcing relations between CPs and CMs via the PNS model, STF model and the perceived outcome performance; such as: company revenue, and % growth rate; were considered. The details of the research methods are as follows:

In this section, we present two properties that characterize optimal solutions for the special cases of problem [P]. Considering these properties, we suggest two construction heuristic algorithms to obtain optimal solutions. To illustrate construction heuristics, all storage locations are supposed to be numbered in the topological order from top to bottom row and from left to right column in storage locations, starting with the number one, as in Figure 2.

Consider the special case [SP1] in which all blocks are scheduled to be stored in the same period. Let Φ be the set of all blocks ranked in descending order of retrieval.

Proposition 1. There is an optimal solution of [SP1] in which all blocks in Φ are sequentially stored in storage locations beginning with location 1.

Proof. Obviously ■

Consider another special case [SP2] in which all blocks are scheduled to be retrieved in the same period. Let Ω be the set of all blocks ranked in ascending order of storage period.

Proposition 2. There is an optimal solution of [SP2] in which all blocks in Ω are sequentially stored in storage locations beginning with location 1.

Proof. Obviously ■

3.1 Construction heuristic 1

This heuristic is based on the idea in Proposition 1. That is, processing from period 1 to period $|T|$, all blocks stored in the period under consideration are ranked in descending order of retrieval period. In addition, they are sequentially stored in empty locations beginning with location 1 until all blocks stored in the period are stored. In the heuristic, for further reducing obstructive blocks, the blocks with different storage periods are stored in locations in different columns since blocks featuring the same storage period are sorted in descending order of retrieval period and storing the blocks with different storage periods in the locations in the same column may result in obstructive blocks.

In the algorithm, $L(j, t)$ represents the block stored in location j in period t. $F(k, t)$ represents the top kth block in the list ranked in descending order of retrieval period in storage period t.

Procedure of construction heuristic 1

Step 1. Initialize period $t = 1$, column $c = 1$, row $d = 1$, and $L(j, t) = 0$ for all j, t.

Step 2. For all stored blocks in period t, rank the blocks and list them in descending order of retrieval period.

Step 3. Process the following steps for the list beginning with the topmost block in the list $k = 1$, i.e., $i = F(1, t)$.

3.1. Find an empty location using the routine: if $d > |D|$, set $d = 1$ and $c = c + 1$; if $c > |C|$, set $c = 1$; set $j = d + (c - 1)\cdot|D|$; if $L(j, t) \neq 0$, set $d = d + 1$ and continue this routine until an empty location is found such at $L(j, t) = 0$.

3.2. Set $L(j, \mathrm{l}) = i$ for $\mathrm{l} = s_i, s_i + 1, s_i + 2, \ldots, r_i - 1$.

3.3. Set $k = k + 1$. If $k > |S_t|$, go to Step 4. Otherwise, $i = F(k, t)$ and go to Step 3.1.

Step 4. Set $t = t + 1$. If $t \leq |T|$, go to Step 2 and set $c = c + 1$ (for storing the blocks with different storage periods in the locations in a different column). Otherwise, terminate the procedure.

3.2 Construction heuristic 2

This heuristic is the same as construction heuristic 1, but this heuristic begins with the last period $|T|$, in following Proposition 2, and arranges the blocks in terms of storage period. That is, processing from period $|T|$ to period 1, all blocks retrieved in the period under consideration are ranked in ascending order of storage period and they are sequentially stored in empty locations beginning with location 1. In the procedure below, $B(k, t)$ represents the top kth block in the list ranked in ascending order of storage period in retrieval period t.

Procedure of construction heuristic 2

Step 1. Set $t = |T|$, $c = 1$, $d = 1$, $L(j, t) = 0$ for all j, t.
Step 2. For all retrieved blocks in period t, rank the blocks and list them in ascending order of storage period.
Step 3. Process the following steps for the list beginning with the topmost block in the list $k = 1$, i.e., $i = B(1, t)$.
3.1. Find an empty location using the routine: if $d > |D|$, set $d = 1$ and $c = c + 1$; if $c > |C|$, set $c = 1$; set $j = d + (c - 1)\cdot|D|$; and if $L(j, l) \neq 0$ for $l = s_i, s_i + 1, s_i + 2, \ldots, r_i - 1$, $d = d + 1$ and continue this routine until an empty location is found such at $L(j, l) = 0$ for $l = s_i, s_i + 1, s_i + 2, \ldots, r_i - 1$.
3.2. Set $L(j, l) = i$ for $l = s_i, s_i + 1, s_i + 2, \ldots, r_i - 1$.
3.3. Set $k = k + 1$. If $k > |R_t|$, go to Step 4. Otherwise, set $i = B(k, t)$ and go to Step 3.1.
Step 4. Set $t = t - 1$. If $t \geq 1$, go to Step 2 and set $c = c + 1$ (for storing the blocks with different retrieval periods in the locations in a different column). Otherwise, terminate the procedure.

4. Tabu search algorithm

The solutions obtained by the construction heuristics are improved by a Tabu search (TS) algorithm. The TS algorithm, an enhanced version of the local search algorithm suggested in Glover (1989, 1990a, 1990b), is one of the most commonly used meta-heuristics to solve various types of combinatorial optimization problems.

Our TS begins with an initial solution obtained by one out of the two construction heuristics and obtains a neighbourhood solution for the current solution, called a move. A neighbourhood solution is obtained using a pairwise interchange of blocks stored in two different storage locations. To generate a feasible neighbourhood solution, our TS selects a period randomly and two locations neither occupied by a block nor occupied by a block to be scheduled in the selected period, and interchanges all blocks in the locations from the period to the end of the planning period.

In order to accentuate the search space, intensification, all possible neighbourhoods are generated by the feasible pairwise interchange method and then the incumbent solution is replaced by the elite neighbourhood solution that gives the most rapid improvement rate in the objective function value. For the intensification length, the number of feasible neighbourhood solutions is not more than $|J|\cdot(|J| - 1)/2$; i.e., our TS generates all neighbourhood combining locations j and k for $j < k \in J$. In order to broaden the search space, diversification, our TS selects a less frequently selected neighbour among neighbourhood solutions with the same objective function value.

To prevent cycling back to a solution, our TS records certain moves called attributes and temporarily prevents a move that has been visited in the recent past called the tabu move. These previously visited solutions are excluded by recording the memories called the tabu list. The tabu moves are invoked by the pre-defined number of iterations called the tabu tenure. In our implementation, the tabu size is set to 7. If the tabu list is full, the oldest attribute in the tabu list is eliminated in order to allow more flexibility in future moves. Since tabu moves are sometimes restricted, the TS may prevent access to good neighbourhood solutions even when there is no possibility of cycling due to a revisit of previous solutions. To avoid missing good solutions, our TS overrides the tabu status of a tabu move if this move leads to a more optimal solution than identified in the search, called the aspiration criterion. The most commonly used stopping conditions are a fixed number of iterations, a fixed amount of CPU time, and the number of iterations without an improvement in the objective function value. Our TS is terminated if the number of iterations without an improvement reaches 50.

5. Computational experiments

To illustrate the performance of the heuristic algorithms suggested in this paper, computational experiments were performed on randomly generated test instances and results are reported in this section. All heuristics were coded in the computer programming language C. The experiments were performed on a personal computer with a Pentium processor operating at 2.67 GHz with 1.99 GB ram. CPLEX 11.2, a commercial optimization software package, was used to solve the integer programming model. To avoid an excessive run of the CPLEX, we limited the run time to 7200 seconds and obtained the best solution if the CPLEX could not obtain an optimal solution within this time limit.

Regarding the tests, 360 test instances were generated; i.e., ten instances for each combination of three levels of the workloads (70%, 80%, 90%), one level of the number of rows (5), four levels of the number of columns (5, 10, 15, 20), and three levels of the number of periods (10, 20, 30). The workload implies the percentage of the maximum storage capacity during the planning horizon, which is calculated by the number of storage locations times the number of periods. The number of blocks at a stockyard was generated such that the resulting workload becomes closer to the workload set in the problem setting. Each stored block was randomly scheduled to stay in the block stockyard between 1- 7 periods. Note that the sizes of the real assembly block stockyard in Park and Seo (2009b) are 5 rows × 10 columns as well as 5 rows × 15 columns.

Test results are summarized in Tables 1 to 4, which show the number of obstructive blocks and computation times. The number of obstructive blocks given in the tables is an average of the ten test instances. In the tables, CH1 and CH2 denote construction heuristic 1 and 2, respectively, and TS1 and TS2 denote the TS algorithms starting from the solutions of CH1 and CH2, respectively. Our heuristics were compared to the genetic algorithm (GA) of Park and Seo (2009b) after

correcting for the counting error. From the tables, the construction heuristics generated smaller obstructive blocks than the GA as the number of columns increases. For instance, in the case of the 5 × 5 stockyard problem with (T, W) = (10, 70%), the CH1 and CH2 generated 22.3 and 17.5 obstructive blocks on average respectively, and the GA generated 6.2 obstructive blocks on average. However, in the case of the 5 × 20 stockyard problem with the same number of periods and workload, the CH1 and CH2 generated 20.2 and 18.1 obstructive blocks on average respectively, and the GA generated 25.8 obstructive blocks on average. In terms of the computation time, the CH1 and CH2 took less than 0.005 seconds, which is significantly shorter than the time of the GA. On the other hand, the TS algorithm overall generated smaller obstructive blocks than the GA while their computation times are comparable, and therefore, we can conclude that our TS algorithm produces higher solution quality that the GA.

Table 1. Results of tests on the 5×5 stockyard problems

Parameters		GA		CH1		CH2		TS1		TS2		CPLEX		
T[a]	W[b]	OV[c]	CT[d]	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OPT[e]
	70%	6.2	1.63	22.3	0.00[f]	17.5	0.00	4.3	0.43	4.1	0.44	0.5	4136.97	(5,5,0)
10	80%	6.5	1.74	25.1	0.00	24.5	0.00	4.8	0.40	6.4	0.42	0.9	5413.87	(3,7,0)
	90%	10.9	1.93	30.6	0.00	30.9	0.00	8.8	0.35	10.5	0.34	2.9	7200.00	(0,10,0)
	70%	24.3	3.40	61.3	0.00	60.1	0.00	19.1	1.14	25.0	1.12	7.4	7200.00	(0,10,0)
20	80%	33.0	3.72	74.7	0.00	69.6	0.00	26.4	1.02	31.3	1.09	13.5	7200.00	(0,10,0)
	90%	43.2	4.10	84.3	0.00	90.4	0.00	35.1	0.73	41.1	0.83	24.4	7200.00	(0,10,0)
	70%	52.7	5.34	99.5	0.00	97.9	0.00	44.1	2.09	58.2	1.29	11.8	7196.84	(1,9,0)
30	80%	67.7	5.86	122.9	0.00	124.1	0.00	60.2	1.61	63.5	1.68	20.9	7200.00	(0,10,0)
	90%	88.8	6.62	140.9	0.00	146.2	0.00	76.3	0.90	84.2	1.15	48.5	7200.00	(0,10,0)

[a] number of periods
[b] workload
[c] number of obstructive blocks (the objective function value)
[d] computation time in seconds
[e] number of optimal solutions, number of feasible solutions, and number of undefined solutions out of the different ten problems
[f] computation time is less than 0.005 seconds

Table 2. Results of tests on the 5×10 stockyard problems

Parameters		GA		CH1		CH2		TS1		TS2		CPLEX		
T	W	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OPT
	70%	8.0	5.16	18.9	0.00	19.1	0.00	2.9	2.68	5.9	4.02	0.5	2793.84	(7,3,0)
10	80%	14.3	5.89	28.2	0.00	29.2	0.00	4.5	2.97	10.1	3.44	1.5	5040.04	(4,6,0)
	90%	19.5	6.52	38.6	0.00	38.5	0.00	7.9	3.15	12.0	4.21	2.0	6462.04	(3,7,0)
	70%	54.0	11.43	68.2	0.00	72.8	0.00	20.7	10.74	26.2	11.31	45.7	7200.00	(0,10,0)
20	80%	74.2	13.31	103.1	0.00	102.9	0.00	34.9	9.66	43.9	10.54	193.6	7200.00	(0,5,5)
	90%	89.9	15.47	122.5	0.00	118.6	0.00	49.1	9.79	62.0	10.16	-	7200.00	(0,0,10)
	70%	119.8	17.82	121.9	0.00	120.7	0.00	52.0	20.22	59.8	21.30	220.1	7200.00	(0,7,3)
30	80%	157.1	22.23	176.3	0.00	176.8	0.00	77.5	20.18	91.8	19.08	294.3	7200.00	(0,4,6)
	90%	192.5	26.03	211.3	0.00	224.3	0.00	100.6	18.65	119.3	17.33	-	7200.00	(0,0,10)

Table 3. Results of tests on the 5×15 stockyard problems

Parameters		GA		CH1		CH2		TS1		TS2		CPLEX		
T	W	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OPT
	70%	15.9	11.07	19.1	0.00	17.1	0.00	1.0	10.24	5.2	10.48	1.5	5252.06	(5,5,0)
10	80%	25.3	13.05	30.1	0.00	28.3	0.00	3.4	12.06	7.7	15.72	4.7	6712.38	(2,8,0)
	90%	31.0	14.73	49.2	0.00	51.1	0.00	8.7	19.22	16.0	19.27	3.4	6676.65	(2,8,0)
	70%	98.9	25.83	74.5	0.00	78.5	0.00	23.4	41.54	29.2	58.22	237.3	7200.00	(0,7,3)
20	80%	132.0	31.73	115.7	0.00	108.3	0.00	38.2	48.41	45.4	47.77	276.0	7200.00	(0,6,4)
	90%	156.6	37.00	164.9	0.00	153.1	0.00	58.9	38.03	72.3	48.63	-	7200.00	(0,0,10)
	70%	202.2	41.97	136.9	0.00	142.5	0.00	59.0	86.19	69.4	73.92	384.4	7200.00	(0,8,2)
30	80%	256.5	54.27	204.1	0.00	199.3	0.00	84.5	97.95	101.3	93.11	477.0	7200.00	(0,4,6)
	90%	305.1	71.56	291.7	0.00	289.6	0.00	129.9	76.21	146.0	81.50	-	7200.00	(0,0,10)

Table 4. Results of tests on the 5×20 stockyard problems

Parameters		GA		CH1		CH2		TS1		TS2		CPLEX		
T	*W*	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OV	CT	OPT
10	70%	25.8	19.88	20.2	0.00	18.1	0.00	1.0	24.93	4.4	32.98	45.7	7200.00	(0,10,0)
	80%	43.2	23.69	33.4	0.00	35.5	0.00	2.8	33.19	6.3	47.82	145.0	7200.00	(0,8,2)
	90%	54.2	26.68	49.8	0.00	54.1	0.00	10.2	38.47	17.6	57.38	196.0	7200.00	(0,1,9)
20	70%	149.9	47.51	69.4	0.00	71.1	0.00	20.2	124.86	32.1	101.97	309.3	7200.00	(0,7,3)
	80%	199.0	62.42	130.6	0.00	134.7	0.00	37.2	136.81	49.4	131.37	402.0	7200.00	(0,8,2)
	90%	230.9	75.50	168.6	0.00	179.0	0.00	58.4	123.24	78.5	138.71	-	7200.00	(0,0,10)
30	70%	302.0	79.90	119.4	0.00	135.9	0.00	55.5	185.64	82.7	182.04	507.9	7200.00	(0,9,1)
	80%	377.9	107.16	223.1	0.00	217.3	0.00	98.6	227.15	111.8	227.93	624.4	7200.00	(0,7,3)
	90%	443.2	147.47	307.7	0.00	277.6	0.00	133.4	206.98	146.4	251.11	763.0	7200.00	(0,2,8)

When compared with the CPLEX taking more than two hours in most cases, the TS algorithm took less than 300 seconds and the obstructive blocks of the TS algorithm are less than those of the CPLEX in many cases. The computation time of the TS algorithm is sensitive to the number of periods; i.e., it increases as the number of periods increases. On the other hand, the computation time does not increase as the workload increases in some cases. This may result from the workload is very high, i.e., 90% in the test, and there may be little room that the solution can be improved. Hence, the algorithms might terminate quickly due to no improvement for many iterations.

6. Conclusion

We considered the planar storage location assignment problem, which is defined as that of assigning assembly blocks to storage locations at a stockyard over a planning horizon, with the objective of minimizing the total number of obstructive blocks. We identified a counting error of the PS model, suggested a new integer program, and developed two construction algorithms and a TS algorithm to solve this problem. To show the performance of the suggested heuristic algorithms, we performed various computational experiments on various randomly generated test instances and compared heuristic algorithms with the existing GA. Test results showed that our TS algorithm guaranteed higher solution quality than the GA.

This research assumed that the obstructive blocks should be returned to their original locations after finishing a block's move as in Park and Seo (2009b). Relaxation of this assumption may generate smaller numbers of obstructive blocks. Therefore, development of the model and algorithm for the problem without this assumption is an important focus for future research. Also, the model and algorithm for the problem with four directional storage and retrieval operations are worthwhile being developed.

Acknowledgement

This work was supported by Jungseok Research Institute of International Logistics and Trade Research Grant.

References

Glover, F., 1989. Tabu search - Part I. ORSA Journal on Computing 1, 190-206.

Glover, F., 1990a. Tabu search -Part II. ORSA Journal on Computing 2, 4-32.

Glover, F., 1990b. Tabu search: A tutorial. Interfaces 20, 74-94.

Park, C., Seo, J., Kim, J., Lee, S., Baek, T.H., Min, S.K., 2007. Assembly block storage location assignment at a shipyard: A case of Hyundai Heavy Industries. Production Planning and Control 18, 180–189.

Park, C., Seo, J., 2009a. Assembly block storage location assignment problem: Revisited. Production Planning and Control 20, 216–226.

Park, C., Seo, J., 2009b. Mathematical modeling and solving procedure of the planar storage location assignment problem. Computers and Industrial Engineering 57, 1062–1071.

Park, C., Seo, J., 2010. Comparing heuristic algorithm of the planar storage location assignment problem. Transportation Research Part E: Logistics and Transportation Review 46, 171–185.