ICIC Express Letters
Part B: Applications
Volume **9**, Number **12**, December **2018**

ICIC International ©2018 ISSN 2185-2766
pp. 1287–1294

REAL-TIME HOIST SCHEDULING WITH A REDUCED BRANCH-AND-BOUND ALGORITHM

Byung Soo Kim[1], Rasaratnam Logendran[2], Jungkoo Lee[3] and Shiegheun Koh[4,*]

[1]Department of Industrial and Management Engineering
Incheon National University
119, Academy-ro, Yeonsu-gu, Incheon 22012, Korea

[2]School of Mechanical, Industrial & Manufacturing Engineering
Oregon State University
1500, SW Jefferson Street, Corvallis, Oregon 97331, USA

[3]Korean Air Co., Ltd.
55, Tech-center-ro, Gangseogu, Busan 46712, Korea

[4]Department of Systems Management and Engineering
Pukyong National University
Yongsoro 45, Namgu, Busan 608-737, Korea
*Corresponding author: sgkoh@pknu.ac.kr

Received May 2018; accepted August 2018

Abstract. *This paper deals with a single-hoist and multiple-products scheduling problem. Although some MILP (mixed integer linear programming) models for the problem were developed earlier, the situation of the company that motivated this study is different from the earlier studies, and thus a modified MILP model is proposed. Applying the model to a commercial optimization software, optimal solutions for small problems can be found in a short time. In the case of realistic industry-size problems, however, the model cannot find an optimal solution in a reasonable computation time, and therefore, a branch-and-bound based heuristic algorithm is developed to solve such problems. To assess the performance of the heuristic, we generate larger example problems and compare the results of the algorithm with the optimal solutions. The comparison shows that the heuristic has very good performance and the computation time is sufficiently short to use the algorithm in real situation.*

Keywords: Hoist scheduling, Branch-and-bound, Heuristics, Integer programming

1. Introduction.

A class of multistage production lines employs a computer-controlled hoist system to perform the transportation of parts between stages. This study was motivated by a company manufacturing aircraft parts, in which there are a number of chemical tank lines for electroplating the parts. Each production line consists of a series of tanks filled with a variety of chemical solutions for surface processing of the metal parts, and the processing of parts is compiled by soaking them into tanks for a certain time and in particular order.

Since Phillips and Unger [4] introduced the problem, various hoist scheduling problems have been extensively studied, and in most of them the jobs being processed in the line when a new schedule is made are not considered. Because various orders of multiple items randomly arrive at the system, however, the dynamic feature should be considered in the hoist scheduling problem. For the dynamic problems, Yin and Yih [7] proposed a one-pass heuristic, and later, Yih [8] developed a two-phase heuristic. Recently, Yan et

DOI: 10.24507/icicelb.09.12.1287

al. [6] proposed a two-phase branch-and-bound algorithm for the dynamic hoist scheduling problem considering disturbance.

However, the above studies related to dynamic scheduling considered only a single new job when the rescheduling is made. Zhao et al. [9] proposed an MILP model for the general multiple jobs rescheduling problem in which various jobs randomly (or dynamically) arrive at the system and should be rescheduled with the jobs-in-process. Later, Tian et al. [5] Feng et al. [2], and Amraoui and Elhafsi [1] modified the model of Zhao el al. [9].

The description above outlines the general situation of the single-hoist and multiple-products scheduling problems. However, the production line that motivated this study has an interesting feature that was not considered in the earlier studies. The company produces parts for assembling an aircraft and an order from the customer consists of a set of parts. In order that the set is transported by the hoist and soaked in a tank all together, a tool for installing parts, which is called a *rack* in the company, is needed. In order to insert a new job from the input buffer into the tank line, an empty rack is needed. After a job arrives at the output buffer, the parts that were installed on the rack are released. In this paper a new MILP model that considers this rack condition is proposed, but since this model cannot find an optimal solution for the real production line with 30 tanks in the company, a branch-and-bound based heuristic algorithm is developed to solve the large-size problems in real situation.

In the remainder of this paper, we first present a formal description of the considered problem and a mixed integer linear programming (MILP) model in Section 2. Following the description of the proposed heuristic algorithm in Section 3, the results of the computational experiments are presented in Section 4. And finally, we conclude this study in Section 5.

2. Model Formulation.

The features of the system including the rack constraint can be described by some assumptions stated below.

1) The number of jobs to be processed (n) is finite and known. A part of the jobs were already released into the production line and are staying in some tanks, and the other jobs are waiting for processing in the input buffer.

2) The number of tanks (m) is finite. We assume the first tank (tank 1) is the input buffer, the second one (tank 2) is the space where the jobs are installed on the rack, tanks 3 through $m-1$ are real tanks for the immersion processes, and the last tank (tank m) is the output buffer where the job is released from the rack and the rack is waiting for a new job. The rack is transported to tank 2 when a job in the input buffer (tank 1) is ready to proceed into the line. After an empty rack arrives at tank 2, the parts of a new job in the input buffer are installed on the rack.

3) The immersion sequences for jobs are known and fixed. Once a job begins its immersion process in a tank, it cannot be interrupted.

4) The processing time for a job in a tank must be within its prescribed time window that depends on the tank and the job.

5) Material flow in this system is performed by a single hoist that can handle only one job at a time. Once the loaded or empty hoist begins a travel, it cannot be interrupted.

The notations are as follows:

q_i: current position of job i at time 0 ($q_i = 1, 2, \ldots, m$)

e_i: elapsed time of job i at current position at time 0

h: current position of the hoist at time 0 ($h = 1, 2, \ldots, m$)

c_r: capacity of tank r

R: total number of racks in the system

$\alpha_{i,r}$: tank that job i has to visit after tank r

$\underline{p}_{i,r}$: lower limit of the processing time of job i in tank r

$\bar{p}_{i,r}$: upper limit of the processing time of job i in tank r

$\theta_{r,s}$: loaded hoist travel time from tank r to tank s
$\delta_{r,s}$: empty hoist travel time from tank r to tank s
M: sufficiently large positive number
T: makespan of the system
$t_{i,r}$: time when job i leaves tank r
b_i: time when a rack is assigned to job i
$w_{i,r,j,s}$: 1, if the hoist picks up job j from tank s after it picks up job i from tank r, or 0, otherwise
$u_{i,j,r}$: 1, if job i arrives at tank r before job j leaves tank r, or 0, otherwise
$v_{i,j,r}$: 1, if job i arrives at tank r after job j arrives at tank r, or 0, otherwise
$x_{i,j,r}$: 1, if job i arrives at tank r when job j stays at tank r, or 0, otherwise
$\lambda_{i,j}$: 1, if a rack is assigned to job i before job j is released from another rack, or 0, otherwise
$\mu_{i,j}$: 1, if a rack is assigned to job i after another rack is assigned to job j, or 0, otherwise
$z_{i,j}$: 1, if a rack is assigned to job i when job j occupies another rack, or 0, otherwise
Using the above notations, we can represent our problem as an MILP model as follows:

$$\text{Minimize} \quad T \tag{1}$$

subject to

$$\underline{p}_{i,q_i} \leq t_{i,q_i} + e_i \leq \bar{p}_{i,q_i} \text{ for all } i \text{ with } q_i > 1 \tag{2}$$

$$\underline{p}_{i,r} \leq t_{i,r} - t_{i,s} - \theta_{s,r} \leq \bar{p}_{i,r} \text{ for all } i, r, s \text{ with } \alpha_{i,s} = r \tag{3}$$

$$t_{i,q_i} \geq \delta_{h,q_i} \text{ for all } i \text{ with } q_i < m \tag{4}$$

$$t_{i,m} = \underline{p}_{i,m} - e_i \text{ for all } i \text{ with } q_i = m \tag{5}$$

$$t_{i,m} \leq T \text{ for all } i \tag{6}$$

$$t_{j,s} - t_{i,r} \geq \theta_{r,\alpha_{i,r}} + \delta_{\alpha_{i,r},s} - M(1 - w_{i,r,j,s}) \quad \text{for all } i, r, j, s \text{ with } \alpha_{i,r} \neq 0,\ \alpha_{j,s} \neq 0 \text{ and } (i,r) \neq (j,s) \tag{7}$$

$$w_{i,r,j,s} + w_{j,s,i,r} = 1 \text{ for all } i, r, j, s \text{ with } \alpha_{i,r} \neq 0,\ \alpha_{j,s} \neq 0 \text{ and } (i,r) \neq (j,s) \tag{8}$$

$$-Mu_{i,j,r} < t_{i,s} + \theta_{s,r} - t_{j,r} \leq M(1 - u_{i,j,r}) \quad \text{for all } i, r, j, s \text{ with } \alpha_{j,r} \neq 0,\ \alpha_{i,s} = r \text{ and } i \neq j \tag{9}$$

$$-Mv_{i,j,q_j} < -t_{i,s} - \theta_{s,q_j} \leq M(1 - v_{i,j,q_j}) \quad \text{for all } i, j, s \text{ with } \alpha_{i,s} = q_j \text{ and } i \neq j \tag{10}$$

$$-Mv_{i,j,r} < t_{j,g} + \theta_{g,r} - t_{i,s} - \theta_{s,r} \leq M(1 - v_{i,j,r}) \quad \text{for all } i, r, j, s, g \text{ with } q_j \neq r,\ \alpha_{j,r} \neq 0,\ \alpha_{j,g} = r,\ \alpha_{i,s} = r \text{ and } i \neq j \tag{11}$$

$$2x_{i,j,r} \leq u_{i,j,r} + v_{i,j,r} \leq 1 + x_{i,j,r} \quad \text{for all } i, j, r \text{ with } \alpha_{i,r} \neq 0,\ \alpha_{j,r} \neq 0 \text{ and } i \neq j \tag{12}$$

$$\sum_{j \neq i, \alpha_{j,r} \neq 0} x_{i,j,r} \leq c_r - 1 \text{ for all } i, r \text{ with } \alpha_{i,r} \neq 0 \text{ and } 1 < r < m \tag{13}$$

$$b_i = 0 \text{ for all } i \text{ with } \alpha_{i,1} = 0 \tag{14}$$

$$b_i = t_{i,1} \text{ for all } i \text{ with } \alpha_{i,1} > 0 \tag{15}$$

$$-M\lambda_{i,j} < b_i - t_{j,m} \leq M(1 - \lambda_{i,j}) \text{ for all } i \neq j \tag{16}$$

$$-M\mu_{i,j} < b_j - b_i \leq M(1 - \mu_{i,j}) \text{ for all } i \neq j \tag{17}$$

$$2z_{i,j} \leq \lambda_{i,j} + \mu_{i,j} \leq 1 + z_{i,j} \text{ for all } i \neq j \tag{18}$$

$$\sum_{j \neq i} z_{i,j} \leq R - 1 \text{ for all } i \tag{19}$$

$$w_{i,r,j,s},\ u_{i,j,r},\ v_{i,j,r},\ x_{i,j,r},\ \lambda_{i,j},\ \mu_{i,j},\ z_{i,j} : 0/1 \text{ integers}$$

In this model Equation (1) shows the purpose of this problem is minimizing makespan. Equations (2) and (3) show the bounds for processing times of the jobs. Equation (4) assures that the time when job i leaves current tank cannot be earlier than empty hoist travel time from current position to the tank. If a job stays in the output buffer at time zero, the finishing time of the job can be calculated by Equation (5). Equation (6) shows the relationship between the finishing times of all jobs and the makespan. Equations (7) and (8) ensure that an adequate travel time of the hoist should be secured between two loaded hoist movements. Equations (9) through (12) state that the variable $x_{i,j,r}$ has the value one if and only if job i arrives at tank r when job j stays at tank r. Equation (13) is the capacity constraint for each tank.

Equations (14) through (19) represent the rack constraint. From Equation (14), the time when a rack is assigned to job i is regarded as zero if the job is in process at time 0; on the other hand, Equation (15) says the rack assigning time is $t_{i,1}$ if the job is in the input buffer at time 0. Similar to Equations (9) through (12), Equations (16) through (18) ensure that the variable $z_{i,j}$ has the value one if and only if a rack is assigned to job i when job j occupies another rack. Equation (19) states that total number of racks assigned to jobs when a rack is assigned to job i is less than R.

Now, a simple example problem is introduced to validate the model. The data are as follows: $m = 8$, $n = 5$, $R = 3$, $h = 4$, $q_i = (8, 6, 4, 1, 1)$, $e_i = (15, 5, 1, 12, 0)$,

$$c_r = (\infty, 2, 1, 1, 2, 1, 1, \infty), \quad \alpha_{i,r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 5 & 6 & 7 & 8 & 0 \\ 2 & 4 & 0 & 5 & 6 & 7 & 8 & 0 \\ 2 & 3 & 4 & 5 & 8 & 0 & 0 & 0 \end{bmatrix},$$

$$\underline{p}_{i,r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 & 0 & 15 & 10 & 30 \\ 0 & 0 & 0 & 10 & 20 & 15 & 10 & 30 \\ 0 & 30 & 0 & 10 & 20 & 15 & 10 & 30 \\ 0 & 30 & 10 & 10 & 20 & 0 & 0 & 30 \end{bmatrix},$$

$$\bar{p}_{i,r} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 & 0 & 30 & 30 & 30 \\ 0 & 0 & 0 & 25 & 40 & 30 & 30 & 30 \\ \infty & \infty & 0 & 25 & 40 & 30 & 30 & 30 \\ \infty & \infty & 25 & 25 & 40 & 0 & 0 & 30 \end{bmatrix},$$

$$\delta_{r,s} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 6 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 1 & 0 & 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 1 & 0 & 1 & 2 & 3 & 4 \\ 3 & 3 & 2 & 1 & 0 & 1 & 2 & 3 \\ 2 & 4 & 3 & 2 & 1 & 0 & 1 & 2 \\ 1 & 5 & 4 & 3 & 2 & 1 & 0 & 1 \\ 0 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}, \quad \theta_{r,s} = \begin{bmatrix} 0 & 11 & 7 & 8 & 9 & 10 & 11 & 12 \\ 6 & 0 & 6 & 7 & 8 & 9 & 10 & 11 \\ 7 & 6 & 0 & 6 & 7 & 8 & 9 & 10 \\ 8 & 7 & 6 & 0 & 6 & 7 & 8 & 9 \\ 9 & 8 & 7 & 6 & 0 & 6 & 7 & 8 \\ 10 & 9 & 8 & 7 & 6 & 0 & 6 & 7 \\ 11 & 10 & 9 & 8 & 7 & 6 & 0 & 6 \\ 12 & 11 & 10 & 9 & 8 & 7 & 6 & 0 \end{bmatrix}.$$

Applying the model and the example data to the optimization software LINGO, an optimal solution was found in 2.6 GHz PC in a very short computation time (less than 0.01 seconds). Figure 1 shows the result, in which the optimal job sequence that the hoist has to serve is $(2, 3, 2, 4, 3, 5, 4, 3, 4, 5, 3, 5, 4, 5, 4, 5, 4)$. The numbers represented for tanks 1 through 7 in the figure are the starting times of the hoist tasks, while the numbers for tank 8 are finishing times of the jobs. The figure also shows the makespan of the system is 212.

This being a small example problem, it can be solved optimally by LINGO in a very short time. However, as shown by Lei and Wang [3], even the single item hoist scheduling

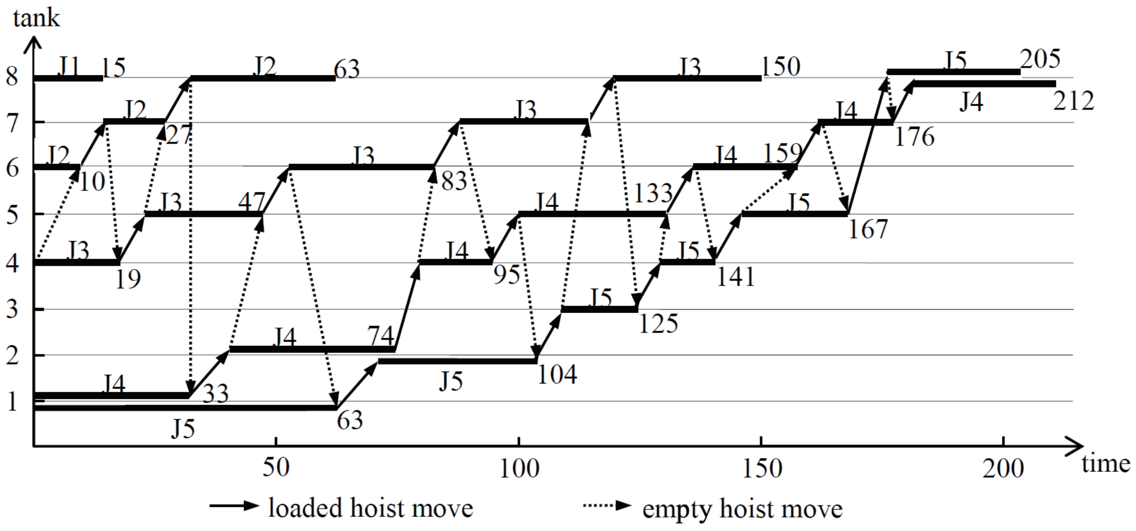

FIGURE 1. Optimal schedule of the example problem

problem is NP-hard. Therefore, an IP solver such as CPLEX or LINGO cannot provide an optimal solution for an industry-size problem in a reasonable CPU time. Therefore, we focus on the development of heuristic algorithms to obtain near-optimal solutions for the problem in a reasonable CPU time.

3. Heuristic Procedure.

The heuristic proposed in this paper is a simplified branch-and-bound algorithm, in which the number of branches is reduced as explained below.

3.1. Branching policy.

In the branch-and-bound algorithm for our problem, a node means a partial schedule. In the optimal solution in Figure 1, for example, the partial schedule after three loaded movements of the hoist is $(2, 3, 2)$. At a node in the branch-and-bound algorithm, n branches are needed in general. However, in this study some branches are discarded from further consideration in order to simplify the algorithm. The detailed procedure is as follows.

1) The jobs that have already arrived at the output buffer are excluded from the candidates for branching.

2) Considering the capacity of succeeding tank, the jobs that cannot go to the next tank are excluded from the candidates for branching.

3) If there is no available rack in the output buffer, the jobs in the input buffer are excluded from the candidates for branching.

4) If the hoist has to wait for a long time to lift a job when the job is selected for the next job to be transported, it would be better to eliminate the job from the candidates for branching. The hoist waiting time under the assumption that each job is selected for the next task of the hoist has to be calculated. Once the hoist waiting time is calculated, all jobs with zero waiting time are selected for branching. If the number of jobs with zero waiting time is greater than or equal to two, no more jobs are considered for branching. In case of zero or one job with zero waiting time, however, two branches are generated for two jobs with zero waiting time and the shortest positive waiting times.

5) Even though the number of branches is restricted to just two, the computation time of the algorithm becomes very long when the problem size is big. Therefore, the number of branches must be reduced more. If the number of branch candidates with zero hoist waiting times is less than two, using *BRATIO* value, which is defined as a predetermined parameter from $(0, 1)$ interval, the number of branches is controlled. To do this, the ratio of the number of jobs in current partial schedule to total number of tasks that the hoist has to perform must be calculated. After calculation, if the resulting ratio is less than the

BRATIO value, the number of branches is restricted to two; otherwise, it is restricted to just one.

3.2. **Lower bound calculation.** After new nodes are generated by the branching policy stated above, to check the feasibility of the partial schedule for each new node, the elapsed times for all jobs in current tanks after the partial schedule is performed should be calculated and compared to the upper limit of the processing time of the job in the tank. After this step, a lower bound for each survived node should be calculated, and the value is $\max\{LB_1, LB_2, LB_3\}$, in which LB_1, LB_2, and LB_3 are calculated for each partial schedule as follows.

1) For each job that has not been completed yet, the earliest completion time of the job is calculated without consideration of the other jobs. Let LB_1 be the maximum value among them.

2) The earliest time when all of the remaining jobs can be processed in a tank is calculated for each tank that has a capacity of just one. Let LB_2 be the maximum value among them.

3) The minimum total loaded hoist travel time that is needed to perform all the remaining tasks is calculated. Additionally, the minimum processing time of the remaining jobs in the output buffer is found. And then, LB_3 is defined as the sum of those two time values and current time of the partial schedule.

3.3. **Algorithm.** Detailed procedure of the 'Reduced Branch-and-Bound Algorithm' is as follows.

```
Step 0 At the root node, the value of CurMin, which is defined as the current value of
       minimum makespan of the system, is set to $\infty$.
Step 1 First branch and bound.
       Step1.1 Make branches for all jobs that do not arrive at the output buffer.
       Step1.2 Eliminate the branches that violate the tank capacity constraint or the
               rack constraint.
       Step1.3 Calculate the elapsed times of jobs in each tank for the partial schedule
               of a branch. If an elapsed time exceeds its upper limit, eliminate the
               branch.
       Step1.4 Calculate the time needed to perform the partial schedule of the branch.
Step 2 If there is no node to branch, stop the algorithm with the current CurMin value.
       The case of CurMin $= \infty$ means the algorithm could not find a feasible solution.
Step 3 Branch and bound.
       Step 3.1 Choose the node having the maximum value of the time needed to per-
                form the partial schedule, and make branches from the node using Sub-
                section 3.1.
       Step 3.2 Check the elapsed times of the jobs in each branch and eliminate the
                branch if an elapsed time exceeds the upper limit.
       Step 3.3 Calculate the time needed to perform the partial schedule of the branch.
       Step 3.4 If the partial schedule of a branch is a complete schedule,
                Step 3.4.1 Calculate a completion time of the complete schedule.
                Step 3.4.2 If the completion time is less than the CurMin value, set the
                           CurMin value to be the completion time. Eliminate the branch
                           from further consideration.
       Step 3.5 If the partial schedule of a branch is not a complete schedule,
                Step 3.5.1 Calculate a lower bound for the branch using Subsection 3.2.
                Step 3.5.2 If the lower bound is greater than the CurMin value, eliminate
                           the branch from further consideration.
Step 4 Go to Step 2.
```

4. **Computational Results.** To evaluate the performance of the heuristic algorithm proposed in this paper, computational experiments have been conducted using larger test problems, in which the number of tanks including buffers is fixed at 12, and the number of jobs is allowed to vary from 6 to 8. Using these problems, the results of the heuristic algorithm are compared with the optimal solutions. To do this, an optimal solution for each problem is first found using LINGO software, and then the solutions of the same problem are found by the heuristic algorithm for 5 *BRATIO* values, i.e., 0, 0.25, 0.5, 0.75, and 1.

Table 1. Performance comparison of the heuristic with optimal solution

n			6	7	8
LINGO (Optimal)		Makespan	100	100	100
		Com. Time	2	5	31
Heuristic	$BRATIO = 0.0$	Makespan	103.3	104.5	110.2
		Com. Time	< 1	< 1	< 1
	$BRATIO = 0.25$	Makespan	101.7	102.3	107.5
		Com. Time	< 1	< 1	< 1
	$BRATIO = 0.5$	Makespan	100.8	101.9	102.8
		Com. Time	< 1	< 1	< 1
	$BRATIO = 0.75$	Makespan	100.0	100.6	101.4
		Com. Time	< 1	< 1	< 1
	$BRATIO = 1.0$	Makespan	100.0	100.2	100.8
		Com. Time	< 1	< 1	1

Table 1 shows the result and some sketchy characteristics of the heuristic may be obtained by reviewing the table as follows.

1) In terms of the solution quality, the makespan of the heuristic looks quite good. The table shows the worst result is 10.2% worse off than the optimal solution and the best result is the same as the optimal solution. Considering the computation time, the heuristic is much more practical than the MILP model.

2) If the n value increases, the solution quality of the heuristic becomes bad and the computation time for the heuristic becomes large.

3) For a problem, the bigger *BRATIO* value the heuristic uses, the better solution it can find and the longer computation time it takes. Therefore, if the computation time is too long to find a solution for a problem, a relatively small *BRATIO* value can be adopted to find a solution of the problem in a reasonable computation time.

5. **Conclusions.** In this research we investigated a dynamic single-hoist and multiple-products scheduling problem with the rack constraint. First, an MILP model was developed, which is an extension of the models proposed in earlier studies. And then, a branch-and-bound based heuristic algorithm was proposed to solve realistic industry-size problems. To assess the performance of the heuristic, we generated large example problems and compared the results of the algorithm with the optimal solutions derived from the MILP model. From the computational experiment we did find very good properties of the heuristic algorithm, i.e., the makespan identified is of good quality and the computation time is much shorter than the optimal solution, and thus, we conclude that the heuristic can be used to solve real industry-size problems.

Several directions for future research are apparent from this study. First, we are currently investigating the case of two hoist system. Second, the possibility of using an intermediate storage buffer in the tank line may also be examined.

Acknowledgement. This work was supported by the Pukyong National University Research Abroad Fund in 2015 (C-D-2015-0515).

REFERENCES

[1] A. E. Amraoui and M. Elhafsi, An efficient new heuristic for the hoist scheduling problem, *Computers & Operations Research*, vol.67, pp.184-192, 2016.

[2] J. Feng, A. Che and C. Chu, Dynamic hoist scheduling problem with multi-capacity reentrant machines: A mixed integer programming approach, *Computers & Industrial Engineering*, vol.87, pp.611-620, 2015.

[3] L. Lei and T. J. Wang, *A Proof: The Cyclic HSP Is NP-Complete*, Technical Report 89-0016, Graduate School of Management, Rutgers University, 1989.

[4] L. W. Phillips and P. S. Unger, Mathematical programming solution of a hoist scheduling program, *AIIE Transactions*, vol.28, no.2, pp.219-225, 1976.

[5] N. Tian, A. Che and J. Feng, AIChE Letter: Real-time hoist scheduling for multistage material handling process under uncertainties, *AIChE Journal*, vol.59, no.4, pp.1046-1048, 2013.

[6] P. Yan, A. Che, X. Cai and A. Tang, Two-phase branch and bound algorithm for robotic cells rescheduling considering limited disturbance, *Computers & Operations Research*, vol.50, pp.128-140, 2014.

[7] N. C. Yin and Y. Yih, Crane scheduling in a flexible electroplating line: A tolerance-based approach, *Journal of Electronics Manufacturing*, vol.2, no.4, pp.137-144, 1992.

[8] Y. Yih, An algorithm for hoist scheduling problems, *International Journal of Production Research*, vol.32, no.3, pp.501-516, 1994.

[9] C. Zhao, J. Fu and Q. Xu, Real-time dynamic hoist scheduling for multistage material handling process under uncertainties, *AIChE Journal*, vol.59, no.2, pp.465-482, 2013.