Jurnal Teknik Industri, Vol. 14, No. 2, Desember 2012, 83-88
ISSN 1411-2485 print / ISSN 2087-7439 online

Harmony Search for Balancing Two-sided Assembly Lines

Hindriyanto Dwi Purnomo[1], Hui-Ming Wee[2], Hsin Rau[2]

Abstract: Two-sided assembly lines balancing problems are important problem for large-sized products such as cars and buses, in which, tasks operations can be performed in the two sides of the line. In this paper, harmony search algorithm is proposed to solve two-sided assembly lines balancing problems type-I (TALBP-I). The proposed method adopts the COMSOAL heuristic and specific features of TALBP in the harmony operators – the harmony memory consideration, random selection and pitch adjustment – in order to maintain the local and global search. The proposed method is evaluated based on 6 benchmark problems that are commonly used in TALBP. The experiment results show that the proposed method work well and produces better solution than the heuristic method and genetic algorithm.

Keywords: Harmony search, two-sided assembly lines, balancing.

Introduction

An assembly line is a sequential of workstations that are connected by material handling system and is commonly used in producing high quality products Askin and Standridge [2], Scholl *et al.* [17], Boysen *et al.* [4, 5]. The desired performance of an assembly line can be controlled by assigning tasks to workstations in such a way the assembly objective is fulfilled, the demand is met and the constraints imposed on the line are satisfied, Simaria and Vilarinho [18]. The most common objective for analytical assembly line balancing is minimizing idle time. Assembly line can be one side or two side lines in parallel. One side line is the most widely studied of line balancing problems, Lee *et al.* [13], however two sided line is very important for large-sized products such as buses and truck, Kim *et al.* [11,12]. In two side assembly line, workers perform tasks on both side of the line and typically there is a pair of workstations (one on each side) at each station, Simaria and Vilarinho [18].

The literature on the two-sided assembly lines balancing problems (TALBP) is limited, Simaria and Vilarinho [18]. The first literature on TALBP is written by Bartholdi [2], who conducted an interactive program embodied with balancing algorithm based on the first fit heuristic. According to Gutjahr and Nemhauser [10], assembly lines balancing problems is belong to the NP-hard combinatorial optimization problems. For this reason, many published papers implement heuristics and metaheuristics methods to solve the TALBP.

[1] Department of Information Technology, Satya Wacana Christian University, Jl. Diponegoro 52-60, Salatiga 50711, Indonesia Email: hindriyanto.purnomo@staff.uksw.edu.
[2] Department of Industrial and System Engineering, Chung Yuan Christian University, Chung Pei Rd. 200, Chung Li, 32023, Taiwan ROC. Email: weehm@cycu.edu.tw, hsinrau@cycu.edu.tw.

* Corresponding author

Kim *et al.* [11, 12] used genetic algorithm to solve two assembly lines balancing. Lee *et al.* [13] propose a group assignment procedure focusing on the maximization of work relatedness and work slackness. Simaria and Vilarinho [18] implemented ant colony optimization for mixed two-sided assembly line balancing. Özcan [15] used simulated annealing for stochastic two-sided assembly line balancing. Ozbakır and Tapkan [14] applied Bee Colony Optimization for zone-constrained TALBP. Purnomo *et al.* [16] compared the genetic algorithm and first-fit rules on TALBP with assignment restrictions. Chutima and Chimklai [6] proposed particle swarm optimization for multi objective mixed model TALBP.

This paper proposes harmony search for minimizing the number of workstations for a given cycle time in two sided assembly lines balancing problem harmony search is an optimization method that uses musicians improvising as its analogy, Geem *et al.* [8, 9] and Geem [10]. Harmony Search has been successfully applied in various engineering applications such as: civil engineering, structural engineering, industrial engineering, traffic engineering, mechanical engineering, etc, Geem [10]. However, to the best of authors' knowledge, there is no published paper on the application of harmony search in TALBP. The rest of the paper is organized as follow: Section methods describes the two-sided assembly lines balancing problems, Section result and discussion explain the implementation of harmony search for two-sided assembly lines balancing and provides the numerical examples from benchmark problems are used to evaluate the performance of the implementation of harmony search in two-sided assembly lines balancing. Conclusion and future research are given in the last section.

Methods

Two –sided Assembly Lines Problems

According to Bartholdi [2], there are several advantages of two-sided line compare to single line, such as: reduce tasks time, reduce operators, reduce throughput time, reduce cost of tools and reduce material handling cost. In the proposed approach, the assembly line has left and right sides in which a pair of workstation is positioned at specified location in the line. The workstation pair performs different tasks simultaneously on the same individual product. According to Simaria and Vilarinho [18], the main difference between one-sided lines and two-sided lines is the sequence of the task performed. In one-sided lines, only the precedence constraint is important in sequencing task within a workstation. In two-sided assembly lines, the precedence constraint may result in interference. Interference increase idle time when a workstation needs to wait for a predecessor task to be completed at the opposite side of the line. The configuration of two sided assembly lines is shown in Figure 1.

Based on its objective, TALBP can be divided into TALBP-I and TALBP-II. The aim of TALBP-I is minimizing the number of workstation for a given cycle time while the objective of TALBP-II is minimizing the cycle time for a given number of workstations. In the TALBP-I, the production capacity is known by its cycle time and the remaining task is deciding the number of workstation. This problem is suitable for first time installation of assembly lines. On the other hand, in the TALBP-II, the number of workstation is known, and the remaining task is maximizing the production capacity by minimizing its cycle time. This problem is appropriate for the reconfiguration of assembly lines. In this paper, we will consider TALBP-I.

Notation use in this paper:

WS	:	number of workstations
n	:	number of task
t_i	:	processing time of task i
t_i^s	:	start time of task i
t_i^f	:	finish time of task i
C	:	cycle time
$\bar{d}$	:	average idle time
x_{ik}	:	a decision variable, x_{ik}= 1 if task i is assigned in workstation k, x_{ik} = 0 if otherwise
I	:	set of tasks, $I = \{1,2,\dots,i,\dots,n\}$
I_L	:	set of tasks that must be assigned in the left side; $I_L \in I$
I_R	:	set of tasks that must be assigned in the right side; $I_R \in I$

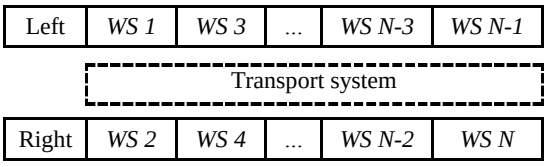


Figure 1. Two-sided assembly line

I_E	:	set of tasks that must be assigned in either side; $I_E \in I$
$P(j)$	:	set of immediate predecessor of task j
LB	:	lower bound
ZP	:	set of pair of tasks that must be assigned in the same workstation
ZN	:	set of pair of tasks that must not be assigned in the same workstation

The objective of the problem is to minimize the number of workstation for a given cycle time.

Minimize WS (1)

Subject to:

Each task is only assigned to one work station

$\sum_{k=1}^{WS} x_{ik} = 1, \ i \in I$ (2)

A task with specific side is assigned to the appropriate side

$\sum_{k=1}^{WS} \frac{2*x_{ik}}{mod(k,2)+1} = 1, \ i \in I_L$ (3)

$\sum_{k=1}^{WS} mod(k,2)x_{ik} + x_{ik} = 1, \ i \in I_R$ (4)

Precedence constraint

$\sum_{k=1}^{WS} \left(\left\lceil\frac{k}{2}\right\rceil - 1\right) x_{ik}C + t_i^f \leq \sum_{k=1}^{WS} \left(\left\lceil\frac{k}{2}\right\rceil - 1\right) x_{jk}C + t_j^s$ for all $i \in P(j)$ (5)

Synchronous tasks (6)

$\sum_{k=1}^{WS} \left(\left\lceil\frac{k}{2}\right\rceil - 1\right) x_{ik}C + t_i^s = \sum_{k=1}^{WS} \left(\left\lceil\frac{k}{2}\right\rceil - 1\right) x_{jk}C + t_j^s$

A pair of synchronous task must be started at the same time.

Tasks without precedence relationship

$\sum_{k=1}^{WS}(k-1)x_{ik}C + t_i^f \leq \sum_{k=1}^{WS}(k-1)x_{jk}C + t_j^s$ (7)

$\sum_{k=1}^{WS}(k-1)x_{jk}C + t_j^f \leq \sum_{k=1}^{WS}(k-1)x_{ik}C + t_i^s$ (8)

The constraints ensure that only one task is assigned in a workstation at a time.

The cycle time constraint

$x_{ik}.t_i^f \leq C$ (9)

Zoning constraint

a. Positive zoning

$\sum_{k=1}^{WS} \left\lceil\frac{k}{2}\right\rceil x_{ik} = \sum_{k=1}^{WS} \left\lceil\frac{k}{2}\right\rceil x_{jk}, \ (i,j) \in ZP$ (10)

b. Negative zoning

$\sum_{k=1}^{WS} \left\lceil\frac{k}{2}\right\rceil x_{ik} \neq \sum_{k=1}^{WS} \left\lceil\frac{k}{2}\right\rceil x_{jk}, \ (i,j) \in ZN$ (11)

The Harmony Search for TALBP

Harmony search consists of 5 steps: (1) Parameter initialization. (2) Harmony memory initialization. (3) New harmony improvisation. (4) Harmony memory (HM) update. (5) Termination criterion check.

In parameter initialization, the objective function and harmony search parameters are determined. In this study, the objective function is to minimize the number of workstation stated in equation 1. Parameters that need to be determined are the harmony memory size (HMS), harmony memory consideration rate (HMCR), pitch adjusting rate (PAR) and number of improvisation. HMS consists of a set of simultaneous solution vector and become a benchmark in every improvisation during the optimization process. HMCR and PAR are used to improve the solution vector while the number of improvisation is used to terminate the iteration.

In this paper, the number of workstation is used in the encoding scheme. Each harmony consists of n string in which n is the number of tasks. The value of each string is integer between 1 and the number of workstation *WS*. For example, assume a problem with 10 tasks as shown in Figure 2.

Suppose there are four workstations and the task are assigned according to Table 1. Then, the harmony representation of the example is (2 1 2 2 1 4 3 1 3 4). The encoding will also specify the side of the workstation. In this paper, odd numbers represent the left workstations while even number is for the right workstations. The tasks sequence is determined based on the mathematical model in this section. The harmony representation and the sequence of the tasks in the assembly line can be illustrated in Figure 3. Figure 3 shows that, there is interference before task 9 is started, because task 9 has two direct predecessor tasks, task 6 and task 7, that are not finish at the same time. The interference occurs as task 9 cannot be processed before task 6 and task 7 are finished. In addition, Figure 3 shows that the interference cause idle time in workstation 3.

After the encoding scheme is decided, the harmony search can be implemented. The harmony memory initialization is a process of randomly generating a set of solution vector.

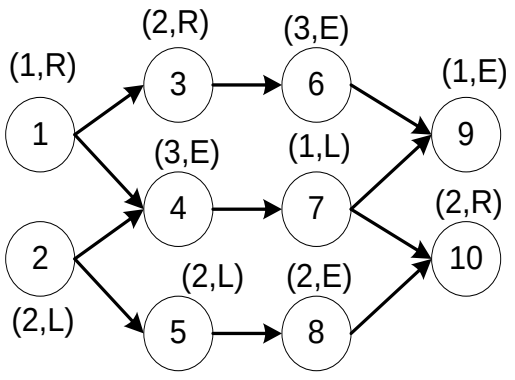
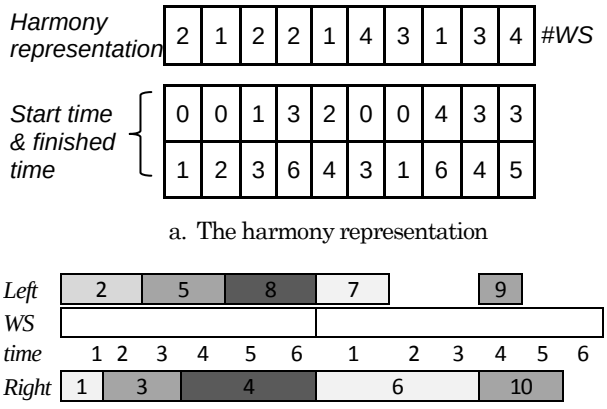


Figure 2. A problem example



a. The harmony representation

b. Illustration of task schedule in the assembly line

Figure 3. The harmony representation and the sequence of the tasks

Table 1. Example of task assignment

Task assigned							
Left side				Right side			
#WS	Task	t_i^s	t_i^f	#WS	Task	t_i^s	t_i^f
1	2	0	2	2	1	0	1
	5	2	4		3	1	3
	8	4	6		4	3	6
3	7	0	1	4	6	0	3
	9	3	4		10	3	5

The number of solution vector equal to the size of HMS. In this paper, we adopt the COMSOAL heuristic, Arcus [1] to generate the initial harmony memory. The procedure is described as follow:

Step 1 : Set $j = 1$,
Step 2 : Create a new harmony vector. List all unassigned tasks (*UT*).
Step 3 : From *UT*, find all tasks in which their predecessor have been assigned (*FT*)
Step 4 : Randomly select a tasks from *FT* that will be assigned in the workstation
Step 5 : If the selected task has specific operations direction, put the task in the appropriate side. If the finished time of the task less than or equal to the desired cycle time *C*, put the task in the current workstation, otherwise open new workstation. If the selected task can be placed in either side, select the side in which it finish early. If both sides have the same finished time, then select the appropriate side randomly.
Step 6 : Remove the selected task for *UT*
Step 7 : If all task have been assigned, then go to step 8, otherwise, go to step 3
Step 8 : If $j = HMS$, then stop. Otherwise $j = j+1$ and go to step 2

The new harmony vector is improvised according to three rules: random selection, HM consideration and

pitch adjustment. The new harmony improvisation procedure is done to assign the tasks into mated workstation. The procedure is also based on COMSOAL heuristic, Chutima and Chimklai [6], however, the HM consideration and pitch adjustment are added in the procedure. The improvisation, HM updates and termination criterion are described as follow:

Step 1 : Improvisation
Step 1.1 : Create a new harmony vector. List all unassigned tasks *UT*.
Step 1.2 : Select a tasks from *UT* that will be assigned in a workstation. Find the earliest *e* and the latest *l* of feasible mated-workstation for the selected task
Step 1.3 : Generate random value *r*, if $r < HMCR$ go to Step 1.4. Otherwise go to Step 1.6
Step 1.4 : Generate a random integer value *k* between [*1, HMS*] to select a potential mated-workstation *w* for the selected task from the harmony memory. If $e \leq w \leq l$, assigned the selected task in *w,* otherwise, go to Step 1.6
Step 1.5 : Generate random value *u*, if $u < HMCR*PAR$ assigned the selected task to mated workstation *w-1* when it is not violated the precedence and cycle time constraint. Go to Step 1.7.
Step 1.6 : Assigned the selected task in the mated-workstation between *e* and *l* in which it is finished early without violated the precedence and cycle time constraint.
Step 1.7 : Remove the selected task from *UT*. If all tasks have been assigned, then stop, otherwise go to Step 2.
Step 2 : Evaluate the new harmony vector
Step 3 : If the new harmony vector is better than the worst solution in HM, then the worst solution in HM is replaced by the new harmony vector
Step 4 : If the maximum number of improvisation is reached, terminate the search, otherwise, go to Step 1.

In the procedure above, Step 1.4 is the harmony memory consideration and Step 1.5 is the pitch adjustment. The harmony memory consideration is illustrated in Figure 4.

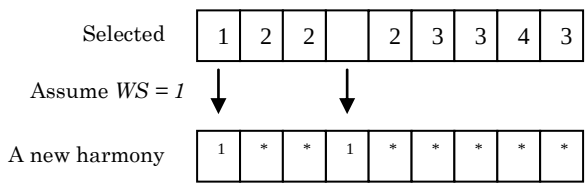


Figure 4. Harmony memory consideration for TALBP

In the procedure, after all tasks have been assigned in the workstation, the tasks in each workstation are sequenced based on the model discussed above. Then, the new solution vector is evaluated whether it is better than any solution vector in the HM. HM is updated when the new solution vector has better value than the worst solution vector in the HM. In this situation, the worst solution vector in HM is removed and the new solution is included. The computation is terminated when the maximum number of improvisation is reached.

The lower bound is given by:

$$LB_L = \left\lceil \frac{\sum_{i \in S_L} \max_m\{t_{im}\}}{C} \right\rceil \quad (12)$$

$$LB_R = \left\lceil \frac{\sum_{i \in S_R} \max_m\{t_{im}\}}{C} \right\rceil \quad (13)$$

$$LB_E = \left\lceil \frac{\sum_{i \in S_E} \max_m\{t_{im}\} - \left((LB_L + LB_E)C - \sum_{i \notin S_E} \max\{t_{im}\}\right)}{C} \right\rceil \quad (14)$$

$$LB = LB_L + LB_R + LB_E \quad (15)$$

Results and Discussions

Numerical Examples

The numerical examples are derived from benchmark problems for TALBP. The problems are represented as precedence of tasks that must be sequenced in the assembly line. Each task in the problem has specific processing time, desired processing orientation, direct successors as well as direct ancestors. The problems are named based on the number of its tasks; P12, P16, P24, P65, P148 and P205. Problems P12 and P24 consist of 12 and 24 task respectively and are described in Kim *et al.* [11]. P16 can be found in Kim *et al.* [12] and it consists of 16 tasks. Problem P65 consists of 65 tasks and P205 consist of 205 tasks. Both problems can be found in Lee *et al.* [13]. P148 is constructed by Bartholdi [3] and it consists of 148 tasks. We use the modified P148 as in Kim *et al.* [12], where the processing time of tasks 79 and 108 were changed from 281 to 111 and from 383 to 43.

The harmony search parameters are set as follow, HMS = 100, HMCR = 0.5, stopping criteria = 50 times of objective function evaluations for small sized problems and 500 times of objective function evaluations for large sized problems. In our proposed method, the pitch adjustment is conducted to assign the empty position in the new harmony vector; therefore, the PAR is not given. For the ordinary COMSOAL, we used the best solution of 50 tries and 500 tries in each runs for small size problems and large size problems respectively. Each problem is run for 20 times.

The first experiment is conducted on small size problems (P12, P16 and P24) to compare the performance of the proposed method with the COMSOAL and the optimal solution. Table 2 shows that the harmony search could obtain optimal solution in all the problems during 20 runs (the best result of 20 runs). However, in problems (P16, C 6) and (P 24, C 23), the method cannot reach the optimal solution in every run. Nevertheless, the harmony search still outperforms the basic COMSOAL method in all these problems except for P16 C 25 (both methods perform the same). In addition, the COMSOAL unable to reach optimal solution in three problems: (P12 C4), (P16 C 20) and (P24 C 23).

The second experiment is conducted on large size problems (P65, P148 and P205) to compare the performance of the proposed method with the COMSOAL, heuristic rules (H) and Genetic algorithm (GA) reported by Lee *et al.* [13], 2-ANTBAL reported by Simaria and Vilarinho[18]. The results are given in Table 3.

The experiment results for large problems show that the harmony search performs better in 13 problems, have the same performance in 3 problems and worse in 1 problem (P65 C 544) when compared to the COMSOAL. The proposed method also outperforms the heuristic method (H) in 16 out of 17 problems (both methods have the same performance for (P148 C 408). The proposed method also overcomes the genetic algorithm Lee *et al.* [13] in most of the problems (outperforms in 11 out of 17 problems and have the same performance in 4 problems). The genetic algorithm only produces better results on (P65 C544) and (P 148 C 357). In addition, the proposed method approximately performs the same for P65, slightly worse for P 148 and slightly better for P 205 when compared with the 2ANTBAL. Table 3 shows that the proposed method has better performance compared to COMSOAL, genetic algorithms and heuristics (H) methods. It is also better than 2-ANTBAL when cycle time and problems are large.

Conclusion

This paper proposed harmony search method to address TALBP problems with the objective of minimizing the number of workstation for a given

Table 2. Performance for small size problems

Problems	C	Optimal solution	*HS*			COMSOAL		
			mean	min	max	mean	min	max
P12	4	7	*7.00*	*7*	*7*	8.1	8	9
	6	5	*5.00*	*5*	*5*	5.1	5	6
	7	4	*4.00*	*4*	*4*	4.3	4	5
P16	16	6	*6.30*	*6*	*7*	6.6	6	8
	20	5	*5.00*	*5*	*5*	6.0	6	6
	25	4	*4.00*	*4*	*4*	4.0	4	4
P24	19	8	*8.00*	*8*	*8*	8.1	8	9
	23	7	*7.05*	*7*	*8*	8.0	8	8
	25	6	*6.00*	*6*	*6*	6.9	6	8

Table 3. Result for large size problems

Problems	C	LB	HS			Lee *et al.* [13]		2-ANTBL (Simaria, [18])			COMSOAL		
			mean	min	max	mean H	mean GA	mean	min	max	mean	min	max
P65	381	14	15.0	15	15	15,7	15.0	**14.8**	14	15	15.0	15	15
	435	12	**13.0**	13	13	14.0	13,4	**13.0**	13	13	**13.0**	13	13
	490	11	**11.4**	11	12	12.1	12.0	12.0	12	12	12.0	12	12
	544	10	11.0	11	11	11.5	**10,6**	10.8	10	11	10.9	10	11
P148	306	17	**18.0**	18	18	19.3	18.0	**18.0**	18	18	19.3	18	20
	357	15	15.9	15	16	16.0	**15.0**	15.4	15	16	16.0	16	16
	408	13	**14.0**	14	14	**14.0**	**14.0**	**14.0**	14	14	**14.0**	14	14
	459	12	**12.0**	12	12	12.1	13.0	**12.0**	12	12	13.1	12	14
	510	11	**11.0**	11	11	12.0	**11.0**	**11.0**	11	11	12.0	12	12
P205	1133	21	22.5	22	23	24.0	23.0	**22.4**	22	23	25.9	24	26
	1322	18	**19.9**	19	20	21.9	20,7	20.0	20	20	22.0	22	22
	1510	16	**17.0**	17	17	18.7	20.0	17.2	17	18	19.4	18	20
	1699	14	**15.1**	15	16	16.7	16.0	15.8	15	16	17.2	16	18
	1888	13	14.0	14	14	15.4	16.0	**13.8**	13	14	16.0	16	16
	2266	11	**12.0**	12	12	12.5	13.0	**12.0**	12	12	12.3	12	14
	2643	9	**10.0**	10	10	11.2	12.0	**10.0**	10	10	11.9	10	12
	2832	9	**9.0**	9	9	10.0	10.0	10.0	10	10	10.0	10	10

cycle time. In the proposed method, the harmony memory initialization adopts the COMSOAL heuristics and the new harmony improvisation adopts the greedy search for ALBP. The proposed method is evaluated based on 6 benchmark problems: P12, P16, P24, P65, P148 and P205. The experiment results convince that the proposed method work well. The experiments show that the proposed method outperform the ordinary COMSOAL heuristics, the heuristic and the genetic algorithm, Lee *et al.*[13] and approximately the same as 2ANTBAL, Simaria and Vilarinho [18].

For future development, the proposed method can be extended to solve various meaningful objectives and practical constraints existing in the real problems such as considering the stochastic processing time, mixed products, and learning process of the operators.

References

1. Arcus, A. L., COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines, *International Journal of Production Research*, 4, 1966, pp. 259-277.
2. Askin, R. G., and Standridge, C. R., *Modeling and Analysis of Manufacturing Systems*, John Wiley and Sons, Florida, 1993.
3. Bartholdi, J. J., Balancing Two-sided Assembly Lines: A Case Study, *International Journal of Production Research*, 31, 1993, pp. 2447-2461.
4. Boysen, N., Fliedner, M., and Scholl, A., A Classification of Assembly Line Balancing Problems, *European Journal of Operational Research*, 183, 2007, pp. 674-689.
5. Boysen, N., Fliedner, M., and Scholl, A., Assembly Line Balancing: Which Model to Use When?, *International Journal of Production Economics*, 111, 2008, pp. 509-528.
6. Chutima, P., and Chimklai, P., Multi-Objective Two-sided Mixed-model Assembly Line Balancing using Particle Swarm Optimization with Negative Knowledge, *Computers & Industrial Engineering*, 62(1), 2012, pp. 39-55.
7. Geem, Z. W., Kim, J. H., and Loganathan, G. V., A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*, 76, 2001, pp. 60-68.
8. Geem, Z. W., Lee, K. S., and Park, Y., Application of Harmony Search to Vehicle Routing, *American Journal of Applied Sciences*, 2, 2005, pp. 1552-1557.
9. Geem, Z. W., Harmony Search Application in Industry, In B. Prasad (ed): *Soft Computing Applications in Industry*, Springer-verlag, Berlin Heidelberg, 2008, pp. 117-134.
10. Gutjahr, A. L., and Nemhauser, G. L., An Algorithm for the Line Balancing Problem, *Management Science*, 11(2), 1964, pp. 308–315.
11. Kim, Y. K., Kim, Y., and Kim, Y. J., Two-sided Assembly Line Balancing: A Genetic Algorithm Approach, *Production Planning & Control*, 11, 2000, pp. 44-53.
12. Kim, Y. K., Song, W. S., and Kim, J. H., A Mathematical Model and a Genetic Algorithm for Two-sided Assembly Line Balancing, *Computers and Operations Research*, 36, 2009, pp. 853-865.
13. Lee, T. O., Kim, Y., and Kim, Y. K., Two-sided Assembly Line Balancing to Maximize Work Relatedness and Slackness, *Computer and Industrial Engineering*, 40, 2001, pp. 273-292.
14. Özbakır, L., and Tapkan, P., Bee Colony Intelligence in Zone Constrained Two-sided Assembly Line Balancing Problem, *Expert Systems with Applications*, 38, 2011, pp. 11947-11957.
15. Özcan, U. u., Balancing Stochastic Two-sided Assembly Lines: A Chance-constrained, Piecewise-linear, Mixed Integer Program and a Simulated Annealing Algorithm, *European Journal of Operational Research*, 205, 2010, pp. 81-97.
16. Purnomo, H. D., Wee, H. M., and Rau, H., Two-sided Assembly Lines Balancing with Assignment Restrictions, *Mathematical and Computer Modeling*, 2011, doi:10.1016/ j.mcm.2011.06.010.
17. Scholl, A., Fliedner, M., and Boysen, N., ABSALOM: Balancing Assembly Lines with Assignment Restrictions, *European Journal of Operational Research*, 200, 2010, pp. 688-701.
18. Simaria, A. S., and Vilarinho, P. M., 2-ANTBAL: An Ant Colony Optimization Algorithm for Balancing Two-sided Assembly Lines, *Computers and Industrial Engineering*, 56, 2009, pp. 489-506.