ICIC Express Letters
Volume 12, Number 11, November 2018

ICIC International ©2018 ISSN 1881-803X
pp. 1123–1130

MULTI-OBJECTIVE U-SHAPED ASSEMBLY LINE BALANCING UNDER MACHINE DETERIORATION AND PREVENTIVE MAINTENANCE

Zikai Zhang[1,2], Qiuhua Tang[1,2,*] and Chenhao Lu[3]

[1]Key Laboratory of Metallurgical Equipment and Control Technology
[2]Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering
Wuhan University of Science and Technology
No. 947, Heping Ave., Qingshan Dist., Wuhan 430081, P. R. China
zhangzikai0703@gmail.com; *Corresponding author: tangqiuhua@wust.edu.cn

[3]Dongfeng Renault Automotive Company
No. 1118, Hanyang Ave., Hanyang Dist., Wuhan 430056, P. R. China
luch@dongfeng-renault.com.cn

Received May 2018; accepted August 2018

Abstract. *In manufacturing industries, some operating machines may become unavailable and hence need to be maintained preventively so as to keep it at the desired reliability, and the original scheduling plan is not applied to current situation. Thus, this paper investigates the incorporation of balance and preventive maintenance in U-shaped assembly line. And two objectives including cycle time and assignment alteration for this new problem need to be optimized. This problem contains two stages in which the first is regular U-shaped assembly line balancing problem and the second stage reassigned tasks into the machines which are not maintained in next cycle. Later, this paper also designed two meta-heuristic algorithms including elitist non-dominated sorting genetic algorithm and multi-objective simulated annealing algorithm to solve this problem. A hypothetical data set based on the benchmark instances is generated for the U-shaped assembly line balancing problem under machines deterioration and preventive maintenance. And the proposed two meta-heuristics are employed to obtain the Pareto frontier of these benchmark instances and their results are compared with each other.*

Keywords: U-shaped assembly lines, Preventive maintenance, Meta-heuristic algorithms

1. Introduction.

U-shaped assembly line is widely utilized in manufacturing systems. It is responsible for the scheduling of tasks in workstations and balancing of the workstation workload. Since it was proposed and modeled by Miltenburg and Wijingaard [1], this high efficiency configuration has been widely studied in the literature. And according to the objective function, the U-shaped assembly line balancing problem (UALBP) can be divided into three sub-problems. (1) For a given cycle time, the tasks are assigned to workstation in the U-shaped assembly line to minimize the number of workstations. This problem is named as type-I UALBP. (2) For a given number of workstations, the objective function is to optimize the cycle time, which is called as type-II UALBP. (3) Without the given cycle time or the number of workstations, maximizing the line efficiency is the mainly function. This problem is named as type-E UALBP. When assigning the tasks into workstations, it should meet the cycle time constraint and precedence relation constraint. Due to this feature, the U-shaped assembly line balancing problem belongs to the NP-hard problems.

Apart from the production scheduling, the machine maintenance also has impacts on the machine's capacity and reliability [2]. In general, machine maintenance includes two

DOI: 10.24507/icicel.12.11.1123

types: corrective maintenance (CM) and preventive maintenance (PM). CM is carried out when the machine has broken down to make it as good as new. PM is taken on a machine when it is still on operating in order to keep it at the desired level and avoid breakdown [3]. In most industrial enterprises, the production will be halted to carry out the PM on some machines, which brings great losses.

Thus, this paper formulates the preventive maintenance integrated in the U-shaped assembly line balancing problem aiming to promise the smooth production when the PM is carried out and hence reduce production cost. However, to our knowledge, there is no research studying the U-shaped assembly balancing problem under preventive maintenance. In this paper, the first contribution is to formulate this new problem to optimize the cycle time and the assignment alteration simultaneously. The former is the regular objective of assembly line to present the production improvement, and the latter is a new proposed objective to reduce the task alteration. And the second contribution is to design two meta-heuristics including elitist non-dominated sorting genetic algorithm (NSGA-II) and multi-objective simulated annealing algorithm (MOSA) since these two algorithms have great performance in the regular UALBP. The Pareto frontiers obtained by these two algorithms are compared with each other.

2. Problem Statement.

Different from the regular U-shaped assembly line balancing problem, this new problem contains two stages where the first stage represents normal U-shaped assembly line balancing and the second stage is the maintenance period. Meanwhile, before assigning the tasks into workstations in these two stages, we should firstly allocate the machines to each station. Each workstation must be equipped with one machine, and the number of machines is equal to that of stations. After the allocation of machines is determined, it keeps unchanged in two stages. And since the processing time of task depends on the machines, the processing time of task can be determined when the task is assigned to a workstation.

In the first stage, each machine is on work and the tasks are assigned to workstations. Here, we describe the first stage as follows. A set of tasks i $\{i = 1, 2, 3, \ldots, n\}$ needs to be assigned to the entrance subline or exit subline of workstation j $\{j = 1, 2, 3, \ldots, m\}$. Among them, if a task is selected to be assigned due to its all allocated immediate predecessors, it is assembled on the entrance subline of current station. On the contrary, since its immediate successors have been assigned, this task is assembled on the exit of current station. And then, the maximum workstation time is regarded as the first cycle time.

In the second stage, some machines need to be maintained and the tasks need to be reassigned to the workstations where the machine is operating. Compared with the first stage, since the number of machines has decreased, the corresponding number of workstations is also smaller than that in the first stage. In this situation, the workstations where the machines need to be maintained are not allocated with tasks. The second objective is assignment alteration, which can be calculated by the equation $\sum_i^n \left| \sum_j^m j \times (X_{ij,1} + Y_{ij,1}) - \sum_j^m j \times (X_{ij,2} + Y_{ij,2}) \right|$, where $X_{ij,1}$ and $Y_{ij,1}$ respectively are task i allocated to the entrance subline or exit subline of workstation j in the first stage, and $X_{ij,2}$ and $Y_{ij,2}$ are in the second stage.

3. Meta-Heuristic Algorithms.

Since the U-shaped assembly line balancing problem is NP-hard problem, with the increase of problem's scale, it can be difficult to obtain the optimal solutions. However, the meta-heuristic algorithm can solve the large and complex problems with an acceptable time [4]. So to achieve the optimization of U-shaped assembly line balancing problem under machines deterioration and preventive maintenance, this paper designs NSGA-II and MOSA, which have been proved to be effective and efficient for large optimization problem. The detail of these algorithms is described as follows.

3.1. **Solution representation.** In the NSGA-II and MOSA, solutions are constructed according to two numeric strings in which the first string forms with a set of machine numbers and the second is composed of rule numbers. For the first numeric string, its length is equal to the number of workstations and each value is randomly generated with non-repeating. For the second numeric string, the length is equal to the number of tasks, and each value is randomly generated between $[1, 10]$. Among them, these heuristic rules are proposed by Baykasoğlu and Özbakır [5] to obtain high quality solutions. For example, a code is constructed as $\{3, 1, 2|3, 2, 1, 1, 6, 3, 4, 5, 6\}$. In this code, the first numeric string is $\{3, 1, 2\}$ which means the workstations 1, 2 and 3 are respectively allocated with machines 3, 1 and 2. And the second numeric string is $\{3, 2, 1, 1, 6, 3, 4, 5, 6\}$ in which the first assigned task is selected from the candidate set based on the rule 3. And a task is put into candidate set if the task is unassigned and its immediate predecessors or immediate successors have been assigned.

Observing the above code, we can find that we just know the allocation of machines while the assignment of tasks into predefined workstation in each stage and the objective functions are unknown. Thus, this paper employs the decode mechanism proposed by Zhang et al. [6,7], in each stage to obtain the information and make the above code into a feasible balancing solution.

3.2. **Elitist non-dominated sorting genetic algorithm.** Elitist non-dominated sorting genetic algorithm (NSGA-II) was first proposed by Deb et al. [8] to deal with bi-objective optimization problems. This algorithm starts with an initial population and then explores the solution space by crossover, mutation and selection operators. Among them, considering the characteristic of U-shaped assembly line balancing problem under machine deterioration and preventive maintenance, this paper designs two types crossover operators and mutation operators on the task and machine vectors respectively. In the selection operator, this algorithm selects new population based on the crowded distance and Pareto stratum. The procedure of the proposed NSGA-II is shown in Figure 1 and the detail is described as follows.

(1) Initiation. The NSGA-II algorithm generates PS solutions by the above proposed encoding and decoding mechanism in the initiation. And then, an archive set is created to store the Pareto frontier. The non-dominated solutions in the initial population are put into this archive.

(2) Crossover. In this study, to promise the global optimization, two crossover operators are implemented. These operators exchange the elements of two parents in task and machine vectors respectively. After the crossover, the offspring chromosomes also meet the precedence relationship.

Task crossover: a random point is generated in parent 1, and the tasks before this point are put into the offspring 1 at the same position. And then, the tasks in parent 2 are successively put behind the offspring 1 without repetition.

Machine crossover: this operator is similar to the task crossover, but it is carried out on the machine vector.

(3) Mutation. Two mutation operators are adopted in this study. In the task mutation, a task is randomly selected and then moved to another position. After moving, the new task vector should meet the precedence relationship. If violating, this task is removed until meeting the precedence relationship. For the machine vector, two machines are selected and swapped their position. At the same time, the process mode of two mutation operators is similar to that of crossovers.

(4) Selection. After crossover and mutation, we update the archive Pareto frontier and select new solutions for next iteration in selection operator. For each offspring, if it is not dominated by the Pareto frontier, it is added into the archive and the solutions in the Pareto frontier dominated by this offspring are removed from the archive. Then

```
Algorithm: NSGA-II algorithm
 1: Input the parameters including the population size ($PS$), crossover
    rate ($CR$) and mutation rate ($MR$);
 2: Generate initial population by the encode and decode;
 3: Update the Pareto frontier;
 4: **For** the Stopping Criterion is not satisfied **do**
 5:     **For** each parent **do**
 6:         Randomly generate numbers $a_1$ and $a_2$;
 7:         **If** $a_1 < CR$ & $a_2 < CR$ **do**
 8:             Create the child by applying crossover on task and machine;
 9:         **If** $a_1 < CR$ & $a_2 > CR$ **do**
10:             Create the child by applying crossover on task;
11:     **End For**
12:     **For** each parent **do**
13:         Randomly generate numbers $a_3$ and $a_4$;
14:         **If** $a_3 < MR$ & $a_4 < MR$ **do**
15:             Create the child by applying mutation task and machine;
16:         **If** $a_3 < MR$ & $a_4 > MR$ **do**
17:             Create the child by applying mutation on task;
18:     **End For**
19:     Update the Pareto frontier;
20:     Selection parents for next iteration;
21: **End For**
22: **Return** Pareto frontier;
```

Figure 1. The flowchart of NSGA-II

we combine the parent and offspring populations to a new population. And this new population is divided into different levels based on the Pareto. The solution in the higher level is dominated by the solution in the lower level. If the number of solutions (*ns*) in the first *i*-th levels is not more than *PS* and that in the first $(i+1)$th levels is more than *PS*, the *ns* solutions in the first *i*-th levels are selected and *PS*-*ns* solution are selected from the $(i+1)$th level based on the crowd distance.

3.3. **Multi-objective simulated annealing algorithm.**

This algorithm starts with one solution and then generates new solution by its neighbor structure. Finally, a particular acceptance criterion is employed to judge whether this neighbor solution is accepted. To deal with our new problems, we also propose a multi-objective simulated annealing algorithm (MOSA) to optimize the cycle time and assignment alteration simultaneously. The procedure of MOSA is shown in Figure 2 and the details are described as follows.

(1) Initiation. This algorithm first sets T_0, α and IT, and set current temperature $T = T_0$. And an initial solution X_0 is generated by above encoding and decoding mechanism and set current solution $X_c = X_0$. Meanwhile, put the initial solution into archive Pareto frontier.

(2) Neighbor structures. A neighbor solution X_1 is obtained from current solution. In the proposed MOSA, we regard the above mutation operators as neighbor structures and then create neighbor solution. And these structures are used in the following operator. Two numbers a_1 and a_2 are generated between $[0, 1]$. If $a_1 < 0.5$ and $a_2 < 0.5$, create the neighbor solution X_1 by the two neighbor structure; if $a_1 < 0.5$ & $a_2 > 0.5$, create the neighbor solution X_1 by the task mutation; otherwise, create the neighbor solution X_1 by the machine mutation.

```
Algorithm: MOSA algorithm
 1: Input the parameters including the initial temperature ($T_0$), cooling
    rate ($\alpha$) and maximum number of iterations ($IT$);
 2: Generate one initial solution $X_0$ by the encode and decode;
 3: Put the initial solution into Pareto frontier;
 4: For the Stopping Criterion is not satisfied do
 5:     $counter = 0$;
 6:     If $counter < IT$ do
 7:         Randomly generate numbers $a_1$ and $a_2$;
 8:         If $a_1 < 0.5$ & $a_2 < 0.5$ do
 9:             Create the neighbor solution $X_1$ by the two neighbor structure;
10:         Else If $a_1 < 0.5$ & $a_2 > 0.5$ do
11:             Create the neighbor solution $X_1$ by the first structure;
12:         Else
13:             Create the neighbor solution $X_1$ by the second structure;
14:         If $X_1$ is non-dominated by the Pareto frontier do
15:             Update the Pareto frontier;
16:         Else
17:             randomly select one objective function $f$;
18:             $\Delta = f(X_1) - f(X_0)$;
19:             If $\Delta < 0$ do
20:                 $X_0 = X_1$;
21:             Else If $EXP(-\Delta/T) >$ random number do
22:                 $X_0 = X_1$;
23:         counter = counter + 1;
24:     Else
25:         $T = \alpha \times T$;
26: End For
27: Return Pareto frontier;
```

Figure 2. The flowchart of MOSA

(3) Pareto frontier update and acceptance criterion. In above steps, a neighbor solution is obtained, and then this solution should be checked whether it can be dominated by the Pareto frontier and whether it can replace the current solution. If the neighbor solution cannot be dominated by the Pareto frontier, this solution is put into the Pareto frontier and then the solutions dominated by the neighbor solution in the Pareto frontier are removed. Meanwhile, the neighbor solution replaces current solution to explore its neighbor structure. If not, a multinomial probability mass function proposed by Kulturel-Konak et al. [9] is applied to checking whether the neighbor solution can replace the incumbent one. For the multi-objective problems, we first randomly select an objective $f(x)$, and then calculate the difference Δ ($\Delta = f(\textit{neighbor solution}) - f(\textit{current solution})$). If the difference is less than 0 or the $EXP(-\Delta/T)$ exceeds a random number, the neighbor solution is accepted as the current solution.

4. Result Comparison.

In this section, a set of benchmark instances is utilized to analyze the performance of the two proposed meta-heuristic algorithms. The precedence relationship and original processing time of these problems come from the website: http://assemb-ly-line-balancing.mansci.de/. And the processing time of task i by machine h is randomly generated between $[t_i \times 0.8, t_i \times 1.2]$ [10], in which t_i is the original processing time. And in the second cycle time, the machines performing maintenance are randomly determined. All these experiments are encoded in C++ programming language and are run on a computer with Intel(R) Core(TM) i5 CPU, 2.80GHz and 2.00GB RAM. Besides, the relevant parameters for these two meta-heuristics are shown in Table 1.

To compare the performance of the two proposed meta-heuristic algorithms with each other, we employ above 32 instances to check it and each instance is respectively run 10 times under five iteration criteria ($N_i \times N_i \times 10$, $N_i \times N_i \times 20$, $N_i \times N_i \times 30$, $N_i \times N_i \times 40$ and $N_i \times N_i \times 50$ millisecond, where N_i is the number of tasks) for total 3200 experiments (2 algorithms $\times$ 32 instances $\times$ 10 times $\times$ 5 iteration criteria). And one multi-objective evaluation indicator, named hyper volume rate (HVR) [11], is utilized to evaluate the

Table 1. The parameters for the NSGA-II and MOSA

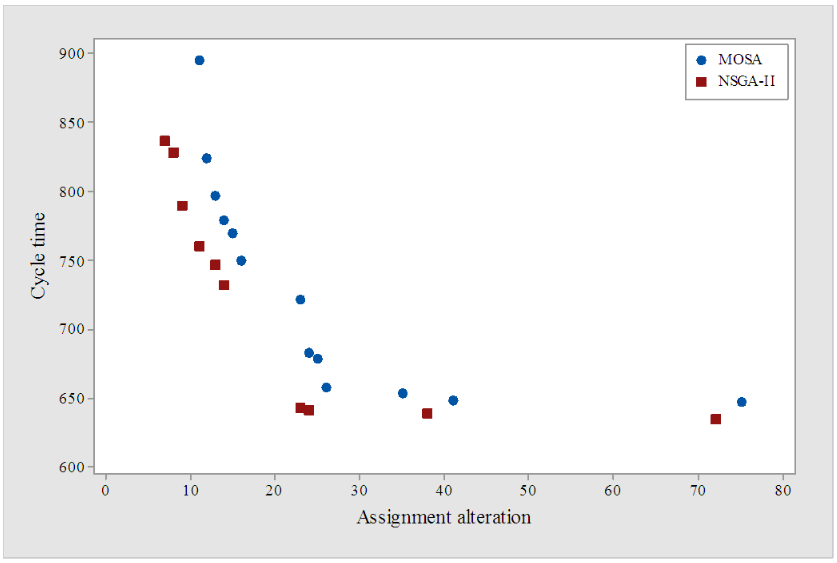
Parameters	Symbol	Value
NSGA-II		
Population size	PS	120
Crossover rate	CR	0.8
Mutation rate	MR	0.2
MOSA		
Initial temperature	T_0	0.5
Cooling rate	α	0.9
maximum number of iterations	IT	500

Table 2. The results of the NSGA-II and MOSA

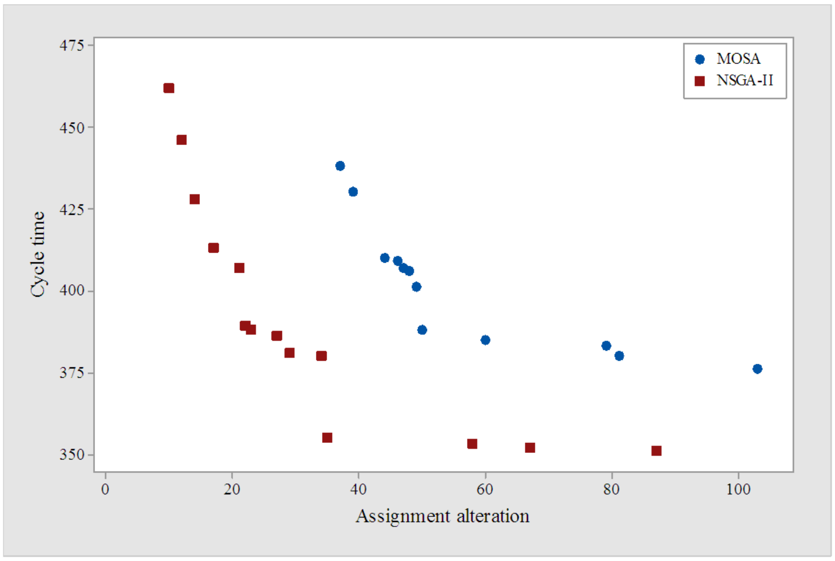
Instance	$N_i \times N_i \times 10$		$N_i \times N_i \times 20$		$N_i \times N_i \times 30$		$N_i \times N_i \times 40$		$N_i \times N_i \times 50$	
	MOSA	NSGA-II	MOSA	NSGA-II	MOSA	NSGA-II	MOSA	NSGA-II	MOSA	NSGA-II
1	0.91	**1.00**	0.91	**1.00**	0.84	**1.00**	0.89	**1.00**	0.91	**1.00**
2	0.90	**1.00**	0.90	**1.00**	0.67	**1.00**	0.83	**1.00**	0.78	**1.00**
3	0.93	**1.00**	0.84	**1.00**	0.75	**1.00**	0.83	**1.00**	0.84	**1.00**
4	0.85	**1.00**	0.84	**1.00**	0.91	**1.00**	0.79	**1.00**	0.79	**1.00**
5	0.91	**1.00**	0.87	**1.00**	0.88	**1.00**	0.85	**1.00**	0.85	**1.00**
6	0.88	**1.00**	0.92	**1.00**	0.90	**1.00**	0.85	**1.00**	0.82	**1.00**
7	0.89	**1.00**	0.85	**1.00**	0.83	**1.00**	0.84	**1.00**	0.69	**1.00**
8	0.74	**1.00**	0.65	**1.00**	0.78	**1.00**	0.76	**1.00**	0.67	**1.00**
9	**1.00**	0.86	**1.00**	0.89	**1.00**	0.94	**1.00**	0.95	**1.00**	0.89
10	**1.00**	0.96	**1.00**	0.94	**1.00**	0.95	**1.00**	0.96	**1.00**	0.93
11	**0.98**	**0.98**	0.93	**1.00**	**0.95**	0.93	**0.97**	0.92	0.88	**1.00**
12	0.91	**1.00**	0.95	**1.00**	0.90	**1.00**	0.97	**1.00**	0.89	**1.00**
13	0.85	**1.00**	0.86	**1.00**	0.86	**1.00**	0.78	**1.00**	0.81	**1.00**
14	0.87	**1.00**	0.83	**1.00**	0.82	**1.00**	0.81	**1.00**	0.63	**1.00**
15	0.71	**1.00**	0.72	**1.00**	0.75	**1.00**	0.70	**1.00**	0.68	**1.00**
16	0.78	**1.00**	0.71	**1.00**	0.67	**1.00**	0.67	**1.00**	0.57	**1.00**
17	0.92	**1.00**	0.92	**1.00**	0.93	**1.00**	0.90	**1.00**	0.95	**1.00**
18	**0.99**	0.96	0.85	**0.92**	**0.96**	**0.96**	**0.99**	0.92	**1.00**	0.90
19	**0.99**	0.86	**1.00**	0.94	0.93	**1.00**	**0.99**	0.98	0.94	**0.95**
20	0.98	**1.00**	0.94	**0.99**	0.93	**1.00**	0.96	**1.00**	0.96	**1.00**
21	0.88	**1.00**	0.92	**1.00**	0.90	**1.00**	0.94	**0.99**	0.91	**1.00**
22	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
23	**1.00**	0.92	**1.00**	0.99	0.95	**0.99**	**1.00**	0.95	0.95	**0.99**
24	0.95	**0.99**	0.92	**1.00**	0.94	**1.00**	**0.97**	0.93	0.91	**1.00**
25	0.92	**1.00**	0.90	**1.00**	0.83	**1.00**	0.79	**1.00**	0.83	**1.00**
26	0.86	**1.00**	0.86	**1.00**	0.74	**1.00**	0.68	**1.00**	0.76	**1.00**
27	0.87	**1.00**	0.78	**1.00**	0.75	**1.00**	0.72	**1.00**	0.78	**1.00**
28	**1.00**	**1.00**	**1.00**	**1.00**	1.00	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
29	0.94	**1.00**	0.80	**1.00**	0.93	**1.00**	0.78	**1.00**	0.83	**1.00**
30	0.89	**1.00**	0.87	**1.00**	0.78	**1.00**	0.77	**1.00**	0.79	**1.00**
31	0.86	**1.00**	0.78	**1.00**	0.76	**1.00**	0.79	**1.00**	0.80	**1.00**
32	0.93	**1.00**	0.93	**0.98**	0.95	**0.99**	0.94	**0.99**	0.90	**1.00**

performance of multi-objective algorithms. This indicator is the ratio between the hyper volume of the obtained Pareto set and that of the true Pareto frontier. The results are presented in Table 2.

From this table, it can be observed that in most instances, the value of HVR obtained by NSGA-II is closer to 1 compared with that obtained by MOSA under five iteration criteria while only in four instances (instances 9, 10, 18 and 19), the HVR of MOSA is larger than that of NSGA-II. These results indicate that the Pareto frontier obtained by NSGA-II is the better approximation to the true frontier. And we also depict the comparison of the Pareto frontiers obtained by these algorithms in Figure 3 to draw the conclusion more intuitively. This figure depicts two instances under different iteration criteria. From this figure, we can find that the solutions obtained by MOSA are absolutely dominated by the Pareto frontier achieved by NSGA-II, which suggests that NSGA-II outperforms MOSA in U-shaped assembly line balancing under machine deterioration and preventive maintenance.



(a)



(b)

FIGURE 3. The comparison of Pareto frontiers for (a) Lutz89 with 12 machines under $N_i \times N_i \times 30$ and (b) Barthol148 with 29 machines under $N_i \times N_i \times 50$

5. **Conclusions.** In manufacturing systems, both the production scheduling and preventive maintenance are important problem. Current researches optimize these two problems separately, which cannot promise the production running smoothly. Thus, this paper aims to balance the U-shaped assembly lines under machine deterioration and preventive maintenance. It has great practical significance on the industry. With respect to this new problem, we first formulate it and then design two meta-heuristics including NSGA-II and MOSA to minimize the cycle time and task assignment alteration simultaneously. Meanwhile, computational experiments are conducted on benchmark to test the performance of the proposed algorithms, and a Pareto-compliant performance indicator HVR is employed to evaluate them. The experiment results indicate that the NSGA-II is superior to MOSA in most benchmark instances, which also suggests that the Pareto frontier obtained by NSGA-II has great convergence and diversity.

Future research will be extended to address U-shaped assembly line balancing problems in which movement of machines in cross workstation should be considered. And the preventive maintenance can be considered in two-sided assembly line balancing problem.

Acknowledgement. The authors would like to thank the anonymous reviewers for their helpful comments and constructive suggestions. This work is supported by National Natural Science Foundation of China (No. 51275366, No. 51305311).

REFERENCES

[1] G. J. Miltenburg and J. Wijingaard, The U-line line balancing problem, *Management Science*, vol.40, no.10, pp.1378-1388, 1994.

[2] M. Touat et al., A hybridization of genetic algorithms and fuzzy logic for the single-machine scheduling with flexible maintenance problem under human resource constraints, *Applied Soft Computing*, vol.59, pp.556-573, 2017.

[3] R. Ruiz, J. C. García-Díaz and C. Maroto, Considering scheduling and preventive maintenance in the flowshop sequencing problem, *Computers & Operations Research*, vol.34, no.11, pp.3314-3330, 2007.

[4] M. K. Oksuz, K. Buyukozkan and S. I. Satoglu, U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics, *Computers & Industrial Engineering*, vol.112, pp.246-263, 2017.

[5] A. Baykasoğlu and L. Özbakır, Stochastic U-line balancing using genetic algorithms, *The International Journal of Advanced Manufacturing Technology*, vol.32, nos.1-2, pp.139-147, 2006.

[6] Z. Zhang et al., Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment, *Neural Computing and Applications*, 2018.

[7] Z. Zhang, Q. Tang and L. Zhang, Concurrent optimization of worker and task assignment within U-shaped assembly lines via iterated greedy algorithm, *ICIC Express Letters*, vol.12, no.1, pp.79-86, 2018.

[8] K. Deb et al., A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evolutionary Computation*, vol.6, no.2, pp.182-197, 2002.

[9] S. Kulturel-Konak, A. E. Smith and B. A. Norman, Multi-objective tabu search using a multinomial probability mass function, *European Journal of Operational Research*, vol.169, no.3, pp.918-931, 2006.

[10] Z. Li, Q. Tang and L. Zhang, Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm, *Journal of Cleaner Production*, vol.135, pp.508-522, 2016.

[11] W. Zhao, S. Alam and H. A. Abbass, MOCCA-II: A multi-objective co-operative co-evolutionary algorithm, *Applied Soft Computing*, vol.23, pp.407-416, 2014.