ICIC Express Letters ICIC International ©2018 ISSN 1881-803X
Volume **12**, Number **1**, January **2018** pp. **79–86**

CONCURRENT OPTIMIZATION OF WORKER AND TASK ASSIGNMENT WITHIN U-SHAPED ASSEMBLY LINES VIA ITERATED GREEDY ALGORITHM

Zikai Zhang[1,2], Qiuhua Tang[1,2,*] and Liping Zhang[1,2]

[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 Avenue, Qingshan District, Wuhan 430081, P. R. China
[*]Corresponding author: tangqiuhua@wust.edu.cn

Received July 2017; accepted September 2017

Abstract. *When allocating tasks within U-shaped assembly lines, the ignorance of heterogeneous workers may result in the unaccomplishment of workload for the lack of skills and even cause the current task assignment infeasible. Thus, this paper optimizes worker assignment and task allocation within U-shaped assembly lines concurrently via an iterated greedy algorithm. In the algorithm, both task assignment and worker allocation vectors are embedded for the concurrent optimization and multiple heuristic rules are employed to extend the diversity of elements in the initial solution. A specified number of tasks and workers are extracted and reinserted respectively in destruction and construction phases for further exploration and exploitation. And the acceptance criterion is modified based on the temperature in simulated annealing algorithm to avoid trapping into local optima. Compared with optimal solutions by GAMS/Cplex and near-optimal solutions by particle swarm optimization of the benchmark problems, computational studies indicate that the iterated greedy algorithm shows excellent performance for small-scaled, middle-scaled and large-scaled benchmark problems.*

Keywords: U-shaped assembly lines, Assembly line balancing, Worker assignment, Iterated greedy algorithm

1. Introduction.

The U-shaped assembly line is increasingly utilized to produce high-volume standardized products such as electronics and appliances. It consists of a series of workstations connected by a U-shaped material handling system. These workstations may perform assembly tasks at either entrance subline or exit subline, or complete jobs first at entrance subline and then at exit subline. Any task for assembling these products can be allocated as long as its immediate predecessors or immediate successors have been completely assigned. Hence, the U-shaped line owns more allocation opportunity and achieves higher productivity and better flexibility compared with the traditional straight assembly line. The problem for balancing workload among these workstations is called the U-shaped assembly line balancing problem (UALBP).

Each worker performing assembly tasks has different ability and agility. These assembly skills play a determinant role on the processing time of assembly tasks [1]. The more skilled it is, the shorter the processing time is. If a task is allocated to a less skilled worker, this task may not be completed in a cycle time, causing the infeasibility of current task assignment and even production break. On the contrary, allocating a task to a worker with higher skills may lead to waste of production times. Thus, the assignment of multi-skilled workers should also be considered in UALBP. However, worker assignment related to UALBP has never been observed so far since it was first studied by Chaves et al. [2]. And then many researchers studied the worker assignment in straight assembly line [3-5]. Meanwhile, the iterated greedy algorithm (IG) is a local search method which has successful applications in discrete and combinational optimization problems. And the main feature of the IG algorithm is its simplicity, which means that this algorithm does not need to embed specific knowledge and has few control parameters [6]. Hence, this paper focuses on developing an iterated greedy algorithm. This paper mainly contains two contributions. (1) The new U-shaped assembly line worker assignment and balancing problem (UALWABP) is defined and a set of benchmark problems is generated. (2) An iterated greedy algorithm (IG) is developed to tackle the concurrent assignment of workers and tasks within U-shaped assembly lines.

The organization of this paper is as the following. Section 2 defines UALWABP and Section 3 describes the proposed iterated greedy algorithm to minimize cycle time. Section 4 reports the experimental results. Finally, Section 5 discusses the conclusion and future work.

2. **Problem Statement.**

Nowadays, most industries such as automotive and electronics employ human to assemble tasks. And each assembly worker has different ability and agility. These assembly skills play a determinate role on the processing time of assembly tasks. Hence, workers and tasks are expected to be allocated into workstations simultaneously. As long as the worker occupying the workstation is suitable, the objective of concurrent optimization is to improve the line efficiency by minimizing the cycle time.

Meanwhile, only one worker is allocated to a workstation and the number of workers is equal to that of workstations. Each task is allocated into exactly one workstation which locates either in entrance subline or exit subline. In a workstation performing tasks on both sublines, walk times of workers in the workstation is negligible. All tasks allocated to a workstation must be processed within the cycle time. Particularly, the precedence relations among the tasks are known and the allocation of tasks needs to satisfy the precedence relations.

3. **Modified Iterated Greedy Algorithm.**

The iterated greedy algorithm (IG) is a local search based meta-heuristic algorithm. It generates a series of solutions by repeating greedy constructive heuristics. In the destruction phase some elements are removed from the candidate solution and in the construction procedure they are reinserted back into the solution for constituting a new candidate solution. Besides, a local search phase is added to further improve the re-constructed solutions and an acceptance criterion is employed to decide whether the newly candidate solution will replace the incumbent solution. This procedure terminates till some predefined stopping criteria are met. Thus, a modified iterated greedy algorithm is developed here to tackle the concurrent assginment problem of workers and tasks within U-shaped assembly lines. Relative details of the proposed IG are reported in the following.

3.1. **Initialization and decoding.**

Since worker and task assignment are both involved, the task assignment and worker allocation vectors are both necessary for the representation of a solution. It is worthy noting that rule-based initialization approaches may promote the performance of solutions such as NEH (Nawaz-Enscore-Ham) heuristic for flow shop scheduling. Hence, nine heuristic rules are integrated here to determine the assignment order of tasks, including the shortest processing time, longest processing time, minimum total number of successor tasks, maximum total number of successor tasks, maximum total time of successor tasks, minimum total time of successor tasks, maximum total number of predecessor tasks, minimum total number of predecessor tasks and maximum total time of predecessor tasks. All the rules take values between one to the total number of heuristic rules. The length of the task assignment vector is equal to the number of tasks and the figure in the element represents the rule number. For worker allocation, each element represents the worker being allocated into the sequenced workstations. As

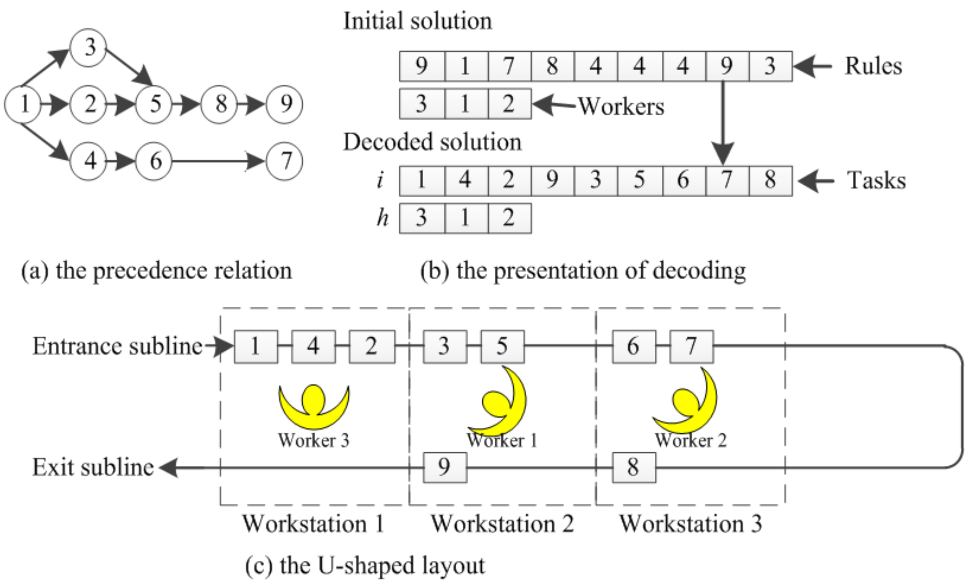

FIGURE 1. The presentation of the decoding and U-shaped layout

shown in Figure 1, both task assignment and worker allocation become encoded in a solution.

Based on this representation, the decode mechanism is subtly designed to assign all tasks into predefined workstations and to calculate the cycle time. While all tasks are not allocated into workstations, the remained tasks satisfying the precedence relations are selected into a candidate set and one task is chosen from the set according to the rule number of the task assignment vectors. Sequentially, the processing time of this task is determined on the ground of the worker allocation vector. If the new station time is not more than the cycle time, this task is allocated directly into the current workstation or otherwise the next.

3.2. **Destruction and construction phase.** In the destruction phase, d tasks and d workers are respectively and randomly extracted from the incumbent solution. And these extracted tasks and workers are respectively inserted into two lists of removed jobs A_{task} and B_{worker}. Then in the construction phase, the tasks and workers in the two lists A_{task} and B_{worker} are successively reinserted into the task assignment vector and worker allocation vector respectively of incumbent solution. Note that the reinserted positions of the tasks and workers must be different from their original positions and should satisfy the precedence relation constraints. The destruction and construction methods are depicted in Figure 2.

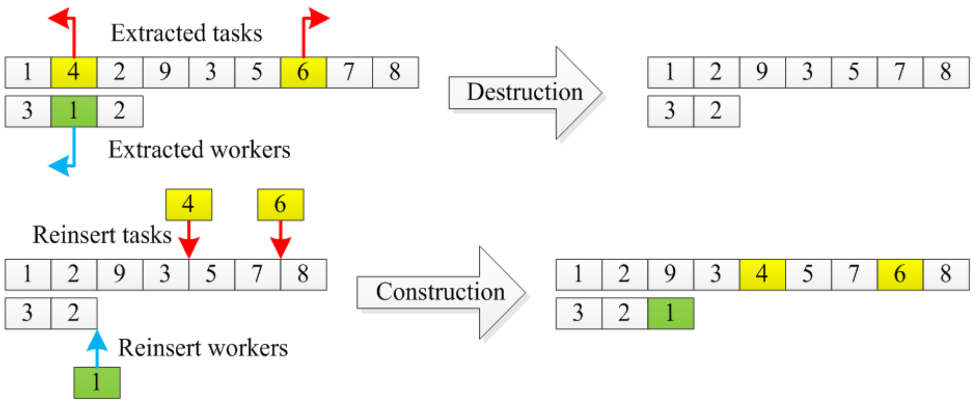

FIGURE 2. Destruction and construction

In Figure 2, tasks 4, 6 and worker 1 are extracted from the original solution in destruction phase. And the remained task assignment and worker allocation are $\{1, 2, 9, 3, 5, 7, 8\}$ and $\{3, 2\}$. Then in construction phase, the extracted tasks 4, 6 are reinserted into the remained task assignment. The new positions of task 4 and 6 are different from their original positions. And the extracted worker 1 is reinserted with the same method. Thus, the new generated task assignment and worker allocation are $\{1, 2, 9, 3, 4, 5, 7, 6, 8\}$ and $\{3, 2, 1\}$.

In the destruction phase, the choice of the d has a great influence in the performance of the proposed algorithm. A small value of d makes it difficult to escape local optima whereas a larger value of d may have the same effect as the randomly regenerated solution. So it is significant to verify the value of d.

3.3. Local search phase.

Since there are two sub-problems: task assignment and worker allocation in the U-shaped assembly worker assignment and balancing problems, two neighbor structures are developed to improve the performance of solution in local search phase. These neighbor structures are depicted in Figure 3.

(1) Local search in task assignment. A neighbor solution is obtained by moving operator. In the moving operator, a task is randomly selected and moved to a different position in the task assignment vector. Note that the moving operator should meet the precedence relation.

(2) Local search in worker allocation. A neighbor solution is generated by swapping operator. In the swapping operator, two different workers are randomly selected and their corresponding workstations are exchanged in the worker allocation vector.

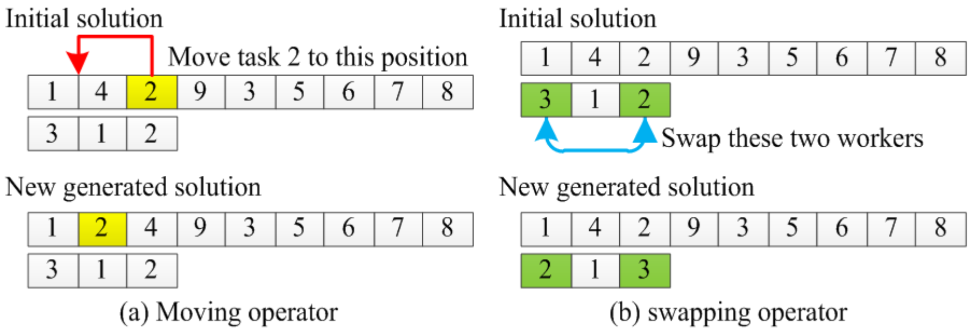

FIGURE 3. Moving and swapping operators

3.4. Acceptance criterion.

After destruction, construction and local search phases, the new solution is checked whether it can replace the incumbent one. Different from the original acceptance criterion, this paper adopts a new acceptance criterion based on the temperature in simulated annealing algorithm. If the new solution π_2 is superior to incumbent one, it replaces the incumbent one. If not, the new solution is accepted with the probability of $e^{(-\Delta/temp)}$, where the constant temperature $temp$ is calculated as Equation (1).

$$Temp = T \times \frac{\sum_{r=1}^{NR} \sum_{i=1}^{NI} t_{ir}}{NR \times NI \times 10} \tag{1}$$

In this equation, the parameter T is another parameter needed to be calibrated.

4. Result Comparison.

Since the concurrent assignment of workers and tasks within the U-shaped assembly lines is a new problem, this paper designs a set of benchmark problems to test the proposed algorithm. These benchmarks include small-scaled problems (Bowman8, Jaeschke9, Jackson11, Mitchell21, Roszieg25 and Gunther35), medium-scaled problems (Hahn53, Tonge70 and Lutz89) and large-scaled problems (Arcus111, Barthol148 and Scholl297). The original data of these problems come from the website: <http://assembly-line-balancing.mansci.de/>. Similar to the approach in [7] the processing times of tasks by worker h are randomly generated between $[t_i \times 0.8, t_i \times 1.2]$, in which t_i is the original processing time. And the algorithm is encoded in C programming language and is run on a computer with Intel(R) Core(TM) i5 CPU, 2.80GHz and 2.00GB RAM.

4.1. **Parameter calibration.** Before the comparative experiments, this paper employs Taguchi method to verify the parameters of the proposed IG. The verified parameters are *the number of extracted tasks* d and *the temperature parameter* T. And the levels are respectively set as: $\{4, 5, 6\}$ and $\{0.3, 0.4, 0.5\}$. After designing the orthogonal array for different levels and setting the average cycle time as the response value we run each experiment for 5 times and obtain the results by Minitab 17. And the results are shown in Figure 4.

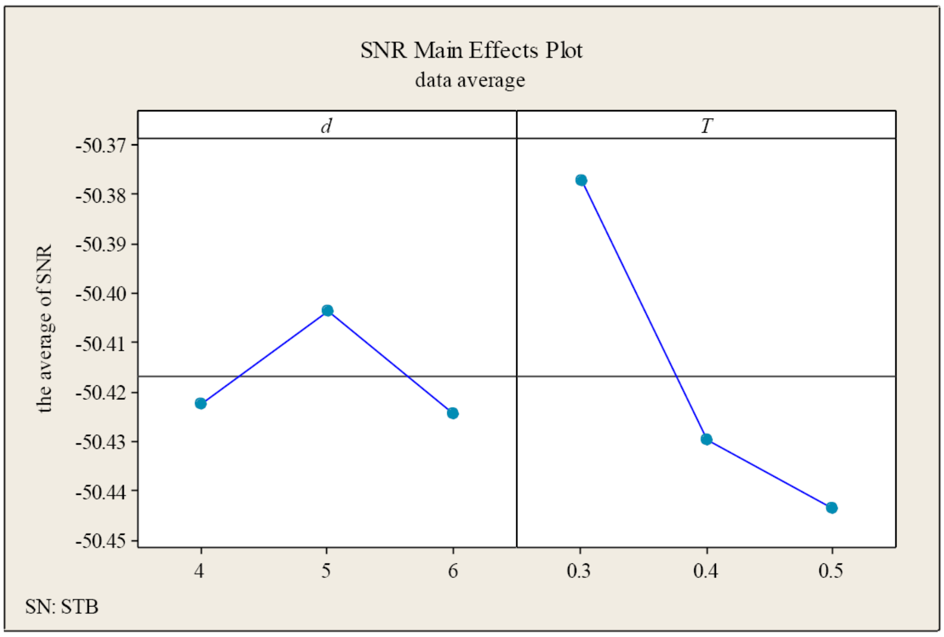

Figure 4. SNR main effects plot

The signal to noise ratio (SNR) is the ratio of the value of objective function to the variance value of objective function. The bigger the SNR is, the greater robustness the parameter combination will have. Thus, the best combination of the parameters is: $\{d = 5, T = 0.3\}$.

4.2. **Convergence analysis.** To test the convergence of the proposed IG GAMS/Cplex 23.8 which can solve problems to optimality is selected. Taking account of computational efforts, several small-scaled instances, Bowman8, Jaeschke9, Jackson11 and Mitchell21 are employed for the comparative analysis. The results are reported in Table 1.

Table 1. The comparative results between the GAMS/Cplex and IG

Case	NW	LB_{CT}	GAMS/Cplex		IG	
			CT	CPU/s	CT	CPU/s
Bowman8	3	139	178	0.477	178	0.64
Jaeschke9	3	155	176	0.378	176	0.81
Jackson11	3	193	229	1.122	229	1.21
Mitchell21	5	165	197	6585.726	204	4.41

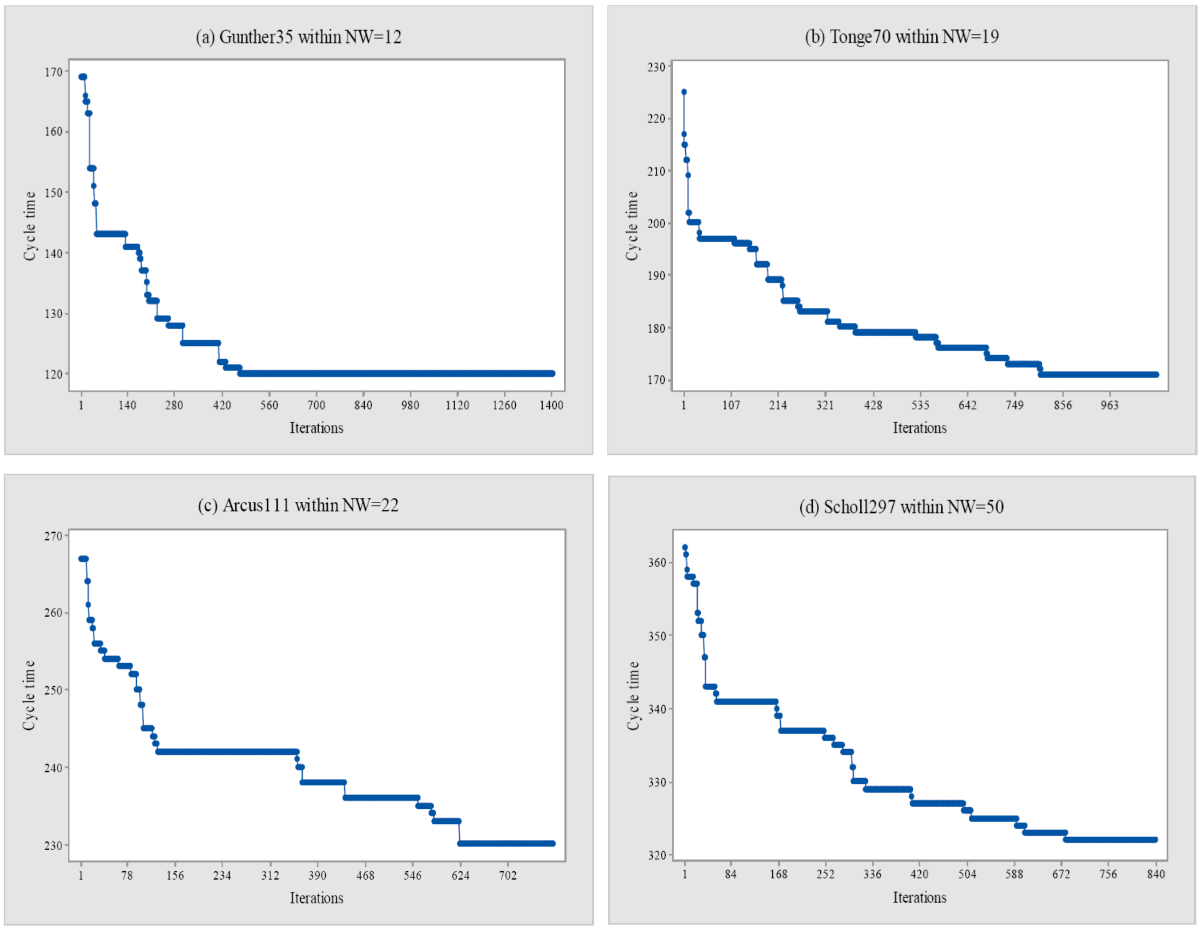

Figure 5. The convergence analysis under different instances

It can be seen from Table 1 that in the instances of Bowman8, Jaeschke9, Jackson11, the proposed IG algorithm obtains the same solutions as those of GAMS/Cplex and the computational times of these two methods are very close. However, for a little bigger instance of Mitchell21, the cycle time obtained by IG is slightly larger whereas the computational time of IG is far less than that of GAMS/Cplex. These results suggest that the IG algorithm obtains a better trade-off between solution quality and computational efforts.

With respect to medium-scaled and large-scaled instances including Gunther35, Tonge70, Arcus111 and Scholl297, IG is solely utilized under the iteration criteria of $N_i \times N_i \times 20$ millisecond and the solution of each generation is reported in Figure 5. It can be found that in small-scaled instance (Gunther35), the cycle time is not updated in the iteration of about 430; in medium-scaled instance (Tonge70), the cycle time is not further optimized in the iteration of about 800; in large-scaled instances (Arcus111 and Scholl297), the cycle time is not decreased in the iteration of about 624 and 672. Thus, it is concluded that the proposed IG algorithm is gradually convergent to the lower bound and there is a significant improvement in convergence with the increase of instance scale.

4.3. **Performance comparison.**

To compare the performance of the proposed IG with other population-based meta-heuristic algorithms, particle swarm optimization (PSO) is selected and verified through the above parameter calibration method. The final population size is set as 60 for 32 instances. Each case is respectively run 10 times under two iteration criteria ($N_i \times N_i \times 10$ millisecond and $N_i \times N_i \times 20$ millisecond) for total 1280 experiments. The computational results are shown in Table 2. And the minimum and average cycle times of PSO and IG are also depicted in Figure 6.

Table 2. The comparative results of PSO and IG under different iterations

				$N_i \times N_i \times 10$ millisecond				$N_i \times N_i \times 20$ millisecond			
No.	Case	*NW*	LB_{CT}	PSO		IG		PSO		IG	
				Min	*Ave*	*Min*	*Ave*	*Min*	*Ave*	*Min*	*Ave*
1	Roszieg25	3	438	475	495.7	**471**	**476.7**	469	490.6	469	**476.6**
2	Roszieg25	4	265	314	327.9	**313**	**317.8**	314	323.9	**312**	**315.8**
3	Roszieg25	6	158	218	224.4	**209**	**211.9**	217	224.4	**204**	**211.2**
4	Roszieg25	9	90	131	133	**120**	**126.3**	130	131.9	**121**	**123.6**
5	Gunther35	4	337	434	452.8	**431**	**439.8**	445	457.6	**416**	**427.7**
6	Gunther35	5	285	373	383.1	**343**	**354.0**	356	375.8	**340**	**347.9**
7	Gunther35	7	181	250	257.3	**223**	**231.9**	250	257.2	**225**	**232.3**
8	Gunther35	12	85	133	135.4	**118**	**124.2**	130	134.7	**117**	**121.2**
9	Hahn53	5	387	550	564.6	**509**	**530.9**	541	563.1	**512**	**528.7**
10	Hahn53	7	238	333	347.9	**311**	**325.9**	343	347.2	**311**	**322.6**
11	Hahn53	10	178	268	270.8	**244**	**248.4**	263	269.3	**240**	**248.1**
12	Hahn53	14	115	187	189.9	**170**	**176.9**	184	190.6	**170**	**174.1**
13	Tonge70	7	355	500	513.9	**468**	**483.5**	502	512.3	**454**	**468.2**
14	Tonge70	10	212	319	323.1	**287**	**298.3**	316	320.5	**278**	**290.0**
15	Tonge70	14	156	250	254.7	**217**	**230.2**	248	252.7	**222**	**228.1**
16	Tonge70	19	109	194	196.3	**175**	**179.9**	190	195.3	**169**	**174.9**
17	Lutz89	8	367	524	533.3	**500**	**509.7**	518	530.3	**476**	**493.8**
18	Lutz89	12	246	414	416.8	**383**	**396.0**	408	415.7	**375**	**385.7**
19	Lutz89	16	173	272	278.6	**258**	**263.8**	270	278.2	**243**	**255.2**
20	Lutz89	21	134	248	250.8	**224**	**234.8**	245	248.0	**215**	**224.5**
21	Arcus111	9	430	662	676.9	**621**	**636.8**	658	672.2	**595**	**625.5**
22	Arcus111	13	248	400	404.4	**369**	**375.3**	391	402.8	**359**	**367.8**
23	Arcus111	17	198	315	319.6	**296**	**302.0**	317	319.9	**287**	**293.6**
24	Arcus111	22	143	252	253.8	**235**	**240.3**	247	253.5	**227**	**233.4**
25	Barthol148	10	491	745	767.8	**693**	**727.2**	745	762.5	**676**	**695.2**
26	Barthol148	14	317	530	538.6	**493**	**511.2**	531	536.3	**489**	**498.4**
27	Barthol148	21	211	380	383.5	**352**	**363.5**	377	381.4	**349**	**355.4**
28	Barthol148	29	145	262	264.7	**249**	**254.4**	262	264.1	**244**	**248.6**
29	Scholl297	19	438	758	769.5	**727**	**746.0**	759	769.0	**723**	**730.2**
30	Scholl297	29	295	541	544.5	**518**	**531.3**	540	542.8	**511**	**520.0**
31	Scholl297	38	222	426	432.1	**416**	**423.7**	428	431.6	**409**	**418.0**
32	Scholl297	50	170	329	331.8	**320**	**327.0**	327	331.2	**314**	**321.7**

From the table, it can be seen that the proposed IG obtains better solutions for all instances under two iteration criteria. Only under Roszieg35 within 3 workstations and iteration criteria of $N_i \times N_i \times 20$ millisecond, the minimum cycle time of PSO is equal to that of IG. In other cases, IG finds smaller solutions in terms of the minimum and average cycle time compared with PSO. And from Figure 6, the cycle time obtained by IG is below that by PSO for all instances. Thus, it can be concluded that when dealing with small-scaled, medium-scaled and large-scaled problems, IG is more efficient and effective than PSO.

5. **Conclusions.**

Due to different skill levels of workers on the U-shaped manual assembly line, the processing time of each task may have multiple choices, and a waste of production times or unexpected production break may arise. To ensure each workstation occupied by a suitable worker and improve the line efficiency, an iterated greedy algorithm is developed in which both workers and tasks are assigned cooncurrently into workstations

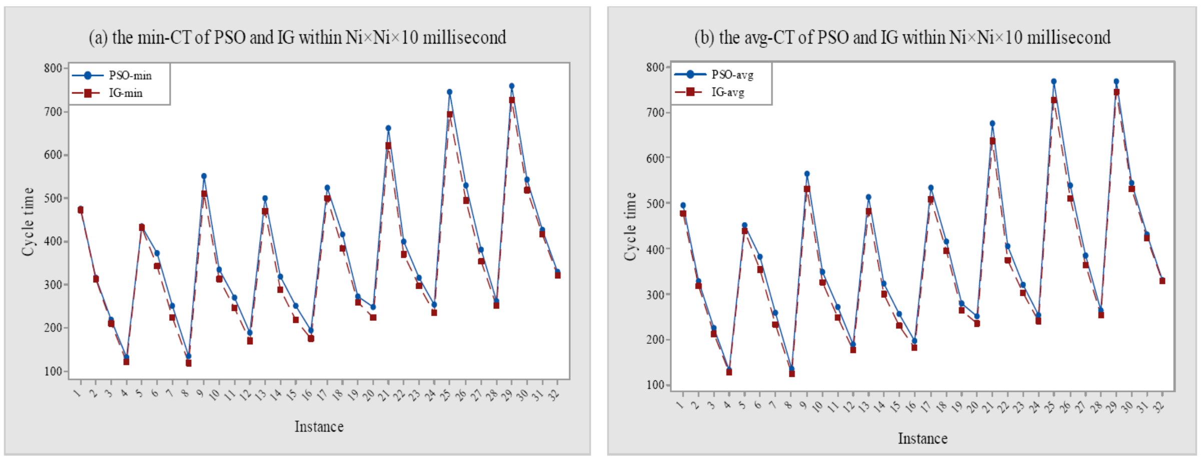

Figure 6. The comparative analysis of PSO and IG for all instances

within the U-shaped assembly lines. In this algorithm, a union of multiple heuristic rules is employed to ensure the diversity of elements in a solution. A specified number of tasks and workers are extracted and reinserted respectively in destruction and construction phases to explore the optimal solutions. The acceptance criterion is improved based on the temperature in simulated annealing algorithm to avoid trapping into local optima. Comparative experiments with GAMS/Cplex demonstrate that the proposed IG obtains a better trade-off between solution quality and computational efforts. And for the medium-scaled and large-scaled instances, the proposed IG shows strong convergence to the lower bound and outperforms PSO in terms of efficiency and effectiveness.

Future research will be extended to address U-shaped assembly line worker assignment and balancing problems in which walk time of workers should be considered. And the collaboration between workers and machines can be extended to further enhance flexibility and productivity.

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] Ö. Mutlu, O. Polat and A. A. Supciller, An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II, *Computers & Operations Research*, vol.40, no.1, pp.418-426, 2013.
[2] A. A. Chaves, C. Miralles and L. A. N. Lorena, Clustering search approach for the assembly line worker assignment and balancing problem, in *Book Clustering Search Approach for the Assembly Line Worker Assignment and Balancing Problem*, 2007.
[3] P. T. Zacharia and A. C. Nearchou, A population-based algorithm for the bi-objective assembly line worker assignment and balancing problem, *Engineering Applications of Artificial Intelligence*, vol.49, pp.1-9, 2016.
[4] M. Vilą and J. Pereira, A branch-and-bound algorithm for assembly line worker assignment and balancing problems, *Computers & Operations Research*, vol.44, pp.105-114, 2014.
[5] L. Borba and M. Ritt, A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem, *Computers & Operations Research*, vol.45, pp.87-96, 2014.
[6] Q. K. Pan and R. Ruiz, An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem, *Omega*, vol.44, no.2, pp.41-50, 2014.
[7] 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.