Int J Adv Manuf Technol (2011) 54:271–285
DOI 10.1007/s00170-010-2937-3

ORIGINAL ARTICLE

Balancing of stochastic U-type assembly lines: an imperialist competitive algorithm

M. Bagher · M. Zandieh · H. Farsijani

Received: 26 December 2009 / Accepted: 17 June 2010 / Published online: 27 October 2010

Abstract Recently, there is a growing interest in the industry to replace traditional straight assembly lines with U-shaped lines for more flexibility and higher productivity. Due to mathematical and computational complexity, assembly line balancing problems are known to be NP hard in nature. Therefore, many meta-heuristics have been proposed to find optimal solution for these problems. This paper presents a new hybrid evolutionary algorithm to solve stochastic U-type assembly line balancing problems, with the aim of minimizing the number of work stations, idle time at each station, and non-completion probabilities of each station (probability of the station time exceeding cycle time). The proposed algorithm is a combination of computer method for sequencing operations for assembly lines (COMSOAL), task assignment rules heuristic, and newly introduced imperialist competitive algorithm (ICA). Unlike the current evolutionary algorithms that are computer simulation of natural processes, ICA is inspired from socio-political evolution processes. Since appropriate design of the parameters has a significant impact on the algorithm efficiency, various parameters of the ICA are tuned by means of the Taguchi method. For the evaluation of the proposed hybrid algorithm, the performance of the proposed method is examined over benchmarks from the literature and compared with the best algorithm proposed before. Computational results demonstrate the efficiency and robustness of our algorithm.

M. Bagher · M. Zandieh (✉) · H. Farsijani
Department of Industrial Management,
Management and Accounting Faculty,
Shahid Beheshti University,
GC, Tehran, Iran
e-mail: m_zandieh@sbu.ac.ir

Keywords Stochastic U-type assembly line balancing problems · Imperialist competitive algorithm · Taguchi method

1 Introduction

Assembly lines have gained great importance in manufacturing of high quantity standardized products particularly automobile manufacturing. Also, there are other industries concerned with the practical applications of assembly lines such as electronic industry [1], production of white goods [2], appliance industry [3], lamp production [4], etc.

An assembly line is an ordered sequence of *work stations* (k=1, 2,..., M) arranged along a conveyer belt or a similar material handling system [5]. The components (work pieces) consecutively enter the assembly line and move from one station to the next station until they reach the end of the line. At each station, a predefined set of tasks is performed on each component within a specified fixed time called as the *cycle time* (CT). Cycle time is the time interval between two successive completed products. Each task i (i=1, 2,..., N) has a *processing time* t_i. Processing times are assumed to be independent random variables with means μ_i and variances σ_i. Task allocations are to be in such a manner to satisfy the precedence relationships among tasks while ensuring the probability of completing tasks within a specified cycle time, CT, is greater or equal to a value, $1-\alpha$ [6].

Arrangement of stations in assembly lines can be classified in two general groups as traditional (straight) and U-shaped assembly lines. The main differences between them is that operators have access to both front side and back side of the line in U-type assembly lines, on the contrary, in straight assembly lines operators work on a length of the line. Figure 1 provides a comparison between a straight line and a U line.

In addition, straight assembly lines allow assignments of tasks whose predecessors are already assigned to stations, but U-type assembly lines allow assignments in both forward and backward directions. Therefore, the number of stations needed for U-type line layout is *never* more than the number of stations needed for traditional straight line environment [7]. Consider, for example, well-known Mertens problem with seven tasks. The precedence graph is shown in Fig. 2. In traditional assembly lines, task 5 cannot be assigned to a station earlier than tasks 1 and 2. However, in U-type lines, task 5 can be assigned to a station if *either* task set 1 and 2 *or* task 6 has already been assigned.

In recent years, many manufacturers have adopted a just-in-time (JIT) approach to manufacturing, for it improves their productivity, profits, and product quality. One of the important changes resulting from JIT implementation is the replacement of the traditional lines with U-shaped production lines [8]. The main benefits of the U line in comparison with straight line include reduction in the wasted movement of operators and work-in-process inventory, improved productivity [9], easier implementation of zero-defects campaign, higher flexibility in workforce planning in the face of changing demand [10], and improvement in material handling [11].

Assembly line balancing (ALB) is how to allocate a certain set of tasks to an ordered sequence of stations, with respect to some objectives, such as minimizing the number of stations for a given cycle time (type I), minimizing the cycle time for a given number of stations (type II), and maximizing efficiency of the assembly line (type E) [12]. Main constraints of ALB are as follows:

- Each task must be assigned to exactly one station.
- The cycle time is greater than or equal to the total time of the tasks assigned to each station. In other words, the station time should not exceed the cycle time.
- Precedence relationship must be considered.

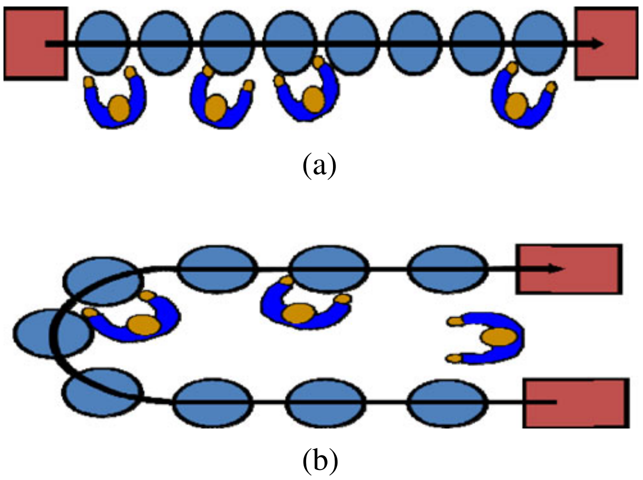


Fig. 1 **a** Straight assembly line and **b** U-shaped assembly line

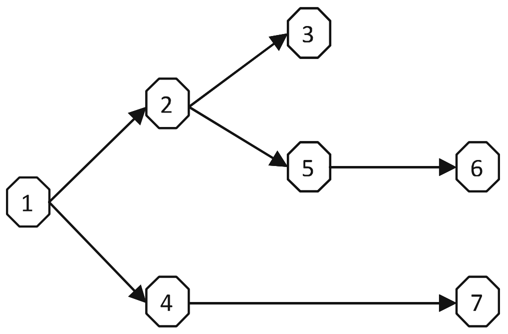


Fig. 2 Precedence graph for Mertens' seven problems

This paper presents a new evolutionary algorithm to solve stochastic, U-type assembly line balancing problems (UALBP) with the aim of minimizing the number of work stations, idle time at each station, and the non-completion probabilities of each station.

The paper has the following structure. Section 2 gives literature review of single-model UALBP. Section 3 is problem description. Section 4 introduces the proposed algorithm. In Section 5, data gathering and calibration of the parameters of the proposed algorithm are discussed. Section 6 presents experimental design in which the results achieved by proposed algorithm compared with those achieved by genetic algorithm (GA) presented by Baykasoğlu and Özbakir [7]. Finally, conclusion and future works are given in Section 7.

2 Literature review

The assembly line balancing problem has been investigated extensively since 1950. Although there are many researches on straight assembly line balancing problems (SALBP), there are still insufficient studies on U-type lines. A detailed review of studies about SALBP problem can be seen in Refs. [13–16]. In this section, we concentrate on literature review of UALBP and stochastic UALBP.

2.1 U-type assembly line balancing

The UALBP was first studied by Baykasoğlu and Özbakir [17]. They proposed a dynamic programming formulation to solve small problems up to 11 tasks. Miltenburg and Sparling [18] developed three exact algorithms for the UALBP including a reaching dynamic programming algorithm, and breadth- and depth-first branch-and-bound algorithms. In another study, Urban [19] proposed integer linear programming to formulate UALBP with deterministic task times. Scholl and Klein [12] developed a branch-and-bound procedure to balance U-shaped JIT assembly lines. Later, Ajenblit and Wainwright [20] proposed a GA, and Erel et al. [21] developed a simulated annealing approach for solving UALBP. Miltenburg [22] and Sparling [23] studied balancing multiple U lines. Kim et al. [24], Sparling and Miltenburg [25], Visich et al. [26], and Kara et al. [27, 28] studied the mixed-model version of UALBP. Nakade and Ohno [29] analyzed the optimal worker allocation problem for a U line. Miltenburg [30] investigated the effect of "U-shape of the line" on the line's effectiveness considering the breakdowns. Miltenburg [31] studied a one-piece flow production system on U lines and examines the related literature. Nakade and Ohno [32] considered two type allocations of workers in a U-shaped production line. Aase e al. [33] analyzed the exact U-shaped line balancing procedures. Later, the same author [34] studied U-shaped assembly line layouts and their impact on labor productivity. Martinez and Duff [35] proposed heuristic approaches and GA to solve UALBP. Gökçen et al. [36] developed a shortest route formulation to solve UALBP. Gökçen and Ağpak [37] proposed a goal programming model for UALBP. Baykasoğlu [38] developed a multi-rule multi-objective simulated annealing algorithm for the SALBP and UALBP. Hwang [39] developed a multi-objective GA to solve the SALBP and UALBP. Boysen et al. [5] and Becker and Scholl [40] proposed a review of exact and heuristic procedure for solving UALBP. Toklu and Özcan [41] developed a fuzzy goal programming model for UALBP problem with multiple objectives. Boysen and Fliedner [42] proposed a shortest-path algorithm to solve SALBP and UALBP using ant colony optimization. Toksari et al. [43] studied simple and U-type assembly line balancing problems with a learning effect. Özcan and Toklu [44] proposed new hybrid improvement heuristic approach to solve SALBP and UALBP based on simulated annealing. Sabuncuoglu et al. [45] examined ant colony optimization to solve the single-model UALBP. Baykasoğlu and Dereli [46] proposed an ant colony-based algorithm to solve simple and U-type assembly line balancing problems. Hwang and Katayama [47] proposed a multi-decision genetic approach to deal with workload balancing problems in mixed-model U-shaped lines. Simaria et al. [48] introduced flexible U-shaped assembly lines and developed a procedure based on ant colony algorithms to solve the problem. Zhang et al. [49] proposed a heuristic approach for fuzzy U-shaped line balancing problem.

2.2 Stochastic U-type assembly line balancing

Most of the existing papers on UALBP focus on deterministic task times, and very little has been done concerning the stochastic U-line balancing problem.

Chand and Zeng [50] examined the difference between traditional assembly lines and U lines under stochastic task times. Ağpak et al. [51] developed a heuristic for stochastic UALBP. Chiang and Urban [52] proposed a hybrid heuristic for the stochastic UALBP.

Guerriero and Miltenburg [53] presented a recursive algorithm to find the optimal solution for small size problems up to 25 or fewer tasks. Erel et al. [54] proposed beam search-based method for the stochastic assembly line balancing problem in U lines. Urban and Chiang [6] proposed an optimal piecewise-linear program for UALBP with stochastic task times. Chiang and Urban [8] presented a hybrid heuristic consisting of an initial feasible solution module and a solution improvement module. Baykasoğlu and Özbakir [7] introduced a hybrid method that integrated COMSOAL method, task assignment heuristics, and GA to solve stochastic UALBP.

In this study, we propose a new evolutionary algorithm to solve stochastic, UALBP, with the aim of minimizing the number of work stations, idle time at each station, and non-completion probabilities of each station. To test the performance of the proposed approach, we have compared the results of the proposed approach with GA proposed [7].

3 Problem description

This paper presents a new evolutionary algorithm to solve stochastic UALBP, with the aim of minimizing the number of work stations, idle time at each station, and non-completion probabilities of each station. The proposed algorithm is a combination of COMSOAL [55] method, task assignment rules heuristic, and imperialist competitive algorithm (ICA).

The general assumptions of the considered problems are as follows:

1. Each task can be assigned to only one work station.
2. Each task time is assumed to be independent random variable with normal distribution (mean μ_i and standard deviation σ_i).
3. The precedence relationship among tasks is known.
4. Parallel stations and work-in-process buffer are not allowed.
5. The travel times of workers between stations are ignored.

Let i be the index number of tasks, each with given precedence relationship (i=1, 2,..., N). The tasks are to be assigned to stations in such a manner to minimize the number of stations while ensuring that the probability of completing the tasks within a given cycle time, CT, is greater than or equal to a value ($1-\alpha$) [6]. We have the following definitions:

Table 1 Task assignment rules

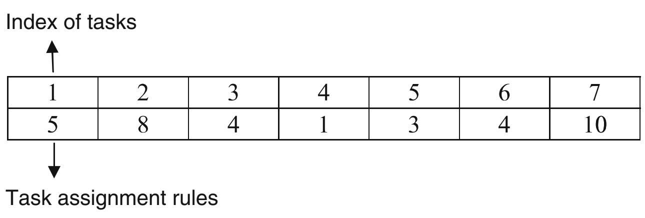
Rule number	Task assignment rules
1	Shortest processing time
2	Longest processing time
3	Minimum total number of successor tasks
4	Maximum total number of successor tasks
5	Maximum total time of successor tasks
6	Minimum total time of successor tasks
7	Maximum total number of predecessor tasks
8	Minimum total number of predecessor tasks
9	Maximum total time of predecessor tasks
10	Minimum total time of predecessor tasks

CT	Cycle time
t_i	Processing time of task i
μ_i	Mean process time of task i
σ_i	Standard deviation of processing time of task i
π_k	Non-completion probability of station k
NE_j	Number of station in jth solution
WL_k	Work load of kth station (set of tasks assigned to the station k)
Z_k	Random variable with mean 0 and standard deviation 1
$F(Z_k)$	Cumulative density function of Z_k
ts_k	Total time of tasks assigned to station k.
$1-\alpha$	Upper bound for non-completion probability
$K_{1-\alpha}$	$(1-a)$ quantile of a cumulative probability normal distribution function
LB	Lower bound for the minimum number of stations. Urban and Chiang [6] have proposed a theoretical lower bound for the minimum number of stations. The proposed lower bound is given in Eq. 1:

$$\mathrm{LB} = \left[\frac{1}{\mathrm{CT}} \left(\sum_{i=1}^{N} \mu_i + K_{(1-\alpha)} \sqrt{\sum_{i=1}^{N} \sigma_i^2} \right) \right] \tag{1}$$

The proposed algorithm goes through following steps:

Step 1 Generate initial solution using imperialist competitive algorithm. Each value of the solution represents one assignment rule (Table 1).



1	2	3	4	5	6	7
5	8	4	1	3	4	10

Fig. 3 Country structure of proposed ICA for Mertens' seven problems

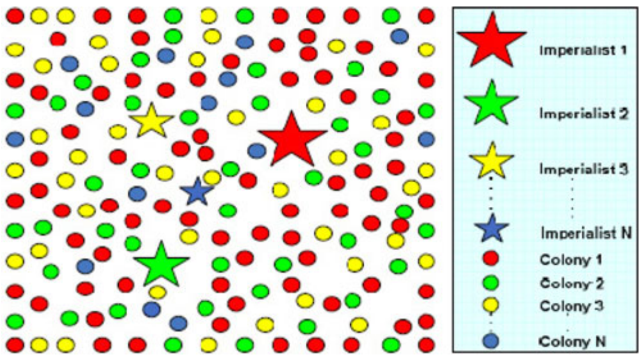


Fig. 4 Generating the initial empires: The more colonies an imperialist possess, the bigger is its relevant *star mark*

Step 2 Form the set of assignable tasks (tasks whose successors and predecessors have already been assigned).

Step 3 Select appropriate task from the set of assignable tasks by considering the assignment rules and cycle time.

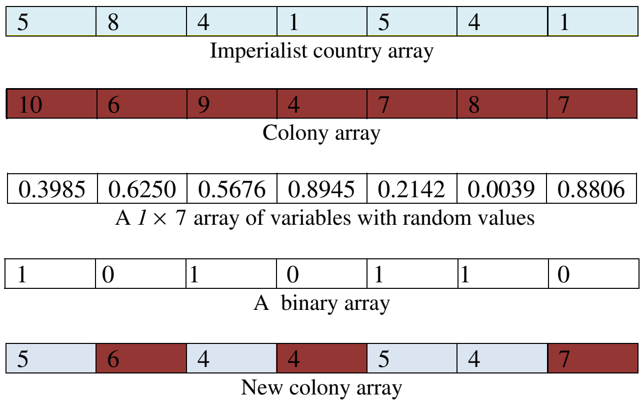
Step 4 Calculate non-completion probability of station k from Eq. 2 :

$$\pi_k = 1 - F(z_k) \tag{2}$$

where $F(z_k)$ is the cumulative density function of z_k with normal distribution and calculated using Eq. 3:

$$z_k = \frac{\left(\mathrm{CT} - \sum_{i \in \mathrm{WL}_k} \mu_i\right)}{\sqrt{\sum_{i \in \mathrm{WL}_k} \sigma_i^2}} \tag{3}$$

Step 5 If the non-completion probability is less than predefined upper bound of non-completion probability, the selected task can be assigned to the



5	8	4	1	5	4	1

Imperialist country array

10	6	9	4	7	8	7

Colony array

0.3985	0.6250	0.5676	0.8945	0.2142	0.0039	0.8806

A *1* × 7 array of variables with random values

1	0	1	0	1	1	0

A binary array

5	6	4	4	5	4	7

New colony array

Fig. 5 The mechanism of assimilation

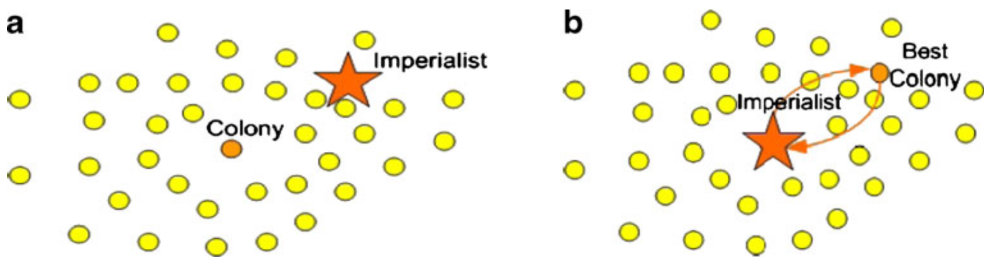


Fig. 6 **a** Exchanging the positions of a colony and the imperialist. **b** The entire empire after position exchange

station; otherwise, open a new station and assign the task to the station.

Step 6 Update the set of assignable tasks.

Step 7 Continue steps 2 to 5 until all tasks are assigned to workstations.

4 The proposed imperialist competitive algorithm

4.1 ICA in general

Different methods have been proposed to solve an optimization problem. Unlike the current evolutionary algorithms, such as GA and simulated annealing (SA), which are a simulation of natural processes such as natural evolution, ICA uses socio-political evolution processes, as source of inspiration. Similar to the other evolutionary algorithms, this algorithm begins with an initial population. Each individual of the population is called a *country*. Some of the best countries (in optimization terminology, countries with the least cost) are selected to be the imperialist states and the others are considered to be colonies of these imperialists. All the colonies of initial countries are divided among the mentioned imperialists based on their power. A set of one imperialist and its colonies is called an *empire*.

After forming initial empires, their colonies start moving toward their relevant imperialist country. This movement is a simple model of assimilation policy, which was pursued by some of the imperialist states and results to improvements in socio-political characteristics of colonies such as *culture*, *language*, and *economical policy*. The total power of an empire depends on both the power of the imperialist

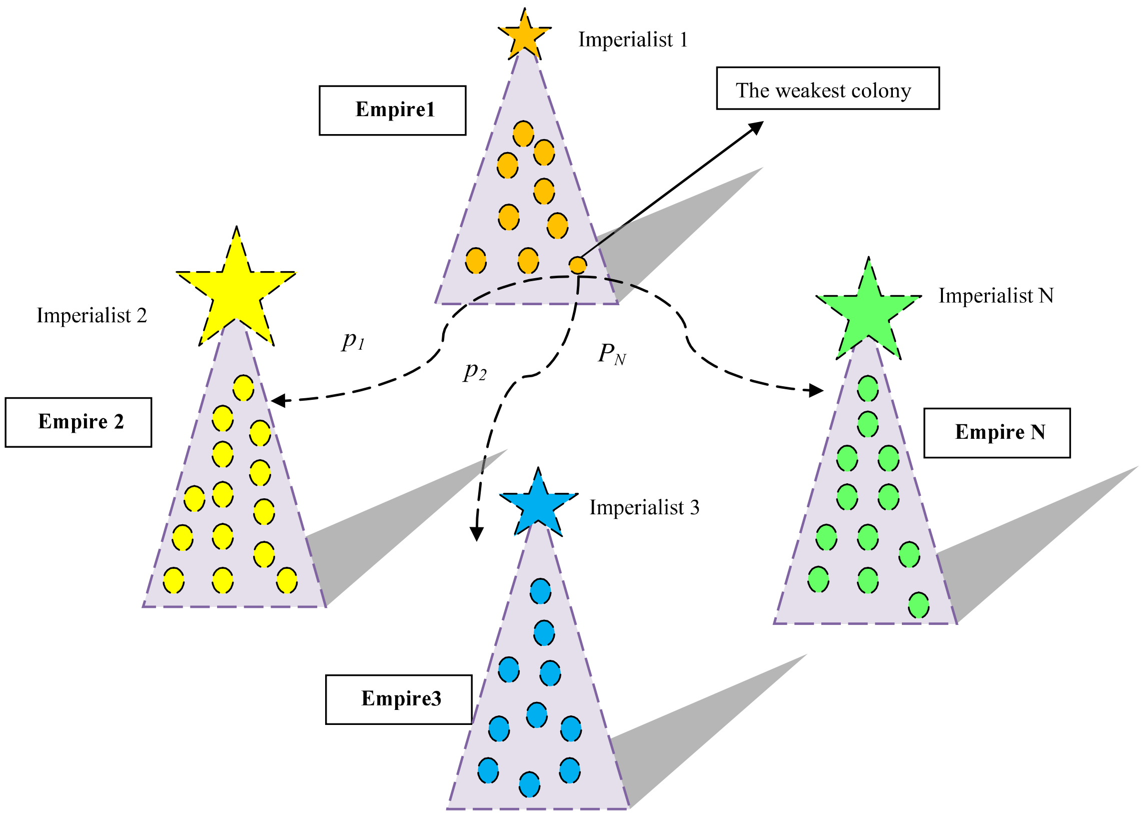


Fig. 7 Imperialistic competition: The more powerful an empire is, the more likely it will possess the weakest colony of the weakest empire

Table 2 Factors and factor levels

Factor	Symbol	Levels	Small	Medium	Large
Assimilation rate (AR)	A	3	A(1), 0.10	A(1), 0.01	A(1), 0.03
			A(2), 0.30	A(2), 0.05	A(2), 0.05
			A(3), 0.80	A(3), 0.10	A(3), 0.10
Revolution rate (RR)	B	3	B(1), 0.10	B(1), 0.05	B(1), 0.10
			B(2), 0.30	B(2), 0.10	B(2), 0.30
			B(3), 0.40	B(3), 0.30	B(3), 0.40
ξ	C	3	C(1), 0.01	C(1), 0.01	C(1), 0.001
			C(2), 0.03	C(2), 0.05	C(2), 0.01
			C(3), 0.05	C(3), 0.07	C(3), 0.05
(N_{country}, N_{imp})	D	3	D(1), (50, 2)		
			D(2), (75, 3)		
			D(3), (100, 4)		

country plus a percentage of mean power of its colonies [56].

Then, the imperialistic competition starts among all the empires. In this competition, imperialist countries should increase their power. Any empire that is not able to succeed in this competition and cannot increase its power (or at least prevent decreasing its power) will be eliminated from the competition. The imperialistic competition will gradually result in an increase in the power of powerful empires and a decrease in the power of weaker ones. Weak empires will lose their power and finally they will collapse. The movement of colonies toward their relevant imperialist states along with competition among empires and also the collapse mechanism will cause all the countries to converge to a state in which there exist just one powerful empire in the world and all the other countries are colonies of that empire. Application of the proposed algorithm in UALBP is discussed through the steps discussed below.

4.2 The proposed ICA

4.2.1 Generating initial empires

Each solution in ICA is in a form of an array. Each array consists of variable values to be optimized. In GA terminology, this array is called "chromosome," but here, we use the term "country" for this array. In an N-dimensional optimization problem, a country is a $1\times N$ array. This array is defined by:

$$\text{Country} = [P_1, P_2, P_3, ..., P_N] \tag{4}$$

where P_i 's are the variables to be optimized. Each variable in a country represents a socio-political characteristic of a country. From this point of view, all the algorithm does is to search for the best country that is the country with the best combination of socio-political characteristics such as culture, language, and economical policy [56].

In our problem, each solution (country) is a $1\times N$ array of integer variables that N represents the total number of tasks to be assigned to workstations. The structure of one solution for a seven-task problem is shown in Fig. 3.

Each variable (P_i) represents a task assignment rule and varies between 1 and total number of assignment rules. In this work, assignment rules proposed by Baykasoğlu and Özbakir [7] are adapted. Table 1 represents the list of assignment rules used in this paper.

In the first step of algorithm, initial countries of size N_{country} are randomly produced. Then, the cost of each country must be evaluated. As each country is not a representation of final solution, extra steps are needed to deduct the assembly line configuration from one country. After the deduction procedure, we can compute the cost of each final solution. The following procedure is proposed by Baykasoğlu and Özbakir [7]:

Table 3 Orthogonal array L_9

Trial	Control factors			
	A	B	C	D
1	A(1)	B(1)	C(1)	D(1)
2	A(1)	B(2)	C(3)	D(2)
3	A(1)	B(3)	C(2)	D(3)
4	A(2)	B(1)	C(3)	D(3)
5	A(2)	B(2)	C(2)	D(1)
6	A(2)	B(3)	C(1)	D(2)
7	A(3)	B(1)	C(2)	D(2)
8	A(3)	B(2)	C(1)	D(3)
9	A(3)	B(3)	C(3)	D(1)

Table 4 Optimal level of factors

Factor	Optimal level
A	Small 0.30, medium 0.05, and large 0.05
B	Small 0.30, medium 0.10, and large 0.30
C	Small 0.03, medium 0.05, and large 0.01
D	(75, 3)

```
1. Initialization
   (a) Form the set of assignable tasks (set of tasks can be assigned to opened station with the consideration of cycle time and precedence relations between tasks).
   (b) Let $k$ be the number of workstations and is set to 1.
2. For $i$ from 1 to $N$
   (a) Select appropriate task from assignable tasks according to the task assignment rule in the $i$th position of variables in the array.
   (b) Calculate non-completion probability of $k$th station.
   (c) If non-completion probability $k$th station<predefined upper bound of $\pi_k$;
      i. Assign the selected task to $k$th work station;else
      ii. Open a new station.
      iii. Assign the task to the new station $k=k+1$;end If
   (d) Determine the remaining time of $k$th work station.
   (e) Update assignable tasks according to the remaining time;end for
3. Compute the cost of the final solution from Eq. 5.
4. Display $k$.
```

In order to evaluate the cost of countries, the objective function proposed by Bautista et al. [57] and extended by Baykasoğlu and Özbakir [7] is adapted. The objective function aims to minimize total number of work stations, idle time at each station, and non-completion probabilities of each station simultaneously. Equation 5 represents the objective function:

$$f_j = \left(\mathrm{NE}_j - \left[\frac{\sum_{i=1}^{N} t_i}{\mathrm{CT}}\right]\right) + \left(\frac{\sqrt{\sum_{k=1}^{\mathrm{NE}_j}(\mathrm{ts}_k - \mathrm{CT})^2}}{\mathrm{CT}\sqrt{\mathrm{NE}_j}}\right) + \left(\sum_{k=1}^{\mathrm{NE}_j}\pi_k\right) \tag{5}$$

After computing the cost of each country, the empires are formed by N_{imp} of most powerful countries (countries with the least cost). The remaining N_{col} of the initial countries will be the colonies each of which belongs to empires. To form the initial empires, the colonies are divided among imperialists based on their power. To proportionally divide the colonies among imperialists, the normalized cost of an imperialist is defined by:

$$F_n = \max_{j \in \mathrm{imperialist}} \{f_j\} - f_n \tag{6}$$

where f_n is the cost of the nth imperialist and F_n is its normalized cost. The normalized power of each imperialist is defined by:

$$P_n = \left[\frac{F_n}{\sum_{i=1}^{N_{\mathrm{imp}}} F_i}\right] \tag{7}$$

The initial colonies are divided among empires based on their power. Then, the initial number of colonies of the nth empire will be:

$$\mathrm{NC}_n = \mathrm{round}(P_n \times N_{\mathrm{col}}) \tag{8}$$

To divide the colonies, NC_n of the colonies are randomly chosen and given to the nth imperialist, so the nth empire will be formed. Figure 4 shows the initial empires. As shown in Fig. 4, bigger empires have greater number of colonies in comparison to weaker ones. In Fig. 4, imperialist

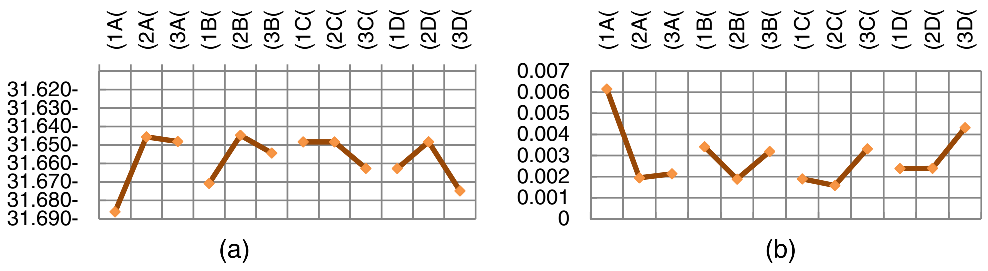


Fig. 8 **a** The mean of *S*/*N* ratio and **b** RPD plot for each level of the factors (small-sized problem)

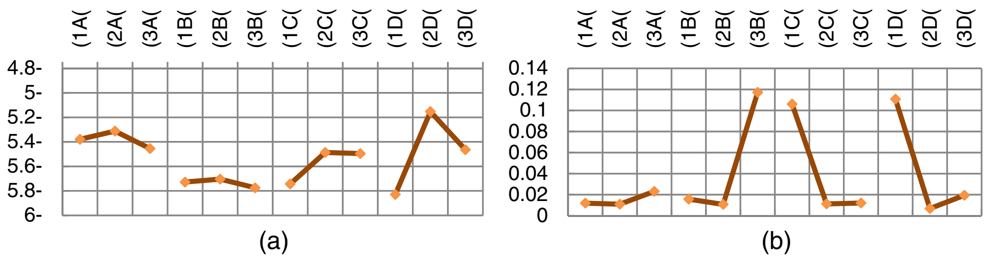


Fig. 9 **a** The mean of *S*/*N* ratio and **b** RPD plot for each level of the factors (medium-sized problem)

1 has formed the most powerful empire and consequently has the greatest number of colonies.

4.2.2 Movement of an empire's colonies toward the imperialist

In assimilation phase, the imperialist states try to absorb their colonies and influence on their socio-political characteristics such as culture and language. In other words, imperialist states change socio-political characteristics of colonies in such a way that they become similar to them (increase their power). The mechanism of assimilation is defined through following steps and shown in Fig. 5:

1. Set assimilation rate (AR). AR represents the percentage of similarity between the imperialist and their colonies. In the example shown in Fig. 5, AR is set to 0.6.
2. Consider the $1 \times N$ array of a colony and an imperialist.
3. Generate a $1 \times N$ array of random variables with uniform distribution [0, 1].
4. Compare each value of random array to AR. If the value is higher than AR, change the value to 0, otherwise change it to 1. Now, we have an array with values 0 and 1 (binary array).
5. Find the positions in the colony array that are equivalent to the positions of value 1 in the binary array.
6. Replace the value of these positions with the value of same positions of imperialist array. In this way, the structure of new colonies will become more similar to the imperialist.

4.2.3 Revolution

After assimilation procedure, in each iteration, sudden changes may happen to the structure empires with a predefined revolution rate (RR). In revolution procedure, new countries are generated and some of the weakest colonies are replaced with these countries. The replacement rate is named as revolution rate.

4.2.4 Exchanging positions of the imperialist and a colony

While moving toward the imperialist, a colony might reach to a position with lower cost than that of imperialist. In this case, the imperialist and the colony change their positions. Then, the algorithm will continue by the imperialist in the new position and then colonies start moving toward this position. Figure 6a shows the position exchange between a colony and the imperialist. In Fig. 6a, the best colony of the empire is shown in a darker color. This colony has a lower cost than that of imperialist. Figure 6b shows the whole empire after exchanging the position of the imperialist and that colony.

4.2.5 Total power of an empire

Total power of an empire is mainly affected by the power of imperialist country. But, power of the colonies of an empire has an effect, albeit negligible, on the total power of that empire. This fact is modeled by defining the total cost by:

$$\mathrm{TF}_n = f(\mathrm{imperialist}_n) + \xi \times \mathrm{mean}\{f(\mathrm{colonies\ of\ empire}_n)\} \tag{9}$$

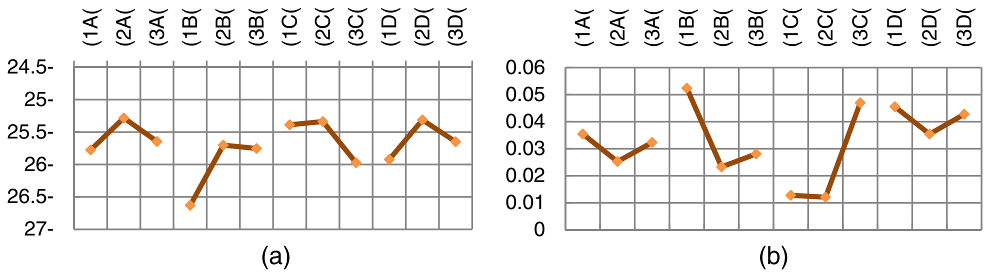


Fig. 10 **a** The mean of *S*/*N* ratio and **b** RPD plot for each level of the factors (large-sized problem)

Table 5 ANOVA table for the *S/N* ratio (Small size problem)

SV	*df*	SS	MS	*F*	Percent of *X*	Cumulative	*P* value
AR	2	0.00045	0.00022	6.29	50.6	50.6	0.024
RR	2	0.00012	0.00006	1.63	6.0	56.6	>0.10
$(N_{\text{country}}, N_{\text{imp}})$	2	0.00011	0.00005	1.54	5.1	61.7	>0.10
ξ	2	0.00007	0.000036				
Total	8	0.00074					
Error	2	0.00007	0.000036				

where TF_n is the total cost of the *n*th empire and ξ is a positive number which is considered to be less than 1. A little value for ξ causes the total power of the empire to be determined by just the imperialist and increasing it will increase the effects of the colonies in determining the total power of an empire [56].

4.2.6 Imperialistic competition

All empires try to take the possession of colonies of other empires and control them. The imperialistic competition gradually brings about a decrease in the power of weaker empires and an increase in the power of more powerful ones. The imperialistic competition is modeled by just selecting some (usually one) of the weakest colonies of the weakest empires and making a competition among all empires to possess these (this) colonies. Figure 7 shows a big picture of the modeled imperialistic competition.

In this competition, each of empires will have a likelihood of taking control of the mentioned colonies based on their total power. In other words, these colonies will not be possessed by the most powerful empires, but these empires will be more likely to possess them. To start the competition, first, the possession probability of each empire which is based on its total power is found. The normalized total cost is simply obtained by:

$$\text{NTF}_n = \max_i\{\text{TF}_i\} - \text{TF}_n \tag{10}$$

where TF_n and NTF_n are the total cost and the normalized total cost of *n*th empire, respectively. Having the normalized total cost, the possession probability of each empire is given by:

$$P_{P_n} = \left[\frac{\text{NTF}_n}{\sum_{i=1}^{N_{\text{imp}}} \text{NTF}_i} \right] \tag{11}$$

To divide the mentioned colonies among empires based on the possession probability of them, vector **P** is formed as:

$$P = [P_{p_1}, P_{p_2}, P_{p_3}, \ldots, P_{p_{N_{\text{imp}}}}] \tag{12}$$

Then, a vector with the same size as **P** whose elements are uniformly distributed random numbers is created:

$$\begin{aligned} &R = [r_1, r_2, r_3, \ldots, r_{N_{\text{imp}}}] \\ &r_1, r_2, r_3, \ldots, r_{N_{\text{imp}}} \sim U(0,1) \end{aligned} \tag{13}$$

Then, vector **D** is formed by subtracting **R** from **P**:

$$D = P - R = [D_1, D_2, D_3, \ldots, D_{N_{\text{imp}}}] \tag{14}$$

Referring to vector **D** the mentioned colony (colonies) is handled to an empire whose relevant index in **D** is maximum. The procedure of selecting an empire is similar to the roulette wheel procedure in GA. But, this method of selection is much faster than the conventional roulette wheel because there is no need to calculate the cumulative distribution function and the selection is just based on the values of probabilities.

Table 6 ANOVA table for the *S/N* ratio (medium-sized problem)

SV	*df*	SS	MS	Percent of X	Cumulative	*P* value
AR	2	0.07630	0.03815	34.3	34.3	0.013
RR	2	0.07535	0.03767	33.9	68.2	0.014
$(N_{\text{country}}, N_{\text{imp}})$	2	0.03366	0.01683	12.5	80.7	0.08
ξ	2	0.00941	0.00471			
Total	8	0.19472				
Error	2	0.00941	0.00471			

Table 7 ANOVA table for the *S/N* ratio (large-sized problem)

SV	*df*	SS	MS	*F*	Percent of *X*	Cumulative	*P* value
AR	2	0.25423	0.12711	5.49	34.3	34.3	0.034
RR	2	0.22682	0.11341	4.89	30.0	64.3	0.043
(N_{country}, N_{imp})	2	0.07422	0.03711	1.60	4.6	68.9	>0.10
ξ	2	0.04635	0.02317				
Total	8	0.60161					
Error	2	0.04635	0.02317				

4.2.7 Eliminating the powerless empires

Powerless empires gradually collapse in the imperialistic competition, and their colonies will be divided among other empires. To model the collapse mechanism, different criteria can be defined to consider an empire powerless. In the implementations of this paper, an empire is assumed to be collapsed when it loses all of its colonies.

Table 8 Comparison of ICA and GA solution (high variance)

Problem	Tasks	CT	$K_{(1-\alpha)}=1.96$			$K_{(1-\alpha)}=1.645$			$K_{(1-\alpha)}=1.28$		
			LB	GA	ICA	LB	GA	ICA	LB	GA	ICA
Merten	7	8	NFS	NFS	NFS	NFS	NFS	NFS	5	5	5
		10	4	5	5	4	5	5	4	5	5
		15	3	3	3	3	3	3	3	3	3
		18	2	3	3	2	3	3	2	2	2
Bowman	8	20	5	6	6	5	6	6	5	6	6
Jaeschke	9	8	6	7	7	6	7	7	6	7	7
		10	5	7	7	5	7	7	5	7	7
		18	3	3	3	3	3	3	3	3	3
Jackson	11	10	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS	NFS
		13	5	5	5	5	5	5	4	5	5
		14	4	5	5	4	5	5	4	5	5
		21	3	3	3	3	3	3	3	3	3
Mitchell	21	21	6	8	8	6	7	7	6	7	7
		26	5	6	6	5	6	6	5	5	5
		35	4	4	4	4	4	4	4	4	4
		41	4	4	4	4	4	4	3	4	4
Heskiaoff	28	205	6	8	8	6	7	7	6	7	7
		216	6	7	7	6	7	7	6	6	6
		256	5	6	6	5	6	6	5	5	5
		324	4	5	5	4	4	4	4	4	4
		342	4	4	4	4	4	4	4	4	4
Sawer	30	41	9	–	13	9	–	13	9	–	11
		47	8	–	11	8	–	11	8	–	10
		47	8	–	10	8	–	9	8	–	8
Killbridge	45	92	7	8	8	7	8	8	7	8	8
		110	6	7	7	6	7	7	6	6	6
		138	5	6	6	5	6	6	5	5	5
		184	4	4	4	4	4	4	4	4	4
Tonge	70	270	15	–	20	15	–	18	14	–	17
		320	13	–	16	12	–	15	12	–	14

NFS no feasible solution

4.2.8 Convergence

After a while, all the empires except the most powerful one will eliminate and all the colonies will be under the control of this unique empire, and the process will terminate and the last empire is the representation of best solution for the problem. Another stopping criterion may be considered, which is the number of decade (iteration). The algorithm continues until it reaches to the number of predefined iteration. In this research, the number of iteration is set to 250. The second stopping criterion is used in this paper.

5 Experimental design

5.1 Data generating

Several test problems with various cycle times, task time variances, and non-completion probabilities are solved. Test

Table 9 Comparison of ICA and GA solution (Low variance)

Problem	Tasks	CT	$K_{(1-\alpha)} = 1.96$			$K_{(1-\alpha)} = 1.645$			$K_{(1-\alpha)} = 1.28$		
			LB	GA	ICA	LB	GA	ICA	LB	GA	ICA
Merten	7	8	5	5	5	4	5	5	4	5	5
		10	4	4	4	4	4	4	4	4	4
		15	3	3	3	3	3	3	3	3	3
		18	2	2	2	2	2	2	2	2	2
Bowman	8	20	5	6	6	5	6	6	4	5	5
Jaeschke	9	6	7	8	8	7	8	8	7	8	8
		7	6	7	7	6	7	7	6	7	7
		8	5	7	7	5	7	7	5	7	7
		10	4	5	5	4	5	5	4	5	5
		18	3	3	3	3	3	3	3	3	3
Jackson	11	9	6	7	7	6	7	7	6	7	7
		10	6	7	7	6	7	7	5	7	7
		13	4	5	5	4	5	5	4	5	5
		14	4	4	4	4	4	4	4	4	4
		21	5	3	3	3	3	3	3	3	3
Mitchell	21	15	NFS	NFS	NFS	NFS	NFS	NFS	8	9	9
		21	6	6	6	6	6	6	6	6	6
		26	5	5	5	5	5	5	5	5	5
		35	4	4	4	4	4	4	4	4	4
		39	3	4	4	3	4	4	3	3	3
Heskiaoff	28	205	6	6	6	6	6	6	6	6	6
		216	6	6	6	6	6	6	6	5	6
		256	5	5	5	5	5	5	5	5	5
		324	4	4	4	4	4	4	4	4	4
		342	4	4	4	4	4	4	4	4	4
Sawer	30	33	11	–	15	11	–	14	11	–	14
		41	9	–	11	9	–	11	9	–	10
		47	8	–	10	8	–	9	8	–	8
		33	11	–	15	11	–	11	11	–	14
Killbridge	45	79	8	9	9	8	9	9	8	8	8
		92	7	8	8	7	8	8	7	7	7
		110	6	6	6	6	6	6	6	6	6
		138	5	5	5	5	5	5	5	5	5
		184	4	4	4	4	4	4	4	4	4
Tonge	70	207	18	–	22	18	–	20	18	–	20
		234	16	–	19	16	–	19	16	–	18
		320	12	–	14	12	–	13	12	–	13

Table 10 The best cost of ICA versus GA (low task variance)

Problem size	Decrease		Similar	Increase	
	Percent of problem	Average decrease (%)	Percent of problem	Percent of problem	Average increase (%)
Small	14.3	15.3	69	16.7	0.7
Medium	73	3.7	24	3	0.1
Large	95	3.8	5	0	0.7

problems are considered in three categories: small-sized problems (Mertens, Bowman, Jaeschke, and Jackson with seven, eight, nine, and 11 tasks, respectively), medium-sized problems (Mitchell, Heskiaoff, and Sawer with 21, 28, and 30 tasks, respectively), and large-sized problems (Killbridge and Tonge with 45 and 70 tasks, respectively). Data of these problems are taken from the website http://www.bwl.tu-darmstadt.de/bwl3/. Task times are assumed to be normally distributed. Task variances are uniformly generated using the method of Carraway [58] in one of two ranges: *low variance* [0, $(\mu_i/4)^2$] and *high variance* [0, $(\mu_i/4)^2$]. The probability of completing time within fixed cycle time is considered 0.90, 0.95, and 0.975 ($K_{(1-\alpha)} = 1.28$, 1.645, and 1.96, respectively).

5.2 Calibration of the parameters and operators of the proposed ICA algorithm

An appropriate design of the parameters and operators has a significant impact on the efficiency and effectiveness of the algorithm. Most users often adjust the parameters and operators manually based on the reference values of previous, similar literature. In this section, we study the behavior of the different parameters of the proposed ICA. In order to calibrate the algorithms, there are several ways to statistically design the experimental investigation, but to reduce the number of required tests, a fractional factorial experiment (FFE) was developed [59]. FFE allows only a portion of the total possible combinations to estimate the main effect of the factors and some of their interactions. Taguchi [60] developed a family of FFE matrices that eventually reduces the number of experiments, but still provides sufficient information. In the Taguchi method, orthogonal arrays are used to study a large number of decision variables with a small number of experiments.

Taguchi separates the factors into two main groups: controllable and noise factors. Noise factors are those over which we have no direct control. Since elimination of the noise factors is impractical and often impossible, the Taguchi method seeks to minimize the effect of noise and to determine the optimal level of the important controllable factors based on the concept of robustness [61]. In addition to determining the optimal levels, Taguchi establishes the relative significance of individual factors in terms of their main effects on the objective function.

Taguchi created a transformation of the repetition data to another value which is the measure of variation. The transformation is the signal-to-noise (*S*/*N*) ratio, which explains why this type of parameter design is called a robust design [62, 63]. Here, the term "signal" denotes the desirable value (response variable), and "noise" denotes the undesirable value (standard deviation). So, the *S*/*N* ratio indicates the amount of variation present in the response variable. The aim is to maximize the signal-to-noise ratio.

Taguchi classifies objective functions into three categories: the smaller-the-better type, the larger-the-better type, and the nominal-is-best type. Since almost all objective functions in assembly line balancing are classified in the smaller-the-better type, the corresponding *S*/*N* ratio [63] is:

$$S/N\,\text{ratio} = -10 \times \log\left(\frac{1}{k}\sum_{i=1}^{k}(\text{objective function})_i^2\right) \tag{15}$$

that *k* represents the number of experiments for each problem.

Table 11 The best cost of ICA versus GA (high task variance)

Problem size	Decrease		Similar	Increase	
	Percent of problem	Average decrease (%)	Percent of problem	Percent of problem	Average increase (%)
Small	27.8	0.6	55.6	16.7	0.5
Medium	70.4	11.5	2	28	1.5
Large	90.5	7.6	9.5	0	0

Table 12 Computational time of ICA versus GA (low task variance)

Problem size	Decrease		Similar	Increase	
	Percent of problem	Average decrease (%)	Percent of problem	Percent of problem	Average increase (%)
Small	71.4	6.2	0	28.6	2.5
Medium	90	19	0	10	0.6
Large	95.2	5.8	0	4.8	0.2

The control factors are assimilation rate (AR), revolution rate (RR), (N_{country}, N_{imp}), and ξ. Different levels of these factors are shown in Table 2.

The associated degree of freedom for theses four factors is 9. Therefore, the selected orthogonal array should have a minimum of nine rows and four columns to accommodate the four factors. From the standard table of orthogonal arrays, L_9 is selected as the fittest orthogonal array design that fulfills all our minimum requirements. The orthogonal array L_9 is presented in Table 3.

In order to conduct the experiments, we implemented ICA in MATLAB 7.6 run on a personal computer with a 2.5 GHz Intel Core 2 Duo processor and 4 GB RAM memory.

There are five replicates for each combination of instances. We use the relative percentage deviation (RPD) for the cost as a common performance measure to compare the methods. The best solutions obtained for each instance (denoted Min_{sol}) are calculated. RPD is obtained from the formula:

$$\text{RPD} = \frac{\text{Alg}_{\text{sol}} - \text{Min}_{\text{sol}}}{\text{Min}_{\text{sol}}} \times 100\% \tag{16}$$

where Alg_{sol} is the cost obtained for a given algorithm and instance. Clearly, lower values of RPD are preferable.

After obtaining the results of the Taguchi experiment for all the trials, the optimal level of the factors A, B, C, and D are shown in Table 4. Results are indicated in Figs. 8, 9, and 10.

To explore the relative significance of individual factors in terms of their main effects on the objective function, analysis of variance (ANOVA) was conducted. The results of the analysis are presented in Tables 5, 6, and 7.

6 Computational results

ICA is implemented in MATLAB 7.6 run on a personal computer with a 2.5 GHz Intel Core 2 Duo processor and 4 GB RAM memory and tested on a set of benchmarks in literature. Several test problems with various cycle times, task time variances, and non-completion probabilities are solved. Test problems are considered in three categories: small-sized problems (Mertens, Bowman, Jaeschke, and Jackson with seven, eight, nine, and 11 tasks, respectively), medium-sized problems (Mitchell, Heskiaoff, and Sawer with 21, 28, and 30 tasks, respectively), and large-sized problems (Killbridge and Tonge with 45 and 70 tasks, respectively). Data of these problems are taken from the website http://www.bwl.tu-darmstadt.de/bwl3/. Task times are assumed to be normally distributed. Task variances are uniformly generated using the method of Carraway [58] in one of two ranges: low variance [0, $(\mu_i/4)^2$] and high variance [0, $(\mu_i/4)^2$]. Probability of completing time within fixed cycle time is considered 0.90, 0.95, and 0.975 ($K_{(1-\alpha)} = 1.28$, 1.645, and 1.96, respectively). The results of ICA and GA proposed by Baykasoğlu and Özbakir [7] are presented in Tables 8 (high variance) and 9 (low variance).

Statistics summaries in Tables 10 and 11 represent higher performance of proposed ICA versus GA for low and high variance problems, respectively. The proposed algorithm is also found to be quiet robust with respect to problem size. Tables 12 and 13 show the required computational time for ICA versus GA. Based on this data, the proposed algorithm has found optimal solutions within shorter computational time than GA.

Table 13 Computational time of ICA versus GA (high task variance)

Problem size	Decrease		Similar	Increase	
	Percent of problem	Average decrease (%)	Percent of problem	Percent of problem	Average increase (%)
Small	61	2.7	0	38	4.7
Medium	92.6	14.1	0	7.4	2.4
Large	100	7.4	0	0	0

7 Conclusion and further research

This paper studied new evolutionary algorithm, ICA, to solve stochastic U-type assembly line balancing problems (UALBP), with the aim of minimizing the number of work stations, idle time at each station, and non-completion probabilities of each station. Unlike the current evolutionary algorithms, such as GA and SA, which are computer simulations of natural processes such as natural evolution and annealing process in materials, ICA uses socio-political evolution processes as source of inspiration. The performance of the proposed algorithm is measured against genetic algorithm proposed by Baykasoğlu and Özbakir [7] (the best algorithm proposed before to solve stochastic UALBP with the aim of minimizing the number of work stations, idle time at each station, and non-completion probabilities of each station). The proposed method is tested on three categories of test problem (small, medium, and large sizes). The computational results indicate that the proposed algorithm outperforms GA. ICA has a higher convergence rate than GA, reaching to a better solution within shorter computational time.

We can also list the following directions for future research: First, as assimilation policy has relatively significant effect on performance of the ICA, some modifications can be implemented to improve the execution of the proposed algorithm. Second, the proposed algorithm can be extended to more complicated assembly line systems such as mixed and multi-model assembly lines. Third, task times are assumed to be normally distributed. Other skewed distributions can be applied in future researches. Fourth, generalization of this method to other type of assembly line balancing problems and multi-objective cases should be investigated in future studies.

References

1. Park K, Park S, Kim W (1997) A heuristic for an assembly line balancing problem with incompatibility, range, and partial precedence constraints. Comput Ind Eng 32:321–332
2. Hautsch K, John H, Schürgers H (1972) Taktabstimmung bei FlieXarbeit mit dem Positionswert-Verfahren. REFANachrichten 25:451–464
3. Lapierre SD, Ruiz AB (2004) Balancing assembly lines: an industrial case study. J Oper Res Soc 55:589–597
4. Malakooti B (1991) A multiple criteria decision making approach for the assembly line balancing problem. Int J Prod Res 29:1979–2001
5. Boysen N, Fliedner M, Scholl A (2007) A classification of assembly line balancing problems. Eur J Oper Res 183:674–693
6. Urban TL, Chiang WC (2006) An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. Eur J Operation Res 168(3):109–120
7. Baykasoğlu A, Özbakir L (2007) Stochastic U-line balancing using genetic algorithms. Int J Adv Manuf Technol 32(1):139–147
8. Chiang WC, Urban TL (2006) The stochastic U-line balancing problem: a heuristic procedure. Eur J Oper Res 175(3):1767–1781
9. Hirano H (1988) JIT factory revolution. Productivity, Cambridge
10. Monden Y (1983) Toyota production system. Industrial Engineering and Management, Norcross
11. Sekine K (1992) One-piece flow. Productivity, Portland
12. Scholl A, Klein R (1999) ULINO: optimally balancing U-shaped JIT assembly lines. Int J Prod Res 37(4):721–736
13. Baybars I (1986) A survey of exact algorithms for the simple line balancing problem. Manage Sci 32:909–932
14. Erel E, Sarin SC (1998) A survey of the assembly line balancing procedures. Prod Plann Control 9:414–434
15. Ghosh S, Gagnon RJ (1989) A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. Int J Prod Res 27:637–670
16. Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168:666–693
17. Miltenburg J, Wijngaard J (1994) The U-line line balancing problem. Manage Sci 40(10):1378–1388
18. Miltenburg GJ, Sparling D (1995) Optimal solution algorithms for the U-line balancing problem. Working Paper, McMaster University, Hamilton, Ontario, Canada.
19. Urban T (1998) Optimal balancing of U-shaped assembly lines. Manage Sci 44(5):738–741
20. Ajenblit DA, Wainwright RL (1998) Applying genetic algorithms to the U-shaped assembly line balancing problem. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, pp. 96–101
21. Erel E, Sabuncuoglu I, Aksu BA (2001) Balancing of U-type assembly systems using simulated annealing. Int J Prod Res 39 (13):3003–3015
22. Miltenburg J (1998) Balancing U-lines in a multiple U-line facility. Eur J Oper Res 109:1–23
23. Sparling D (1998) Balancing JIT production units: the N U-line balancing problem. Inf Syst Oper Res 36:215–237
24. Kim YK, Kim JY, Kim Y (2000) A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. Appl Intell 13:247–258
25. Sparling D, Miltenburg J (1998) The mixed-model U-line balancing problem. Int J Prod Res 36:485–501
26. Visich JK, Diaz-Saiz J, Khumawala BM (2002) Development of heuristics to reduce model imbalance for the mixed model, U-shaped assembly line. Proc Annu Meet Decis Sci Inst 2002:1786–1791
27. Kara Y, Özcan U, Peker A (2007) An approach for balancing and sequencing mixed-model JIT U-lines. Int J Adv Manuf Technol 32:1218–1231
28. Kara Y, Özcan U, Peker A (2007) Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives. Appl Math Comput 184(2):566–588
29. Nakade K, Ohno K (1999) An optimal worker allocation problem for a U-shaped production line. Int J Prod Econ 60(1):353–358
30. Miltenburg J (2000) The effect of breakdowns on U-shaped production lines. Int J Prod Res 38:353–364
31. Miltenburg J (2001) One-piece flow manufacturing on U-shaped production lines: a tutorial. IIE Trans 33:303–321
32. Nakade K, Ohno K (2003) Separate and carousel type allocations of workers in a U-shaped production line. Eur J Oper Res 145:403–424
33. Aase GR, Schniederjans MJ, Olson JR (2003) U-OPT: an analysis of exact U-shaped line balancing procedures. Int J Prod Res 41 (17):4185–4210
34. Aase GR, Olson JR, Schniederjans MJ (2004) U-shaped assembly line layouts and their impact on labor productivity: an experimental study. Eur J Oper Res 156:698–711
35. Martinez U, Duff WS (2004) Heuristic approaches to solve the U-shaped line balancing problem augmented by genetic algorithms.

In: Proceedings of the 2004 Systems and Information Engineering Design Symposium (SIEDS 2004), Charlottesville, Virginia, April 2004, pp. 287–293
36. Gökçen H, Ağpak K, Gencer C, Kizilkaya E (2005) A shortest route formulation of simple U-type assembly line balancing problem. Appl Math Model 29:373–380
37. Gökçen H, Ağpak K (2006) A goal programming approach to simple U-line balancing problem. Eur J Oper Res 171:577–585
38. Baykasoğlu A (2006) Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. J Intell Manuf 17:217–232
39. Hwang RK, Katayama H, Gen M (2007) U-shaped assembly line balancing problem with genetic algorithm. Int J Prod Res 46 (16):3797–3822
40. Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 168(3):694–715
41. Toklu B, Özcan U (2008) A fuzzy goal programming model for the simple U-line balancing problem with multiple objectives. Eng Optim 40(3):191–204
42. Boysen N, Fliedner M (2008) A versatile algorithm for assembly line balancing. Eur J Oper Res 184:39–56
43. Toksari MD, Isleyen SK, Guner E, Baykoc OF (2008) Simple and U-type assembly line balancing problems with a learning effect. Appl Math Model 32(12):2954–2961
44. Özcan U, Toklu B (2008) A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems. J Intell Manuf 20(1):123–136
45. Sabuncuoglu I, Erel E, Alp A (2009) Ant colony optimization for the single model U-type assembly line balancing problem. Int J Prod Econ 120(2):287–300
46. Baykasoğlu A, Dereli T (2009) Simple and U-type assembly line balancing by using an ant colony based algorithm. Math Comput Appl 14(1):1–12
47. Hwang R, Katayama H (2009) A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. Int J Prod Res 47(14):3797–3822
48. Simaria AS, Zanella M, de Sá M, Vilarinho PM (2009) Meeting demand variation using flexible U-shaped assembly lines. Int J Prod Res 47(14):3937–3955
49. Zhang Z, Cheng Z, Song L, Yu O (2009) A heuristic approach for fuzzy U-shaped Line balancing problem. Sixth International Conference on Fuzzy Systems and Knowledge Discovery 4:228–232
50. Chand S, Zeng T (2001) A comparison of U-line and straightline performances under stochastic task times. Manuf Serv Oper Manage 3(2):138–150
51. Ağpak K, Gökçen H, Saray N, Özel S (2002) A heuristic for single model U-line assembly line balancing with stochastic task time. J Fac Eng Arch Gazi Univ 17(4):115–124
52. Chiang WC, Urban TL (2002) A hybrid heuristic for the stochastic U-line balancing problem. Working Paper, University of Tulsa, Oklahoma
53. Guerriero F, Miltenburg J (2003) The stochastic U-line balancing problem. Nav Res Logist 50(1):31–57
54. Erel E, Sabuncuoglu I, Sederci H (2005) Stochastic assembly line balancing using beam search. Int J Prod Res 43:1411–1426
55. Arcus AL (1966) COMSOAL: a computer method of sequencing operations for assembly lines. Intell J Prod Res 4(4):259–277
56. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. IEEE Congress on Evolutionary Computation, Singapore
57. Bautista J, Suarez R, Mateo M, Companys R (2000) Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, California, April 2000, pp. 2404–2409
58. Carraway RL (1989) A dynamic programming approach to stochastic assembly line balancing. Manage Sci 35(4):459–470
59. Cochran WG, Cox GM (1992) Experimental designs, 2nd edn. Wiley, New York
60. Taguchi G (1986) Introduction to quality engineering. Asian Productivity Organization/UNIPUB, White Plains
61. Tsai JT, Ho WH, Liu TK, Chou JH (2007) Improved immune algorithm for global numerical optimization and job-shop scheduling problems. Appl Math Comput 194:406–424
62. Al-Aomar R (2006) Incorporating robustness into genetic algorithm search of stochastic simulation outputs. Simul Model Pract Theory 14:201–223
63. Phadke MS (1989) Quality engineering using robust design. Prentice-Hall, Englewood Cliffs