International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

A modified colonial competitive algorithm for the mixed-model U-line balancing and sequencing problem

Kunlei Lian [a], Chaoyong Zhang [a], Liang Gao [a] & Xinyu Shao [a]
[a] State Key Lab of Digital Manufacturing Equipment & Technology, Huazhong University of Science & Technology (HUST), Wuhan, Hubei, PR China

Available online: 01 Mar 2012

To cite this article: Kunlei Lian, Chaoyong Zhang, Liang Gao & Xinyu Shao (2012): A modified colonial competitive algorithm for the mixed-model U-line balancing and sequencing problem, International Journal of Production Research, DOI:10.1080/00207543.2011.653453

To link to this article: http://dx.doi.org/10.1080/00207543.2011.653453

International Journal of Production Research
2012, 1–15, iFirst

A modified colonial competitive algorithm for the mixed-model U-line balancing and sequencing problem

Kunlei Lian, Chaoyong Zhang*, Liang Gao and Xinyu Shao

State Key Lab of Digital Manufacturing Equipment & Technology, Huazhong University of Science & Technology (HUST), Wuhan, Hubei, PR China

(*Received 29 July 2011; final version received 22 December 2011*)

Implementation of mixed-model U-shaped assembly lines (MMUL) is emerging and thriving in modern manufacturing systems owing to adaptation to changes in market demand and application of just-in-time production principles. In this study, the line balancing and model sequencing (MS) problems in MMUL are considered simultaneously, which results in the NP-hard mixed-model U-line balancing and sequencing (MMUL/BS) problem. A colonial competitive algorithm (CCA) is developed and modified to solve the MMUL/BS problem. The modified CCA (MCCA) improves performance of original CCA by introducing a third type of country, independent country, to the population of countries maintained by CCA. Implementation details of the proposed CCA and MCCA are elaborated using an illustrative example. Performance of the proposed algorithms is tested on a set of test-bed problems and compared with that of existing algorithms such as co-evolutionary algorithm, endosymbiotic evolutionary algorithm, simulated annealing, and genetic algorithm. Computational results and comparisons show that the proposed algorithms can improve the results obtained by existing algorithms developed for MMUL/BS.

Keywords: mixed-model U-line balancing and sequencing; colonial competitive algorithm; genetic algorithm

1. Introduction

This paper deals with the balancing and sequencing problem in mixed-model U-shaped assembly lines. A typical assembly line consists of a number of workstations on which work contents of products are performed and a mechanical material handling system that is responsible for conveying product among stations. To begin with, work content of a product is divided into some basic portions called tasks that are assigned to existing stations. The necessary time to perform a task is called task time. All the tasks need to be performed to create a finished product. A product visits the stations sequentially, and on each station, one or more tasks is performed within a limited operation time called the cycle time. The product is moved to the next station after the cycle time is reached. An important issue in the design of an assembly line is assembly-line balancing (ALB). ALB is the act of assigning the tasks to the stations according to the precedence constraints among tasks to achieve some specific objectives, such as the minimisation of the number of stations for a given cycle time (type I), the minimisation of the cycle time for a given number of stations (type II), or the maximisation of the efficiency of the assembly line (type III) (Scholl and Klein 1999). This paper addresses the ALB problem of type II. This type of problem generally exists when the number of stations within the organisation is restricted, or no new stations are available. In other words, this type II problem aims at producing the maximum number of items on the basis of existing stations (Özcan *et al.* 2010).

The mixed-model U-shaped line (MMUL) studied in this study is characterised by two aspects: mixed-model production and a U-shaped assembly line. Mixed-model production refers to the strategy that one or more product types with similar characteristics are produced on the same assembly line. Compared with a single-model assembly line, mixed-model production could provide more flexibility to adapt to the changes in market demand. On the other hand, a U-shaped assembly has been widely used in various industries as a consequence of adaptation of Just-In-Time (JIT) principles. The most distinctive feature that distinguishes the U-line from a straight line is that both the entrance and the exit of U-line are at the same position. The benefits of U-shaped production line include a reduced number of workers, increased visibility and communication, and improved capability of adjusting to changes in external environments (Özcan *et al.* 2010).

*Corresponding author. Email: zcyhust@mail.hust.edu.cn

ISSN 0020–7543 print/ISSN 1366–588X online

http://dx.doi.org/10.1080/00207543.2011.653453
http://www.tandfonline.com

For the efficient implementation of MMUL, two interrelated problems named line balancing and model sequencing must be solved simultaneously. Line balancing is the problem of assigning tasks to stations such that some objectives are optimised. Model sequencing is the problem of determining the production sequence of models. This paper investigates the mixed-model U-line balancing and sequencing problem (MMUL/BS). MMUL/BS is more complex than the mixed-model U-line balancing problem that is proved to be NP-Hard (Sparling and Miltenburg 1998) because model sequencing in MMUL/BS is also to be determined.

To tackle this computationally intractable problem, efficient algorithms that can obtain an optimal or near-optimal solution in reasonable time are necessary. In this paper, a novel metaheuristic algorithm named a colonial competitive algorithm (CCA) is employed and modified to address the MMUL/BS problem. CCA is a population-based algorithm that gets its inspiration from the socio-political process of imperialistic competition. It starts from an initial population of potential solutions named countries and converges to optimality through assimilation, imperialistic competition, and revolution. CCA has been applied to solve many combinatorial optimisation problems successfully. Therefore, we utilise CCA in this paper to address the MMUL/BS problem.

The remainder of the paper is organised as follows. Section 2 presents the literature review about MMUL/BS problem and the proposed CCA; Section 3 describes the MMUL/BS in detail; implementation details of CCA are given in Section 4; Section 5 provides the modified CCA; Section 6 shows the computational results; and Section 7 concludes the paper.

2. Literature review

There exist many studies addressing ALB or the assembly sequencing problem. Readers are referred to Ghosh and Gagnon (1989), Erel and Sarin (1998), Becker and Scholl (2006), and Scholl and Becker (2006) for reviews on these problems. This paper focuses on the existing studies on the MMUL/BS problem and CCA.

Sparling and Miltenburg (1998) has been widely accepted as the first study to address the mixed-model U-line balancing (MMUL/B) problem. They proposed an approximate solution algorithm that first transforms the MMUL/B into a single-model U-line balancing (SMUL/B) problem by calculating the weighted average processing times and merging each model's precedence graph into a single precedence graph. Then, the SMUL/B problem is solved by a SMUL/B-T branch-and-bound algorithm to generate an initial solution that is smoothed by a smoothing algorithm. The objective is to minimise the absolute deviation of workloads (ADW) among stations. Kim *et al*. (2000a) presented a co-evolutionary algorithm (CEA) to solve the balancing and sequencing problems simultaneously in mixed model assembly lines. Kim *et al*. (2000b) studied the line balancing and model sequencing problem in mixed-model U-lines and suggested a co-evolutionary algorithm to solve the two problems at the same time. They proposed a localised evolution strategy to promote population diversity and search efficiency. Miltenburg (2002) designed a genetic algorithm (GA) to solve the joint problem of line balancing and model sequencing in mixed-model, U-shaped, asynchronous assembly lines. Kim *et al*. (2006) developed an endosymbiotic evolutionary algorithm (EEA) that extended the cooperative evolutionary algorithm by imitating the natural evolution process of endosymbionts. EEA was applied to solve the both problems of balancing and sequencing in MMULs simultaneously. Kara *et al*. (2007a) proposed a simulated annealing (SA) algorithm to minimise the number of workstations as well as the ADW among stations in mixed-model U-lines. Kara *et al*. (2007b) developed a simulated annealing algorithm to deal with the MMUL/BS problem with multiple objectives, such as ADW across workstations, part usage rate and cost of setups. Kara (2008) presented a mixed, zero-one, nonlinear mathematical programming formulation for balancing and sequencing MMULs simultaneously with the objective of reducing work overload. They suggested a simulated annealing algorithm for this problem. Kara and Tekin (2009) presented a mixed integer programming formulation for optimal balancing of MMUL and proposed a new heuristic solution procedure to handle large-scale MMUL balancing problems. Hwang and Katayama (2009) proposed a multi-decision genetic approach for workload balancing of MMULs. The performance criteria they considered include the number of workstations and the variation of workload. Özcan *et al*. (2010) proposed a GA for the MMUL/BS problem with stochastic task time. Owing to its computationally intractability MMUL/BS, efficient algorithms that can obtain optimal or near-optimal solutions in reasonable time are necessary. Most of the current studies focus on utilising GA or SA to solve MMUL/BS; this paper proposes a new approach to address this problem with the objective of minimising ADW.

CCA is a novel population-based metaheuristic algorithm that gets its inspiration from the socio-political process of imperialistic competition (Atashpaz-Gargari and Lucas 2007). Initially designed for solving continuous

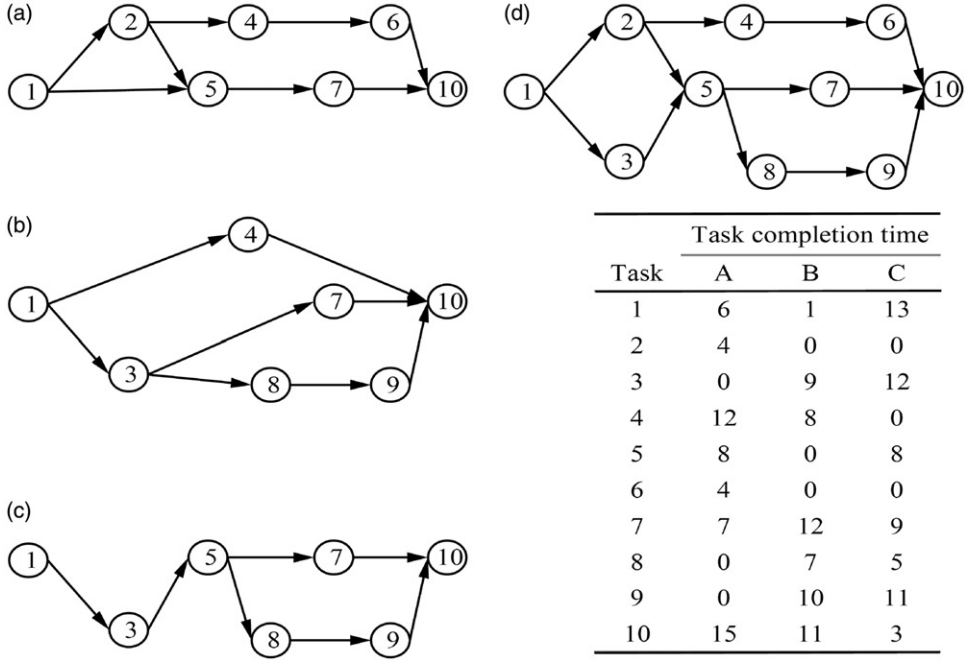
optimisation problems, CCA has been modified and applied to address many combinatorial optimisation problems in various domains. Successful implementations of CCA include PID controller design (Atashpaz-Gargari *et al*. 2008), Nash equilibrium point achievement (Rajabioun *et al*. 2008), Stochastic U-shaped line balancing (Bagher *et al*. 2010), group scheduling in flexible flow shops (Karimi *et al*. 2011), integrated product mix-outsourcing (Nazari-Shirkouhi *et al*. 2010), dynamic cell formation (Sarayloo and Tavakkoli-Moghaddam 2010), bi-criteria scheduling of the assembly flow shops (Shokrollahpour *et al*. 2010), and data clustering (Niknam *et al*. 2011). In this paper, the CCA is developed and modified to solve the MMUL/BS problem with the objective of ADW minimisation.

3. Mixed-model U-line balancing and sequencing problem

As discussed above, MMUL is characterised by two aspects: mixed-model production and U-shaped assembly line. Mixed-model refers to the production type that a set of products (models) $M$with similar production characteristics are produced on the same assembly line over a specified planning horizon. The demand for each model $m(m \in M)$ is denoted by D_m. Each model has its own precedence relationships among tasks that can be depicted using a precedence diagram. All the precedence graphs are then combined into a single precedence diagram. Figure 1 shows precedence diagrams and task times of three different models (a, b, c) and their combined precedence diagram (d). Each node represents a task and the arrow connecting two different tasks indicates their precedence relationships. Each task in the combined precedence diagram has a different task time for a different model. A task time of zero indicates that this task is not performed by a model. The model sequencing problem considered in this study is based on the Minimum Part Set (MPS) principle. Let q be the greatest common divisor of $D_m(m \in M)$. An MPS is defined as $d = (d_1, d_2, \ldots, d_M), d_m = D_m/q$. The demands for all models can be accomplished by repeating the MPS q times. Then, the length of the model sequence (MS) for one MPS can be calculated as:

$$L = \sum_{m=1}^{M} d_m. \tag{1}$$

The type II ALB problem considered in this paper assumes that the number of stations is determined in advance, and all tasks in the combined precedence diagram are assigned to these stations, which results in a line balance (LB).



	Task completion time		
Task	A	B	C
1	6	1	13
2	4	0	0
3	0	9	12
4	12	8	0
5	8	0	8
6	4	0	0
7	7	12	9
8	0	7	5
9	0	10	11
10	15	11	3

Figure 1. Precedence diagrams and task times for three models.

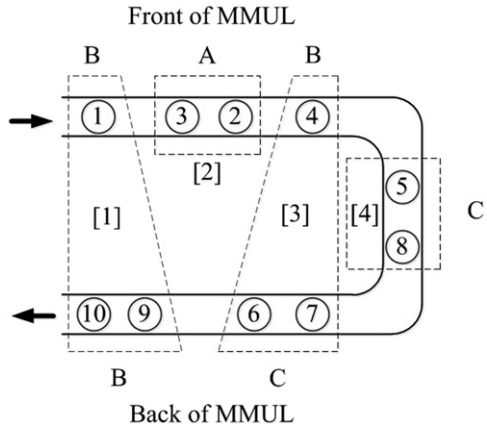


Figure 2. Illustrative MMUL example.

Figure 2 shows the assignment of tasks in Figure 1 on an MMUL with four stations. Assume the MPS to be $d = (1, 2, 2)$ and the resulting model sequence (MS) to be BCCBA.

In Figure 2, there are two crossover stations ([1] and [3]) and two regular stations ([2] and [4]). The existence of a crossover station is the most important difference between U-shaped line and straight line. A crossover station can have tasks located at both the front and the back of the assembly line, while a regular station can only have tasks located at the front or the back of the line.

After the determination of LB and MS, the objective considered in this paper, ADW, can be calculated. The following notations are used to describe ADW:

I	total number of tasks in the combined precedence diagram ($i = 1, 2, \ldots, I$)
J	total number of stations utilised on the MMUL ($j = 1, 2, \ldots, J$)
t_{im}	completion time of task i for model m
C_{min}	theoretical minimum cycle time: $C_{min} = (1/(J \times L)) \sum_{i=1}^{I} \sum_{m=1}^{M} d_m t_{im}$
SF_j	set of tasks in workstation j located on the front of the MMUL
SB_j	set of tasks in workstation j located on the back of the MMUL
α_j^r	model produced on the front of station j at cycle r
β_j^r	model produced on the back of station j at cycle r
W_{jr}	workload of station j at cycle r

$$W_{jr} = \sum_{i \in SF_j} t_{i\alpha_j^r} + \sum_{i \in SB_j} t_{i\beta_j^r} \; j = 1, 2, \ldots J; \; r = 1, 2, \ldots, L. \tag{2}$$

ADW can be computed as follows:

$$ADW = \sum_{j=1}^{J} \sum_{r=1}^{L} \left| W_{jr} - C_{min} \right|. \tag{3}$$

4. Colonial competitive algorithm (CCA) for MMULBS

4.1 *Schematic workflow of proposed CCA*

The Colonial Competitive Algorithm (CCA) is a novel population-based metaheuristic algorithm inspired by the socio-political process of imperialistic competition. Similar to some other population-based metaheuristic algorithms, CCA maintains a population of solutions to the problem to be solved throughout the optimisation process. *Cost* and *power* are used to represent the objective value and fitness of each country respectively. Note that

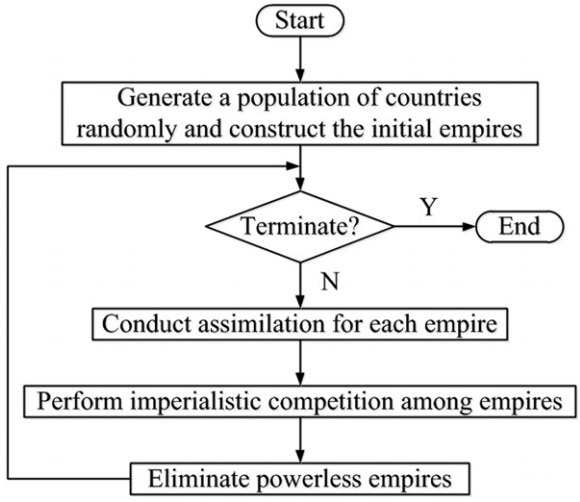


Figure 3. Workflow of basic colonial competitive algorithm.

in the optimisation problem considered in this paper, the power of each country is inversely proportional to its cost (ADW). The basic CCA algorithm mainly consists of the following four components:

(1) *Initial empires construction*. Each individual in the population of CCA is called a *Country* that is generated either randomly or using some heuristic rules. Countries in the initial population are classified into two categories based on the objective considered: *Imperialist* and *Colony*. To begin with, some best countries are selected as the imperialists and the rest countries form the colonies. Each colony is then assigned to an imperialist and an imperialist and all its colonies construct an *Empire*. The number of population N_{pop} and the number of imperialists N_{imp} in the initial population are parameters of CCA.
(2) *Assimilation*. Assimilation functions in each empire at each iteration of CCA. It refers to the process that the imperialist in an empire tries to influence all its colonies by making them more similar to itself. For the optimisation problem considered in this paper, assimilation is accomplished by moving the colony toward its imperialist using the crossover operator (the concept borrowed from GA) (Shokrollahpour*et al.* 2010). In addition, mutation operator is also applied to each colony after movement to enhance population diversity. Assimilation would probably result in better colonies compared with original ones. If the resulting colony after assimilation is better than the imperialist, it becomes the imperialist and vice versa.
(3) *Imperialistic competition*. Different from assimilation which happens inside an empire, imperialistic competition acts among all the empires. During the imperialistic competition, all the empires try to possess more colonies, which is realised by freeing the weakest colony of the weakest empire and making a competition among all empires to possess this freed colony. Each empire has a probability to possess this free colony based on its total power. Imperialistic competition results in power increase in some empires and power decrease in other empires.
(4) *Elimination*. Imperialistic competition would result in an empire with no colonies, which means that its colonies are possessed by the other empires. In this case, the empire with no colony is eliminated from the population.

Procedures of CCA are described as follows and shown in Figure 3:

Step 1: Parameters initialisation.
Step 2: Initial population generation and empires construction.
Step 3: If termination criterion is not met, repeat the following steps.
Step 4: Assimilation.
Step 5: Imperialistic competition.
Step 6: Elimination.

Details of CCA are elaborated in the following subsections.

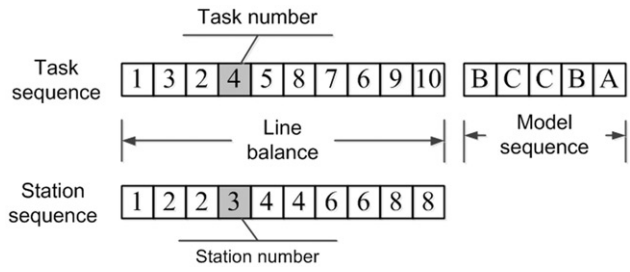


Figure 4. Proposed encoding scheme.

4.2 *Proposed encoding and decoding scheme*

4.2.1 *Solution representation*

In this paper, each country represents a solution to the MMUL/BS problem. Since line balancing (LB) and model sequencing (MS) of MMUL are solved simultaneously, the encoded solution in this paper consists of two parts: LB part and MS part. For the LB part, different from the group-number representation scheme proposed by Kim *et al.* (2006), we suggested a new encoding scheme in this study. Figure 4 illustrates the encoded solution of the example given in Figure 2. Specifically, LB is determined by two sequences: task sequence (TS) and station sequence (SS). TS refers to the order of all the tasks assigned to stations from the front to the back of MMUL. Note that TS must satisfy the precedence constraints among tasks. SS is a non-decreasing sequence with the same length of TS, and each element of SS is used to decide the station number on which the task with the same index in TS is performed. The value of each element in SS falls into the scope of $[1, 2J]$. For example, the fourth element of TS in Figure 4 is task 4, and the fourth element of SS in Figure 4 represents station 3. The length of TS and SS equals the total number of tasks. MS part of each solution determines the sequence that models are released to performed on MMUL.

In the proposed encoding scheme, both TS and SS are generated randomly to ensure the diversity of initial population. Note that the TS generated randomly or manipulated by mutation (described in a later section) may violate the precedence constraints among tasks. Therefore, this paper introduces a repairing algorithm to adjust an infeasible TS to feasible one (Tseng 2006). Following the repairing algorithm given in Table 1, an infeasible TS could be adjusted into an feasible one. Note also that SS must be non-decreasing, so SS is sorted non-decreasingly as long as it violates the rule.

4.2.2 *Solution evaluation*

We illustrate the calculation process of ADW using the example given in Figure 2. To compute ADW, the workload W_{jr} of each station at each cycle should be obtained first. Table 2 shows the workload of each station at each cycle.

The theoretical minimum cycle time C_{min} for the illustrative example is:

$$\begin{aligned} C_{min} &= (1/(J \times L)) \sum_{i=1}^{I} \sum_{m=1}^{M} d_m t_{im} \\ &= (1/(4 \times 5)) \sum_{i=1}^{4} \sum_{m=1}^{3} d_m t_{im} \\ &= (1/(4 \times 5))(56 + 58 \times 2 + 61 \times 2) \\ &= 14.7. \end{aligned}$$

Then, the ADW for the example could be computed as:

$$\begin{aligned} ADW &= \sum_{j=1}^{J} \sum_{r=1}^{L} \left| W_{jr} - C_{min} \right| \\ &= \sum_{j=1}^{4} \sum_{r=1}^{5} \left| W_{jr} - 14.7 \right| \\ &= 118.8. \end{aligned}$$

Table 1. Repairing algorithm for *TS*.

Notations used in the repairing algorithm:
- TS task sequence
- g_h the task in the hth position of TS
- r root node point
- l leaf node point

Repairing algorithm:
Step 1: Set $h=2$.
Step 2: Set g_1's corresponding task at root node point R.
Step 3: Set g_h's corresponding task at leaf node point l, and decide the precedence relationship of r and l.
(1) If $p_{r,l}=1$, task l should be performed before task r.
  (a) If r's left child node point is not empty, then set r's left node point at the new root node point r and repeat Step 3.
  (b) If r's left child node point is empty, then insert l at r's left node point. Set $h=h+1$ and go to Step 4.
(2) If $p_{r,l}=0$, there is no precedence constraints between task r and l.
  (a) If r's right node child node point is not empty, then set r's right node point at the new root node point r and go to Step 3.
  (b) If r's right child node point is empty, then insert l at r's right node point. Set $h=h+1$, and got to Step 4.
Step 4: If $h=m$, go to Step 5; otherwise go to Step 2.
Step 5: List feasible TS according to the in-order traversal rank and stop the algorithm.

Table 2. Workload of each station.

	Workstation											
	1			2			3			4		
SF_j SB_j	[1]	[9, 10]		[3, 2]	[∅]		[4]	[6, 7]		[5, 8]	[∅]	
Cycle	α_j^r	β_j^r	W_{jr}	α_j^r	β_j^r	W_{jr}	α_j^r	β_j^r	W_{jr}	α_j^r	β_j^r	W_{jr}
1	B	B	**22**	A	–	4	B	C	**17**	C	–	13
2	C	C	**27**	B	–	9	A	C	**21**	B	–	7
3	C	C	**27**	C	–	12	B	B	**20**	A	–	8
4	B	B	**22**	C	–	12	C	A	11	B	–	7
5	A	A	**21**	B	–	9	C	B	12	C	–	13

4.3 *Initial empires construction*

After the generation of initial population of countries, some of the best countries are selected as the imperialists. The size of initial population is N_{pop}, the number of imperialists is N_{imp}, and the number of colonies is N_{col}. To distribute the colonies among imperialists, the normalised cost (ADW) of each imperialist must be computed using the following formula:

$$C_n = max(c_i) - c_n, \tag{4}$$

where c_n is the cost of the nth imperialist, and C_n is its normalised cost. The colonies are distributed among imperialists based on their normalised power. The normalised power of each imperialist is defined by

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right|. \tag{5}$$

Then, the number of colonies of an empire will be

$$NC_n = round(p_n \cdot N_{col}). \tag{6}$$

An imperialist with its corresponding colonies constructs an empire.

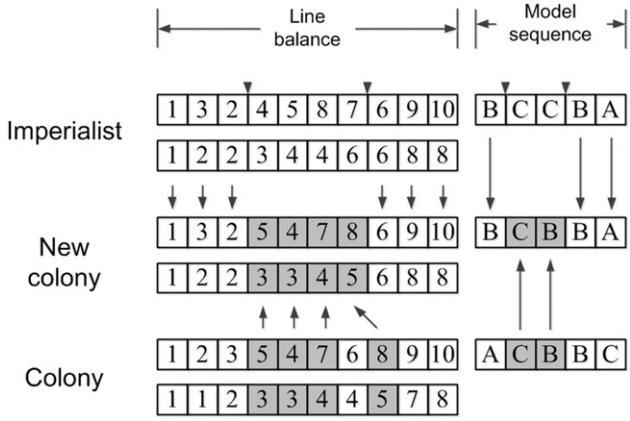


Figure 5. Crossover operator applied to an imperialist and its colonies.

4.4 *Assimilation*

4.4.1 *Moving colonies of an empire toward the imperialist*

Assimilation refers to the movement of a colony to its imperialist. Since CCA was initially designed to solve continuous optimisation problems, the operators of crossover and mutation were borrowed from a GA to tackle with discrete optimisation problems. Specifically, crossover is applied between a colony and its imperialist to exchange properties with each other, which makes the colony more like its imperialist. In addition, mutation is performed on each colony after crossover to introduce diversity to the population. Crossover and mutation are described as follows.

The crossover operator consists of LB crossover and MS crossover, and two-point crossover is used in this paper. The crossover of LB is defined as follows: (1) initialise an LB for the new colony and select two crossover points, c1 and c2 (c1 $<$ c2), randomly; (2) copy the elements of LB in imperialist that locate before c1 and after c2 to the same position of LB in the new colony. Delete these elements from the current colony; (3) copy the remaining elements of the current colony sequentially to the remaining positions of the new colony. The crossover of MS is defined as follows: (1) initialise an MS for the new colony and select two crossover points, p1 and p2 (p1 $<$ p2), randomly; (2) copy the elements of MS in imperialist that locate before p1 and after p2 to the same positions of MS in the new colony; (3) copy the elements of MS in the current colony that locate between p1 and p2 to the same positions of MS in new colony. Another new colony could be obtained by exchanging roles of the imperialist and the current colony. The new colony with lower cost becomes the final new colony. Figure 5 illustrates the crossover operator using an example.

The mutation operator consists of mutation of TS in LB, mutation of SS in LB and mutation of MS. Mutation of TS in LB is defined as the following: randomly select two tasks and swap their positions. Mutation of SS in LB is defined as: randomly select a station and change it to another value. Mutation of MS is defined as: randomly select two different models and swap their positions. Note that the TS and SS in LB may become infeasible after crossover of mutation; in this case, the repairing algorithm described in former sections will function to adjust them into feasible ones.

4.4.2 *Position exchange of an imperialist and a colony*

The colony after crossover and mutation may reach a position with a lower cost than that of its imperialist. In this case, the colony will become the imperialist in the current empire and vice versa. In the following iterations, colonies in the empire will move to the new imperialist.

4.5 *Imperialistic competition*

In ICA, all empires compete to take possession of more colonies besides their current colonies. The imperialistic competition gradually results in a decrease in the power of weaker empires and an increase in the power of

powerful ones. To model this competition among imperialists, the weakest colony of the weakest empire is freed from its current imperialist and waited to be possessed by other empires. During the competing process, each empire will have a likelihood of taking possession of the freed colony based on their total power, that is, empires with more total power will be more likely to possess it.

The total power of an empire is determined by the power of the imperialist and that of all its colonies, that is, the equation of total cost is:

$$T.C._n = c_n(\text{imperialist}_n) + \alpha \cdot \text{mean}(c_n(\text{colonies of empire}_n)), \tag{7}$$

where $T.C._n$ is the total cost of the nth empire, and α is a positive number that is in the range of [0, 1]. Different values of α indicate the weight of the cost of the imperialist on the total cost of an empire.

The normalised total cost is computed by

$$N.T.C._n = \max(T.C._i) - T.C._n, \tag{8}$$

where $T.C._n$ and $N.T.C._n$ are total cost and normalised total cost of the nth empire respectively. Then, the possession probability of each empire is given by

$$p_{p_n} = \left| \frac{N.T.C._n}{\sum_{i=1}^{N_{imp}} N.T.C._i} \right| \tag{9}$$

$$P = \left[p_{p_1}, p_{p_2}, \cdots, p_{p_{N_{imp}}} \right]. \tag{10}$$

Then, a vector R with the same size as P is created, and its elements are uniformly distributed random numbers.

$$R = \left[r_1, r_2, \cdots, r_{N_{imp}} \right]. \tag{11}$$

Then, a vector D is formed by simply subtracting R from P.

$$D = P - R = \left[D_1, D_2, \cdots, D_{N_{imp}} \right]. \tag{12}$$

The empire whose relevant index in D is largest will take possession of the freed colony.

4.6 *Elimination*

When an imperialist loses all of its colonies, it will be eliminated from the population.

4.7 *Termination*

In this paper, the predefined run time is used as the termination criterion.

5. Proposed modified colonial competitive algorithm

5.1 *Workflow of the modified CCA*

This paper modifies the original CCA by introducing a third type of country named *Independent Country* based on the more realistic modelling of the socio-politically process. Original CCA classifies the initial population of countries into only two types, namely, imperialists and colonies. However, independent countries generally coexist with imperialists and colonies historically. Table 3 describes the behaviours of three different types of countries.

The main differences between the modified CCA and original CCA lie in the phases of 'Initial empires construction' and 'Assimilation' owing to the existence of independent countries. The following sections will elaborate on the steps of modified CCA in detail.

Table 3. Country types and their behaviours.

Country type	Behaviour
Imperialist	Assimilate its colonies and try to possess more colonies
Colony	Learn from its imperialist and try to become imperialist or independent country
Independent country	Learn from all imperialists and try to become imperialist

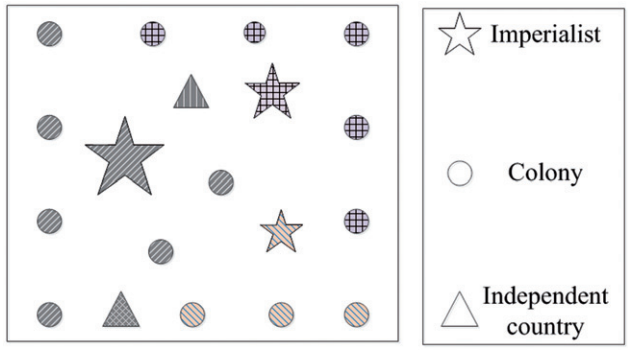


Figure 6. Initial empires construction and independent countries determination.

5.2 *Initial empires construction and independent countries determination*

Since the initial population of countries is classified into three types, the parameters of modified CCA include the size of population N_{pop}, the number of imperialists N_{imp}, and the number of independent countries N_{ind}. The selection of imperialists and distribution of colonies are the same as the steps described in Section 4.3. The N_{ind} independent countries are generated randomly. Note that N_{pop} equals the sum of N_{imp}, N_{ind}, and N_{col}. Figure 6 illustrates the initial population of countries including imperialists, colonies, and independent countries. The larger one pentagon's size is, the more colonies it possesses.

5.3 *Assimilation*

5.3.1 *Assimilation inside each empire*

Details of assimilation inside each empire are the same as those described in Section 4.4 and omitted here.

5.3.2 *Assimilation occurs between imperialists and independent countries*

In the modified CCA, all the independent countries try to become more powerful by learning from imperialists. For each independent country, it makes a move to all the current imperialists, which results in a number of new independent countries, and is then replaced by the best new independent countries. We illustrate the difference between assimilation inside each empire and assimilation between imperialists and independent countries using Figure 7. The two assimilation process also utilises the crossover and mutation operator described in Section 4.4.

5.3.3 *Position exchange among imperialists, colonies, and independent countries*

After assimilation inside each empire and between imperialists and independent countries, the following position exchange occurs:

(1) If a new colony inside an empire reaches a better solution than the imperialist, it become the new imperialist and vice versa. This is the same as described in section 4.4.
(2) If the weakest imperialist among all empires is worse than the best independent country among all independent countries, the best independent country become the imperialist and vice versa.

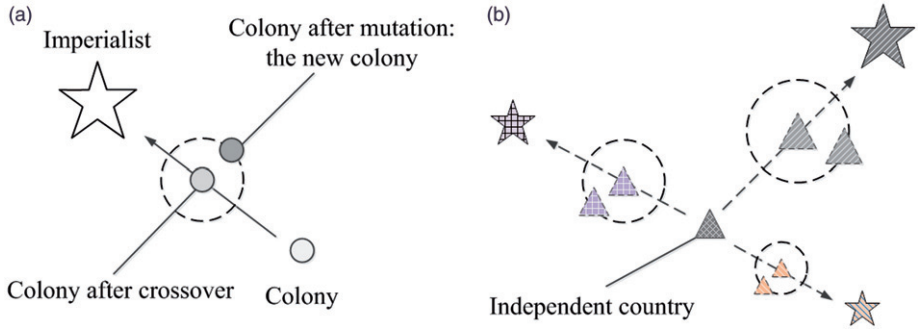


Figure 7. Assimilation comparison. (a) Assimilation inside each empire. (b) Assimilation between imperialists and independent country.

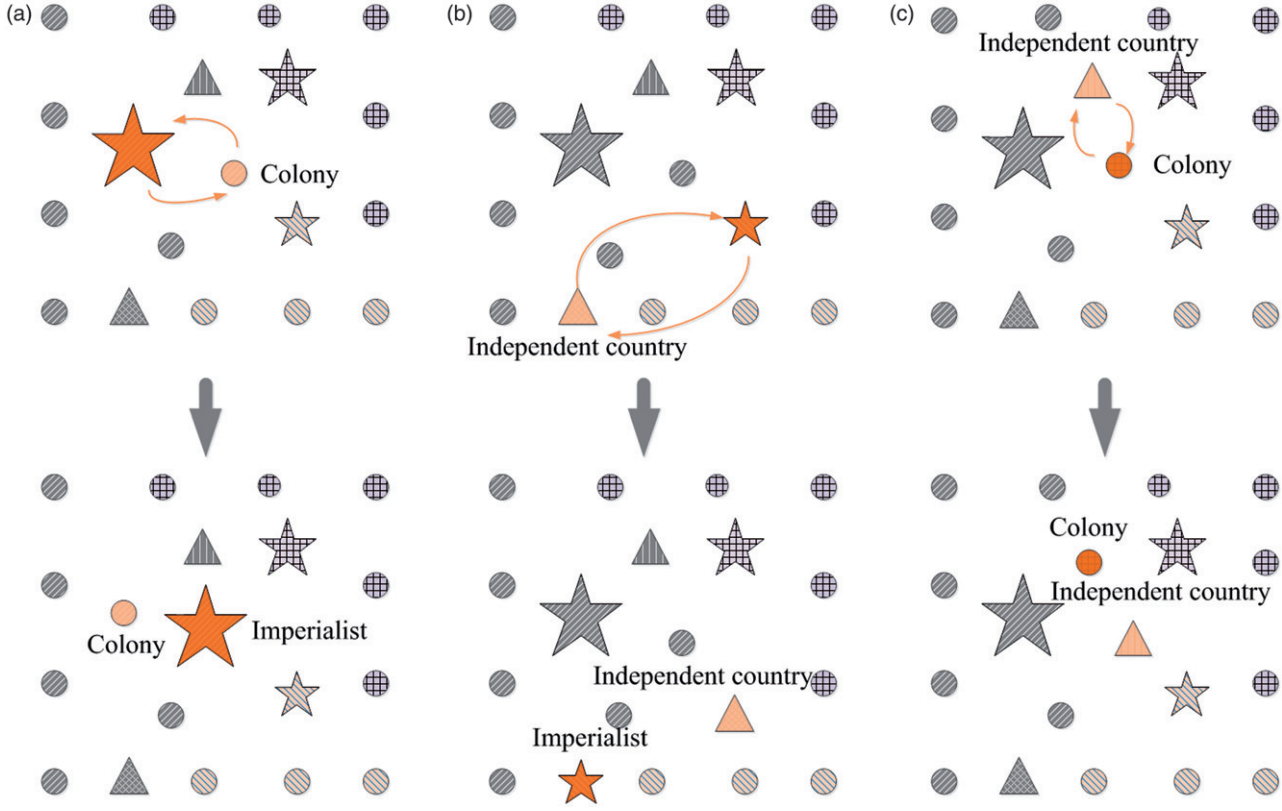


Figure 8. Position exchange of different types.

(3) If the best colony among all empires is better than the worst independent country among all independent countries, the best colony become the independent country and vice versa.

These three types of position exchange are depicted in Figure 8.

5.4 *Imperialistic competition, elimination, and termination*

The processes of imperialistic competition, elimination, and termination are the same as those described in Sections 4.5, 4.6, and 4.7, respectively, and are omitted here.

Table 4. Test-bed problems.

Problem	No. of tasks	No. of models	No. of stations	MPS
Thom1	19	3	3	1 1 1
Thom2	19	3	3	3 2 1
Thom3	19	3	4	1 1 1
Kim1	61	4	6	1 1 1 1
Kim2	61	4	6	1 3 4 5
Kim3	61	4	6	6 4 2 1
Kim4	61	4	12	1 1 1 1
Kim5	61	4	12	1 3 4 5
Kim6	61	4	12	6 4 2 1
Arcus1	111	5	12	1 1 1 1 1
Arcus2	111	5	12	5 3 2 1 1
Arcus3	111	5	12	1 2 4 5 8
Arcus4	111	5	12	1 4 8 3 1
Arcus5	111	5	15	1 1 1 1 1
Arcus6	111	5	15	5 3 2 1 1
Arcus7	111	5	15	1 2 4 5 8
Arcus8	111	5	15	1 4 8 3 1
Arcus9	111	5	27	1 1 1 1 1
Arcus10	111	5	27	5 3 2 1 1
Arcus11	111	5	27	1 2 4 5 8
Arcus12	111	5	27	1 4 8 3 1

6. Experiments

6.1 *Test-bed problems and compared algorithms*

Performance of the proposed CCA is evaluated on a set of benchmark problems taken from the literature (Kim *et al*. 2006). The problems include 19-task, 61-task, and 111-task problems with different number, of stations and MPS. Table 4 lists details of the test-bed problems.

6.2 *Parameter determination*

The algorithm implemented in this paper was coded in C++ and run on a personal computer with a 2.0 GHz Intel Core2 Duo CPU. The parameters of CCA include the total number of countries N_{pop}, the total number of empires N_{imp}, the total number of independent countries N_{ind}, the weight α, and the maximum number of iterations $MaxIter$. Preliminary experiments were conducted to determine the values of these parameters. Computational results showed that the following parameter values could yield satisfactory results: $N_{pop} = 100$, $N_{imp} = 7$, $N_{ind} = 5$, $\alpha = 0.6$.

6.3 *Computational results and performance comparison*

The characteristics and search capability of CCA and modified CCA (MCCA) are compared with those of the following algorithms: the co-evolutionary algorithm (CEA) proposed by Kim *et al*. (2000b), the endosymbiotic evolutionary algorithm (EEA) proposed by Kim *et al*. (2006), simulated annealing (SA) algorithm proposed by Kara (2008), and the GA proposed by Özcan *et al*. (2010). Table 5 shows the computational results and comparison of CCA with these algorithms. The first column indicates the problem numbers. The following four columns show the ADW obtained from the CEA, EEA, SA, and GA approaches respectively. These results are compared with those for the original CCA and MCCA presented in columns 7 and 8 respectively. For both CCA and MCCA on each test-bed problem, 10 independent experiments are conducted, and the best results are reported. Note that the results listed in columns 2 and 3 represent the average ADW, while the results given in columns 4 and 5 indicate the best ADW within ten runs of this algorithm. The last column indicates the improvement rate of MCCA to CCA for each test-bed problem. The last row represents the average improvement rate (AIR) for MCCA compared with each approach. The computation time for Thomopoulos's, Kim's, and Arcus's problems are 4, 47, and 308 s, respectively, to maintain consistency with EEA and GA (Kim *et al*. 2006, Özcan *et al*. 2010).

Table 5. Computational results and comparisons with existing approaches.

Problem	CEA	EEA	SA	GA	CCA	MCCA	Improvement rate (%)
Thom1	1.40	1.30	0.40	0.40	0.2	**0.2**	0.00
Thom2	2.78	2.40	2.20	2.20	2.0	**1.46667**	26.67
Thom3	1.56	1.30	0.80	0.60	0.6	**0.6**	0.00
Kim1	22.95	21.90	10.60	7.10	14.7	**7.0**	52.38
Kim2	86.39	78.30	64.22	52.59	74.859	**50.7**	32.27
Kim3	79.78	73.90	62.70	48.36	63.8205	**46.2923**	27.46
Kim4	44.75	38.00	24.70	19.75	37.45	**17.0**	54.61
Kim5	152.04	123.20	104.62	86.34	114.821	**86.2205**	24.91
Kim6	130.75	105.70	92.01	86.23	100.318	**78.559**	21.69
Arcus1	14,671.81	12,598.80	10,768.30	6168.70	7826.9	6490.40	17.08
Arcus2	33,800.51	31,080.60	28,057.23	12,303.56	24,267.3	17,060.40	29.70
Arcus3	50,044.05	45,342.10	37,121.96	16,720.00	19,643.3	**16,088.30**	18.10
Arcus4	49,218.96	42,661.60	37,899.05	21,734.39	26,386.4	**21,478.80**	18.60
Arcus5	19,852.23	16,809.80	8392.00	6356.96	8435.76	6870.48	18.56
Arcus6	48,490.48	39,379.20	26,584.00	15,282.13	26,810.4	16,819.60	37.26
Arcus7	76,396.39	61,223.60	24,863.10	18,592.53	22,550.5	**17,915.70**	20.55
Arcus8	69,053.40	54,659.80	26,252.08	24,642.56	28,616.2	**24,325.50**	14.99
Arcus9	63,909.82	46,254.70	29,291.52	20,700.00	31,468.7	25,491.90	18.99
Arcus10	15,0486.70	108,231.40	67,384.36	55,370.02	83,843.6	56,914.70	32.12
Arcus11	29,4429.10	209,222.20	123,041.70	93,412.89	155,308.0	**89,168.60**	42.59
Arcus12	23,3102.30	162,998.80	99,685.30	80,912.79	126,177.0	**79,091.10**	37.32
AIR	59.62	51.45	28.51	2.11	25.99	–	25.99

The following conclusions can be drawn from the observations in Table 5. First, the original CCA succeeds in outperforming both CEA and EEA for all test-bed problems. However, the original CCA fails to provide better results than SA and GA in nearly all cases, which indicates that the original CCA is less effective in exploring the solution space of MMUL/BS. Second, the comparison between CCA and MCCA shows that MCCA improves the original CCA greatly for all test-bed problems except Thom3 (the same best results are obtained), which validates MCCA's advantage over CCA in solving the MMUL/BS problem. Third, MCCA outperforms CEA, EEA, and SA for every test-bed problem, and MCCA improves 15 in 21 results obtained by GA. The bold values show that our algorithm obtains better results than those of the others. It can be seen from Table 5 that MCCA is very efficient in solving the MMUL/BS problem.

7. Conclusions

This paper proposes a novel metaheuristic algorithm named the colonial competitive algorithm (CCA) and its modified version (MCCA) to address the mixed-model U-line balancing and sequencing (MMUL/BS) problem. Implementation details of CCA/MCCA to MMUL/BS are elaborated using an illustrative example. Computational experiments are conducted to validate the performance of the proposed algorithms. Comparisons with existing algorithms show the effectiveness of CCA and MCCA. The contributions of this paper are summarised as follows:

(1) A novel metaheuristic algorithm called CCA is applied to solve the MMUL/BS problem. To the best of the authors' knowledge, this is the first application of CCA to an MMUL/BS problem. Computational results and comparisons with existing algorithms showed that CCA can yield promising results in a reasonable time.
(2) The original CCA is modified by introducing a third type of country, an independent country, in the initial population of countries. The performance of the modified CCA (MCCA) is compared with CCA and some other algorithm developed for MMUL/BS. Computational results show that MCCA outperforms CCA and improves several results obtained by other algorithms.
(3) A new representation scheme for line balancing of MMUL/BS problem is suggested. This representation scheme is simple to implement and efficient in solving the MMUL/BS problem.

In future research, CCA and MCCA can be applied to solve other combinatorial optimisation problems. The representation scheme proposed in this paper could be incorporated with other metaheuristic algorithms such as GA to obtain better results.

Acknowledgements

The authors thank the anonymous referees whose comments helped considerably to improve this paper. This research is supported by the State Key Programme of the National Natural Science Foundation of China (Grant No. 51035001), National Natural Science Foundation of China (Grant No. 50825503), the National High-Tech Research and Development Programme of China 863 Programme (Grant No. 2007AA04Z107) and the National Natural Science Foundation of China (Grant No. 50875101).

References

Atashpaz-Gargari, E., Hashemzadeh, F., and Lucas, C., 2008. Designing mimo piid controller using colonial competitive algorithm: Applied to distillation column process. *In: IEEE World Congress on Computational Intelligence*, 1–6 June, Hong Kong. Piscataway, NJ: IEEE, 1929–1934.

Atashpaz-Gargari, E. and Lucas, C., 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competitioned. *IEEE Congress on Evolutionary Computation*, 7, 4661–4666.

Bagher, M., Zandieh, M., and Farsijani, H., 2011. Balancing of stochastic u-type assembly lines: An imperialist competitive algorithm. *The International Journal of Advanced Manufacturing Technology*, 54 (1–4), 271–285.

Becker, C. and Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168 (3), 694–715.

Erel, E. and Sarin, S.C., 1998. A survey of the assembly line balancing procedures. *Production Planning & Control*, 9 (5), 414–434.

Ghosh, S. and Gagnon, R.J., 1989. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27 (4), 637.

Hwang, R. and Katayama, H., 2009. A multi-decision genetic approach for workload balancing of mixed-model u-shaped assembly line systems. *International Journal of Production Research*, 47 (14), 3797–3822.

Kara, Y., 2008. Line balancing and model sequencing to reduce work overload in mixed-model u-line production environments. *Engineering Optimization*, 40 (7), 669–684.

Kara, Y., Ozcan, U., and Peker, A., 2007a. An approach for balancing and sequencing mixed-model JIT u-lines. *International Journal of Advanced Manufacturing Technology*, 32 (11/12), 1218–1231.

Kara, Y., Ozcan, U., and Peker, A., 2007b. Balancing and sequencing mixed-model just-in-time u-lines with multiple objectives. *Applied Mathematics and Computation*, 184 (2), 566–588.

Kara, Y. and Tekin, M., 2009. A mixed integer linear programming formulation for optimal balancing of mixed-model u-lines. *International Journal of Production Research*, 47 (15), 4201–4233.

Karimi, N., Zandieh, M., and Najafi, A.A., 2011. Group scheduling in flexible flow shops: A hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism. *International Journal of Production Research*, 49 (16), 4965–4977.

Kim, Y.K., Kim, J.Y., and Kim, Y., 2000a. A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. *Applied Intelligence*, 13 (3), 247–258.

Kim, Y.K., Kim, J.Y., and Kim, Y., 2006. An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model u-lines. *European Journal of Operational Research*, 168 (3), 838–852.

Kim, Y.K., Kim, S.J., and Kim, J.Y., 2000b. Balancing and sequencing mixed-model u-lines with a co-evolutionary algorithm. *Production Planning & Control: The Management of Operations*, 11 (8), 754–764.

Miltenburg, J., 2002. Balancing and scheduling mixed-model u-shaped production lines. *International Journal of Flexible Manufacturing Systems*, 14 (2), 123–155.

Nazari-Shirkouhi, S., *et al*., 2010. Solving the integrated product mix-outsourcing problem using the imperialist competitive algorithm. *Expert Systems with Applications*, 37 (12), 7615–7626.

Niknam, T., *et al*., 2011. An efficient hybrid algorithm based on modified imperialist competitive algorithm and k-means for data clustering. *Engineering Applications of Artificial Intelligence*, 24 (2), 306–317.

Özcan, U., Kellegöz, T., and Toklu, B., 2011. A genetic algorithm for the stochastic mixed-model u-line balancing and sequencing problem. *International Journal of Production Research*, 49 (6), 1605–1626.

Rajabioun, R., Atashpaz-Gargari, E., and Lucas, C., 2008. Colonial competitive algorithm as a tool for nash equilibrium point achievement. *In*: O. Gervasi, B. Murgante, A. Laganá, D. Taniar, Y. Mun, and M.L. Gavrilova, eds. *Computational science and its applications – iccsa 2008*. Berlin: Springer, 680–695.

Sarayloo, F. and Tavakkoli-Moghaddam, R., 2010. Imperialistic competitive algorithm for solving a dynamic cell formation problem with production planning. *In*: D.-S. Huang, X. Zhang, C.A. Reyes Garcia, and L. Zhang, eds. *Advanced intelligent computing theories and applications*. Berlin: Springer, 266–276.

Scholl, A. and Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168 (3), 666–693.

Scholl, A. and Klein, R., 1999. Ulino: Optimally balancing u-shaped JIT assembly lines. *International Journal of Production Research*, 37 (4), 721–736.

Shokrollahpour, E., Zandieh, M., and Dorri, B., 2010. A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. *International Journal of Production Research*, 49 (11), 3087–3103.

Sparling, D. and Miltenburg, J., 1998. The mixed-model u-line balancing problem. *International Journal of Production Research*, 36 (2), 485–501.

Tseng, H.E., 2006. Guided genetic algorithms for solving a larger constraint assembly problem. *International Journal of Production Research*, 44 (3), 601–625.