Mathematical model and metaheuristics for simultaneous balancing and sequencing of robotic mixed-model assembly line

Zixiang Li[1,2], J. Mukund Nilakantan[3*], Qiuhua Tang[1,2], Peter Nielsen[3]
[1]Industrial Engineering Department, Wuhan University of Science and Technology, Wuhan 430081, Hubei, China.
[2]Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, Hubei, China.
Email: zixiangli@126.com, tangqiuhua@wust.edu.cn
[3]Department of Materials and Production, Aalborg University, Denmark.
Email: {mnj, peter}@m-tech.aau.dk

Abstract: This research provides the first method to simultaneously balance and sequence robotic mixed-model assembly lines (RMALB/S), which involves three sub-problems: task assignment, model sequencing and robot allocation. A new mixed-integer programming model is developed to minimize makespan and using CPLEX solver, small-size problems are solved for optimality. Two metaheuristics; restarted simulated annealing algorithm and co-evolutionary algorithm, are developed and improved to address this NP-hard problem. Restarted simulated annealing method replaces the current temperature with a new temperature to restart the search process. Co-evolutionary method utilizes a restart mechanism to generate a new population by modifying several vectors simultaneously. Proposed algorithms are tested on a set of benchmark problems and compared with five other recent high-performing metaheuristics. The proposed algorithms outperform their original editions and the benchmarked methods. Proposed algorithms are able to solve the balancing and sequencing problem of a robotic mixed-model assembly line effectively and efficiently.

Keywords: Assembly line balancing; Model sequencing; Robotic assembly line; Simulated annealing; Co-evolutionary algorithm

1. Introduction

Assembly line balancing problems are extensively studied combinatorial optimization problems (Sivasankaran and Shahabudeen 2014). They have great applications in the automotive industries and consumer electronics industries. It can be characterized with loss of generality, by a set of tasks that must be divided to and processed on a set of workstations. In modern industry, different variants of assembly lines are studied due to complex realistic production environments. Among these, two important variants are robotic assembly line and mixed-model assembly line (Çil, Mete, and Ağpak 2017).

In robotic assembly lines, robots are allocated to workstations to perform tasks replacing manual labor (Gao et al. 2009). Robots can operate for 24 hours a day without fatigue and provide large flexibility in assembly of products. Robots can be programmed to perform different types of tasks, while preserving the quality of products. A layout of a robotic assembly line is depicted in Figure.1. Major goal of such a robotic assembly line is to balance it by efficient assignment of tasks and robot allocation; two sub-problems that must be optimized simultaneously (Li, Tang, and Zhang 2016a).

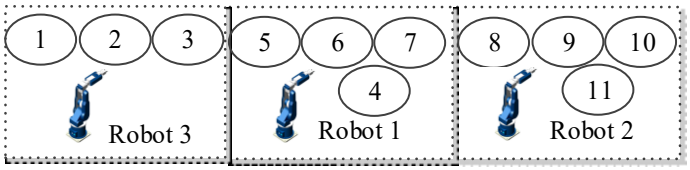


Figure.1 Layout of robot assembly line

Mixed-model production mode is extensively utilized due to the diversified needs of customers. In this type of assembly line, two sub-problems are encountered: mixed-model assembly line balancing and model sequencing (Kim, Kim, and Kim 2000a; Kim, Kim, and Kim 2000b). Mixed-model assembly line balancing tackles the assignment of the tasks to workstations (Akpinar and Bayhan 2014), whereas model sequencing determines the model operation sequence on the assembly line (Rabbani et al. 2015; Liu et al. 2014). When combining the features of the robotic assembly line with the mixed-model assembly line, a robotic mixed-model assembly line emerges, where a robot operates variations of tasks of different models on each workstation. In robotic mixed-model assembly lines, task assignment, model sequencing and robot allocation are interrelated sub-problems.

To the authors' best knowledge, no research has been reported which addresses robotic mixed-model assembly line balancing and sequencing. To tackle the balancing and sequencing of robotic mixed-model assembly lines (RMALB/S), this paper presents the following main contributions. (1) A generic mixed-integer programming model (MILP) is developed to minimize the makespan. This model is solved using CPLEX solver for small-size problems and a set of benchmark problems are generated to test the performance of the model. (2) Two metaheuristic methods: restarted simulated annealing algorithm (RSA) and co-evolutionary genetic algorithm (RCoGA) are developed to tackle large-size problems. These algorithms are selected due to their effectiveness and wide applications in other mixed-model assembly line balancing and sequencing (Kim, Kim, and Kim 2000b; Kim, Kim, and Kim 2000a; Mosadegh, Zandieh, and Ghomi 2012). To apply these two methods effectively to RMALB/S, several improvements are developed. A comprehensive study and comparative evaluation demonstrate the superiority of the two improved methods.

The remainder of the paper is organized as follows. In Section 2, the relative literature is provided. In Section 3, the proposed mathematical model is presented. Section 4 details the proposed two algorithms. Section 5 presents the benchmark problems and experimental comparisons. Finally, Section 6 concludes this research and presents possible future works.

2. Literature review

Although robotic assembly lines and mixed-model assembly line balancing and sequencing have been thoroughly studied independently, no work has been reported where both the problems are solved together. This section first presents a brief review of robot assembly line balancing (RALB) problems and later a review of literature on simultaneous balancing and sequencing of mixed-model assembly lines (MALB/S).

Since the first work on RALB problems by Rubinovitz and Bukchin (1991), many contributions have been reported. Rubinovitz, Bukchin, and Lenz (1993) develop a branch-and-bound algorithm for designing and balancing the robotic assembly lines. After that, metaheuristic methods have been widely applied to solve this problem due to its NP-hard nature (Gao et al. 2009). Levitin, Rubinovitz, and Shnits (2006) use genetic algorithm for RALB problems with the objective of cycle time minimization. Gao et al. (2009) propose an improved genetic algorithm for the same problem. Yoosefelahi et al. (2012) presents multi-objective model for the RALB problem and develop an evolution-based algorithm. Mukund Nilakantan et al. (2015) utilize bio-inspired search algorithms. Subsequently, they tackle the U-shaped layout of the RALB problem (Mukund Nilakantan and Ponnambalam 2016) and energy consumption in the RALB problem (Mukund Nilakantan, Huang, and Ponnambalam 2015) using particle swarm optimization algorithms. The research on energy consumption is followed by Li, Tang, and Zhang (2016a) where the energy consumption on two-sided RALB problem is handled. Li et al. (2016) consider the two-sided RALB problem using co-evolutionary particle swarm optimization algorithm and Aghajani, Ghodsi, and Javadi (2014) consider the mixed-model two-sided RALB problem with a simulating annealing method. Recently, Rabbani, Mousavi, and Farrokhi-Asl (2016) tackle the multi-objective type II robotic mixed-model assembly line balancing problem. Çil, Mete, and Ağpak (2017) develop a beam search to minimize the sum of cycle times of all models in robotic mixed-model assembly lines.

With regards to mixed assembly line balancing and sequencing (MALB/S) problems, two main methods have been applied: hierarchical method and simultaneous approach. This section primarily focuses on the simultaneous approach since the simultaneous approach outperforms hierarchical method (Kim, Kim, and Kim 2000a) and it is considered in this paper. The research on MALB/S can be divided into three categories based on the layout of the assembly line: one-sided MALB/S, U-type MALB/S and two-sided MALB/S. For one-sided MALB/S, Kim, Kim, and Kim (2000a) develop a co-evolutionary genetic algorithm to minimize the utility work. Battini et al. (2009) consider the MALB/S with finite buffer capacity and is tested using simulation software. Özcan et al. (2010) consider balancing and sequencing of the parallel mixed-model assembly lines. Mosadegh, Zandieh, and Ghomi (2012) take into account the station-dependent MALB/S. Öztürk et al. (2013) introduce balancing and scheduling of flexible mixed-model assembly lines. They also extend the problem by considering parallel stations and develop a MILP model. Saif et al. (2014) solve the multi-objective MALB/S using an artificial bee colony algorithm. Faccio, Gamberi, and Bortolini (2016) solve the paced MALB/S with jolly operators and develop a hierarchical approach.

Regarding the U-type MALB/S, Kim, Kim, and Kim (2000b) solve this problem using a co-evolutionary algorithm. Kim, Kim, and Kim (2006) develop a new method called endosymbiotic evolutionary algorithm. Kara, Ozcan, and Peker (2007a) address U-type MALB/S where the number of stations is optimized. Subsequently, they extend this problem by involving multiple-objectives (Kara, Ozcan, and Peker 2007b). Hamzadayi and Yildiz (2012) consider the U-type MALB/S with

parallel workstations and zoning constraints. In further work they improve the simulated annealing algorithm by employing tabu search algorithm (Hamzadayi and Yildiz 2013) to tackle the U-type MALB/S with the objective of minimizing workstations. Manavizadeh, Rabbani, and Radmehr (2015) consider multiple objectives in U-type MALB/S. In the case of two-sided MALB/S, Kucukkoc and Zhang (2014b) report the first research on mixed-model parallel two-sided assembly lines and develop an agent-based ant colony optimization method. Kucukkoc and Zhang (2014a) solve the same problem with the improved edition of this algorithm. Subsequently, Kucukkoc and Zhang (2016) develop a hybrid algorithm by hybridizing ant colony optimization and genetic algorithm for solving the same problem.

From the literature review and to authors' best knowledge, no research deals with robotic mixed model assembly line balancing and sequencing. Furthermore, it is observed that simulated annealing and co-evolutionary algorithms are widely utilized in solving MALB/S. Therefore, this research applies the simulated annealing and co-evolutionary algorithm to solve this new RMALB/S problem.

3. Mathematical formulation

3.1 Problem assumptions

In robotic mixed-model assembly lines, several tasks are assigned to each workstation and each workstation is allocated with a robot to perform the allocated tasks. Different types of products, referred to as models, are assembled in a sequence. RMALB/S involves three sub-problems: task assignment, model sequencing and robot assignment. This section mainly presents the basic assumptions based on Gao et al. (2009) and Mukund Nilakantan, Huang, and Ponnambalam (2015).

(1) Tasks are assigned to workstation only when the precedence constraint is satisfied.
(2) A robot can be allocated to any workstation and each robot must be allocated to a workstation.
(3) The number of the types of the robots is equal to that of workstations and each workstation has to have a robot allocated.
(4) All models are similar and merged into a combined precedence diagram.
(5) The operation times of models differ from each other and they are determined by the type of robots allocated.
(6) The operation times of a model by a robot are deterministic.
(7) Parallel workstations, setup times, work-in-process inventory, material handling are negligible.

3.2 Notations

▪ Indices:
i,j,h: Index of tasks.
k: Index of stations.
r,t: Index of robots.
m: Product model.
s: A model in the model sequence.

▪ Parameters:
Nt: Number of tasks.
Nk: Number of workstations.
Nr: Number of robots.
Nm: Types of models.
I: Set of tasks in combined precedence diagram, $I = \{1,2, \dots, i, \dots, Nt\}$.
K: Set of stations, $K = \{1,2, \dots, k, \dots, Nk\}$.
M: Set of product models, $M = \{1,2, \dots, m, \dots, Nm\}$
d_m: Demand for product model m in Master Production Schedule (MPS).
D: Total demand for all products in minimum part set, $D = \sum_{m}^{Nm} d_m$.
S: Set of sequence of product model, $S = \{1,2, \dots, s, \dots, D\}$.
t_{imr}: Operation time of task i in model m by robot r.
$Pre(i)$: Set of predecessors of task i.
$Suc(i)$: Set of successor of task i.
SNP: Set of ordering pairs which have no precedence relations,
$SNP = \{(i,j) | i \in I, j \in \{h | h \in I - Pre(i) \cup Suc(i) and \ i < j\}\}$.

▪ Decision variables:
x_{ik}: 1, if task i is assigned to workstation k; 0, otherwise.

r_{rk}: 1, robot r is allocated to workstation k; 0, otherwise.
t_{imr}: Finishing time of task i.
z_{ms}: 1, if product model m is in the s sequence; 0, otherwise.
P_{km}: Total operation time of tasks on station k for model m.
C_{sk}: Completion time of all the tasks of the model in s sequence on workstation k.

▪ Indicator variables:
w_{ij}: if task i is assigned before task j in the same workstation; 0, if task j is assigned before task i in same workstation.

3.3 Mathematical model for RMALB/S

This research considers RMALB/S with an objective of minimizing the makespan. In RMALB, once the robot allocation and task assignment are determined, the model sequencing is somewhat transferred into a permutation flow shop scheduling problem (Ruiz and Stützle 2007; Ruiz, Maroto, and Alcaraz 2006). In permutation flow shop scheduling, a set of independent tasks are processed on a set of machines and job processing sequence are set to be the same throughout all the machines (Ruiz and Stützle 2007). In this case, each workstation is regarded as a machine, total assigned tasks on a workstation can be regarded as a job and model sequencing corresponds to job sequencing. Since the makespan minimization criterion is the most commonly studied in literature, this paper adopts it as the objective for both balancing and sequencing problems. The makespan minimization criterion is roughly analogue to the reduction of cycle time, which is often found in line balancing literature. The detailed formulation is presented as follows.

$$\min Makespan = \max_{s=\{1,2,\dots,D\}} (C_s, Nk) \quad (1)$$

$$\sum_{k=1}^{Nk} x_{ik} = 1 \quad \forall i \quad (2)$$

$$\sum_{k=1}^{Nk} k.x_{ik} - \sum_{k=1}^{Nk} k.x_{jk} \le 0 \quad \forall i \in Pre(j); j \quad (3)$$

$$t_{jm}^{f} - t_{im}^{f} + \psi(1 - x_{ik}) + \psi\left(1 - x_{jk}\right) \ge \sum_{r=!}^{Nr} t_{jmr} \cdot y_{rk} \quad \forall i \in Pre(j); j \quad (4)$$

$$t_{jm}^{f} - t_{im}^{f} + \psi(1 - x_{ik}) + \psi\left(1 - x_{jk}\right) + \psi\left(1 - w_{ij}\right) \ge \sum_{r=!}^{Nr} t_{jmr} \cdot y_{rk} \quad \forall (i,j) \in SNP \quad (5)$$

$$t_{im}^{f} - t_{jm}^{f} + \psi(1 - x_{ik}) + \psi\left(1 - x_{jk}\right) + \psi.w_{ij} \ge \sum_{r=!}^{Nr} t_{jmr} \cdot y_{rk} \quad \forall (i,j) \in SNP \quad (6)$$

$$t_{im}^{f} + \psi(1 - x_{ik}) \ge \sum_{r=!}^{Nr} t_{imr} \cdot y_{rk} \quad \forall i,k \quad (7)$$

$$\sum_{r=1}^{Nr} y_{rk} = 1 \quad \forall k \quad (8)$$

$$\sum_{k=1}^{Nk} y_{rk} = 1 \quad \forall r \quad (9)$$

$$P_{km} + \psi(1 - x_{ik}) \ge \sum_{r=!}^{Nr} t_{imr} \cdot y_{rk} \quad \forall i,k \quad (10)$$

$$\sum_{m=1}^{Nm} z_{ms} = 1 \quad \forall s \in S \quad (11)$$

$$\sum_{s=1}^{D} z_{ms} = d_m \quad \forall m \in M \quad (12)$$

$$C_{11} + \psi(1 - z_{m1}) \ge P_{1m} \quad \forall m \in M \quad (13)$$

$$C_{s,k+1} + \psi(1 - z_{ms}) \ge C_{sk} + P_{k+1,m} \quad \forall s \in \{1,2,\dots,D\}, k \in \{1,2,\dots,Nk-1\} \quad (14)$$

$$C_{s+1,k} + \psi(1 - z_{ms}) \ge C_{sk} + P_{k,m} \quad \forall s \in \{1,2,\dots,D-1\}, k \in \{1,2,\dots,Nk\} \quad (15)$$

Equation (1) minimizes the makespan which is the maximum of $C_{s,Nk}$. Constraint (2) indicates that each task must be allocated to a workstation. Constraint (3) handles precedence constraints that guarantee that the processors of task j must be assigned before task j. Constraints (4-6) evaluates the finishing time of tasks of models. If there is a precedence relationship between task j and task i, constraint (4) takes effect and is reduced to $t_{jm}^{f} - t_{im}^{f} \ge \sum_{r=1}^{Nr} t_{jmr}.y_{rk}$. If there is no precedence relationship between task i and task j, constraint (5) and constraint (6) come into play. If task i is assigned before task j in a same workstation, constraint (5) is reduced to $t_{jm}^{f} - t_{im}^{f} \ge \sum_{r=1}^{Nr} t_{jmr}.y_{rk}$; otherwise, constraint (6) is reduced to $t_{im}^{f} - t_{jm}^{f} \ge \sum_{r=1}^{Nr} t_{imr}.y_{rk}$. Constraint (7) ensures that the finishing time of each task is larger than or equal to its operation time. Constraints (8-9) deal with the robot allocation, where constraint (8) indicates that each workstation has a robot and constraint (9) denotes that each robot is allocated to a workstation. Constraint (10) evaluates the total operation times of tasks on station k for model m, which is the largest finishing time of tasks assignment station k for model m. Constraint (11) guarantees that there is a model at each sequence and constraint (12) fulfills the MPS demand. Constraints (13-15) ensures that the completion time of all the tasks of the model in the s sequence on workstation k does not exceed the total processing time and that the

sequence is maintained. Constraint (13) obtains the value of the finish time of the model in the first sequence on the first workstation. Constraint (14) indicates that the tasks on the latter workstations cannot be operated until the tasks on the former workstations are finished. Constraint (15) ensures that the tasks of the model on the latter position of the model sequence can be operated only when the tasks of model on the former position are finished.

4. Proposed metaheuristic algorithms

RMALB/S problems mainly aim at optimizing the assignment of tasks, the allocation of the robots and the model sequence simultaneously. The proposed RMALB/S problem belongs to the NP-hard category due to the complexity involved. To optimize these sub-problems simultaneously, encoding scheme, decoding scheme and effective algorithms are essential. This section first introduces the applied encoding scheme and decoding scheme and later describes the two improved metaheuristics methods.

4.1 Encoding and decoding

A task assignment vector, robot allocation vector and model sequence vector are employed to encode three sub-problems. Task assignment vector is a $1\times Nt$ vector, in which each element denotes a workstation. If element in i^{th} position is k, task i is assigned to workstation k. The robot allocation is a vector $1\times Nr$ vector, where each element denotes a robot. If the element in the i^{th} position is r, robot r is allocated to workstation i. The model sequence vector is a $1\times Nm$ vector, each element corresponding to a model. If the element in the i^{th} position is m, model m is i^{th} one to be assembled. Decoding for the robot allocation vector and model sequence is straightforward, whereas the detailed task assignment vector needs the task assignment vector and the consideration of precedence constraints. A task is assignable only when its predecessors have been applied and task sequence on a workstation is determined in sequence based on the precedence constraint. An example with 11 tasks, 4 robots and 2 models is depicted in Figure.2, where demands for two models (model A and model B) from the MPS are 1 and 2. In Figure.2, the element in the third position of the task assignment vector is 1, and thus task 3 is assigned to workstation 1. The element in the first position of robot allocation vector is 3, and thus robot 3 is allocated to workstation 1. The element in the first position of model sequence vector is B, and thus model B is first assembled.

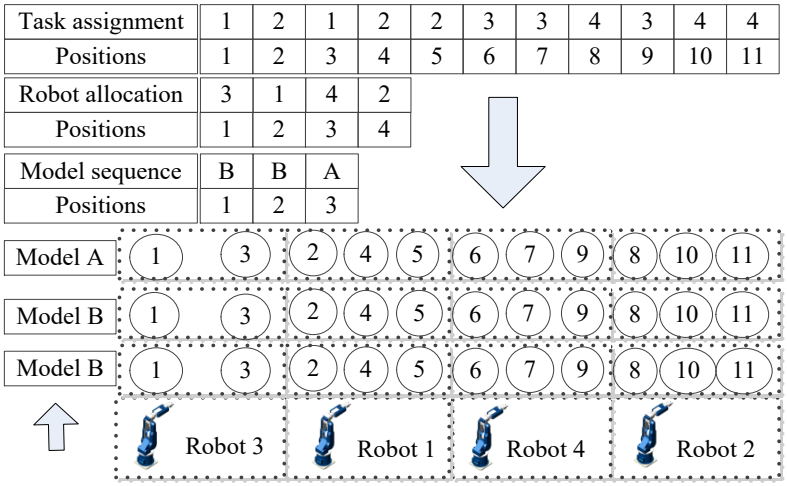


Figure.2 Example of encoding and decoding

For the initialization process, three vectors are randomly generated. For task assignment vector, a random number between 1 and Nk is generated in each position. This random task assignment vector might violate precedence constraints, and thus a repair procedure presented in Li, Tang, and Zhang (2016a) is utilized. This repair mechanism exchanges the positions of two tasks when successor of a task is assigned to the former workstation. This repair mechanism terminates when corresponding workstation of each task is smaller than or equal to any of the corresponding workstations of its successors. Regarding the robot allocation vector, a number is randomly selected with uniform likelihood from $1,2,\ldots,\mathrm{Nr}$ in the first position, a number is randomly selected among the remaining numbers in the second position and the last remaining number is allocated to the last position. For sequence vector, the same method as used for robot allocation is applied. The encoding and decoding scheme are the basis of the metaheuristic algorithms and the two proposed metaheuristic algorithms

are detailed in the following sections.

4.2 Restarted simulated annealing algorithm

Simulated annealing (SA) algorithms is a local search method that has shown outstanding results for solving MALB/S problems (Mosadegh, Zandieh, and Ghomi 2012). This paper improves the original edition by employing a restart mechanism, where the current temperature is replaced with a new restart temperature when new global best individual cannot be further achieved. The logic behind this modification is utilizing proper values of initial temperature (T_0) and cooling rate (α) to quickly find a local optimum and replacing the current temperature to escape from local optima. The procedure of the proposed restarted simulated annealing (RSA) algorithm is illustrated in Figure.3. In this figure, T indicates the current temperature updated with $T = \alpha \ X \ T$, NS is the number of iterations before current temperature update, $f(S)$ is the achieved fitness of solution S, T_R denotes restart temperature utilized to replace current temperature and RT is the restart time (RT) before replacing the current temperature.

This algorithm has five parameters: T_0, α, NS, T_R and RT. RSA starts with generating an initial solution by generating three vectors randomly. Then a loop repeats until a termination criterion is satisfied, in which *NewBest* is applied to check whether a new best individual is achieved. Within the loop, a neighborhood solution is firstly generated utilizing one of neighborhood structures which are further explained in Section 4.4. Subsequently, this new solution S' is compared with the incumbent one $S(\Delta = f(S') - f(S))$. If this new solution is better than the incumbent one $(\Delta \leq 0)$, the incumbent one is updated with the new one. Otherwise, the incumbent one is replaced by the new one with a probability calculated by $exp^{-\Delta/(T \times f(S))}$. Thirdly, if a new best solution has not been achieved for RT iterations and the current temperature is less than the restart temperature, the current temperature is replaced by the restart temperature. It is to be noted that the restarted simulated annealing algorithm is translated into the original edition when the restart temperature is set to 0.0. This simple modification improves the performance of the simulated annealing algorithm by increasing the capacity of escaping from local optima.

Algorithm RSA for RMALB/S problem
Begin:
  ns:= 0, *rt*:= 0, $T := T_0$;
  Generate a initial solution *S*;
  While (Termination criterion is not met) **do**
    NewBest:=0; // Check whether new best solution is obtained
    While (*ns*<*NS*)
      Obtain new solution *S*'with a neighborhood structure;
      Calculate $\Delta = f(S') - f(S)$; *ns*:=*ns*+1;
      If ($\Delta \leq 0$) $S \leftarrow S'$;
        If (New best solution is obtained) *NewBest*:=1;
      Else $S \leftarrow S'$ with a probability of $\exp^{-\Delta/(T \times f(S))}$;
    Endwhile
    If (*NewBest*>0) *rt*:=0; **Else** *rt*:=*rt*+1;
    If (*rt*>=*RT*)
      If(T< T_R) $T := T_R$;
      Else *T*:=*αT*;
    Else *T*:=*αT*;
  Endwhile

Figure.3 Procedure of the restarted simulated annealing algorithm

4.3 Co-evolutionary genetic algorithm

Co-evolutionary genetic algorithms have been widely applied to other MALB/S problems (Kim, Kim, and Kim 2000a; Kim, Kim, and Kim 2000b), which constitutes several sub-swarms, each addressing a sub-problem. Inspired by recent research on co-evolutionary algorithms by Li et al. (2016), this research develops a restarted co-evolutionary genetic algorithm (RCoGA). This algorithm makes two adjustments in employing vectors of the best solution rather than the best one of a sub-swarm for swam evaluation which will help to enhance the local search and application of a restart mechanism helps to escape from local optima. The outline of the proposed co-evolutionary

method is depicted in Figure.4.

```
Algorithm CA for RMALB/S problem
Begin:
  rt=0;
  Initialize three sub-swarms for three sub-problems;
  Select the best solution by testing the combination of the ith individuals from sub-swarms ;
  While (Termination criterion is not met) do
  For i=1,2,⋯,PS do  //Update task assignment
    Decode with ith individual from task assignment swarm and other two vectors from best solution;
    Utilize tournament selection, crossover and mutation operators to obtain new sub-swarm;
    Update best solution if new best solution is achieved;
    The last individual is replaced with the corresponding vector in the best solution;
  For i=1,2,⋯,PS do  //Update robot allocation
    Decode with ith individual from robot allocation swarm and other two vectors from best solution;
    Utilize tournament selection, crossover and mutation operators to obtain new sub-swarm;
    Update best solution if new best solution is achieved;
    The last individual is replaced with the corresponding vector in the best solution;
  For i=1,2,⋯,PS do  //Update model sequence
    Decode with ith individual from model sequence swarm and other two vectors from best solution;
    Utilize tournament selection, crossover and mutation operators to obtain new sub-swarm;
    Update best solution if new best solution is achieved;
    The last individual is replaced with the corresponding vector in the best solution;
  If (New best solution is obtained)  rt=0;
  Else rt:=rt+1;
  If (rt>=RT)          //Restart mechanism
    Execute restart mechanism and update the best solution;
  Endwhile
```

Figure.4 Procedure of the restarted co-evolutionary genetic algorithm

The proposed co-evolutionary genetic algorithm has four parameters: population size for each sub-swarm (*PS*), crossover rate, mutation rate and restart time before executing the restart mechanism (*RT*). This algorithm begins with initializing three sub-swarms and selecting the best solution from *PS* solutions achieved by combining the i^{th} individuals from the sub-swarms. Then the following steps are repeated in a cycle. Three sub-swarms are updated utilizing tournament selection, crossover and mutation operations successively, which are further explained in Section 4.4. Each individual is evaluated by combining with the other two vectors in the best solution. When a new best solution is achieved, the incumbent one is updated with the new one. After generating new offspring, the last one is replaced with the corresponding vector in the best solution. When a new best solution cannot be found for *RT* iterations, the restart mechanism is executed. A set of *PS* solutions are generated by modifying all three vectors simultaneously, each going through *Num_move*' neighborhood operations. Best among them is selected to replace the current best solution and subsequently the individuals of the three sub-swarms are replaced with the corresponding vector in the best solution after *Num_move*' neighborhood operation. With preliminary experiments, the value of *Num_move* is set to be 2. Further descriptions of neighborhood operation are presented in Section 4.4. Utilization of best solution serves as the elitist strategy to preserve the best individual in the offering, and restart mechanism aims at increasing the search space and escaping from local optima by modifying all the vectors simultaneously.

4.4 Neighborhood structures

Since three sub-problems are encoded with three vectors, this research develops different neighborhood operations for these vectors. Regarding task assignment vector, both swap and alteration operation are applied and these operations are randomly selected. For alteration operation, a task is randomly selected and its corresponding workstation is modified into another different workstation. It should be noted that neighborhood operation on the task assignment vector might lead to an infeasible task assignment violating precedence constraints. Due to this, this work utilizes the same repair mechanism as presented in Li, Tang, and Zhang (2016a) for robotic two-sided assembly line balancing problems. Regarding the robot allocation vector, swap operation and insert operation are both employed and they are randomly selected. As for model sequence, both swap and insert

operation are also applied and randomly selected. When applying these neighborhood operations in the proposed algorithms, RSA algorithm randomly selects a vector to be modified and RCoGA utilizes the neighborhood operation corresponding to the sub-problem.

For crossover operation, two-point crossover is applied for each vector. The aforementioned repair mechanism is also applied after executing crossover operations on the task assignment vector since an infeasible task assignment violating precedence constraints might be achieved.

4.5 Numerical example

A problem instance with 11 tasks, 4 workstations, 4 robots and 2 models is solved to illustrate the makespan calculation process. Precedence relationships and task operation times by robots for models are shown in Table 1. From this table, the operation times of the same task by robots for a same model differ from each other and the operation times of the same task by the same robot for two models are different from each other. Supposed that the demands for two models (model A and model B) in the MPS are 1 and 2 respectively, an optimal solution is depicted in Figure.5 and the corresponding model sequence is {B, B, A}.

Table 1 Precedence relationships and operation times of 11 tasks by 4 robots for two models

Task	Immediate successors	Model 1				Model 2			
		Robot 1	Robot 2	Robot 3	Robot 4	Robot 1	Robot 2	Robot 3	Robot 4
1	4	109	61	56	54	144	65	53	54
2	4, 5	47	28	24	17	60	28	32	19
3	11	42	32	46	26	37	34	35	29
4	6	57	40	80	53	63	46	64	53
5	7	83	40	51	43	90	41	47	35
6	8	103	50	111	27	91	56	102	29
7	9	145	96	74	42	168	75	85	35
8	10	122	32	125	51	148	36	124	38
9	10	107	22	61	24	129	28	51	24
10	11	59	27	61	97	58	33	72	92
11	-	76	24	40	18	72	33	28	20

The detailed makespan calculation is presented in Table 2 and plotted in Figure.6 with the Gantt chart. Tasks 1, 2 and 3 are assigned to workstation 1 and robot 3 is allocated to workstation 1, the total operation time of tasks on workstation 1 for model A is 56+24+46=126.

After achieving the workloads on workstations for models, completion time of all the tasks of the model in the s sequence on workstation k or C_{sk} is calculated. The tasks of the model in the s sequence on workstation k are able to be operated only when the tasks of the model in the s-1 sequence on workstation k and tasks of the model in the s sequence on workstation k-1 have finished. For instance, the completion time of all the tasks of the model A on workstation 2 is calculated with max(303, 366)+ 57=423. The makespan is the completion time of all the tasks of the last model in the model sequence (Model A) on workstation 4.

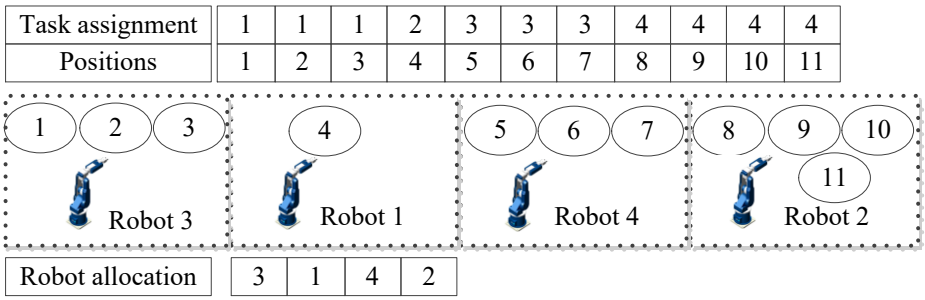


Figure.5 An example of task assignment and robot allocation

Table 2 Makespan calculation

Description	Workstations	1	2	3	4
Total operation times on stations for models	Model A	126	57	112	105
	Model B	120	63	99	130
The completion time of all the tasks of the model in the s sequence on workstation k	Model B	120	183	282	412
	Model B	240	303	402	542
	Model A	366	423	535	647
Makespan calculation	Makespan	647			

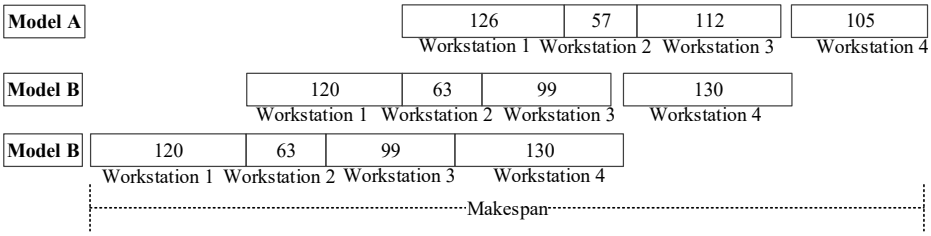


Figure.6 Gantt chart of the illustrated example

5. Computational study

This section explains the tested benchmark problems, the compared methods and the termination criterion. Since no research has been reported on RMALB/S prior to this, this paper generates nine sets of benchmarks based on the instance in Gao et al. (2009). The operation time of task i in model m by robot r is a random number within $[0.8 \times t_{ir}, 1.2 \times t_{ir}]$, where t_{ir} is the original published operation time of task i by robot r generated in Gao et al. (2009). The tested problem is listed in Table 3, where there are 104 cases in total.

Table 3 Description of tested cases

Problem	Nm	Nk	Demands in MPS
P11	2	2, 3, 4, 5	{1, 2}, {3, 1}
P25	2	3, 4, 6, 9	{1, 2}, {3, 1}
P35	2	4, 5, 7, 12	{1, 2}, {3, 1}
P53	3	5, 7, 10, 14	{1, 1, 1}, {3, 2, 1}, {1, 2, 4}
P70	3	7, 10, 14, 19	{1, 1, 1}, {3, 2, 1}, {1, 2, 4}
P89	4	8, 12, 16,21	{1, 1, 1, 1}, {1, 3, 4, 5}, {6, 4, 2,1}
P111	4	9, 13, 17, 22	{1, 1, 1, 1}, {1, 3, 4, 5}, {6, 4, 2,1}
P148	5	10, 14, 21, 29	{1, 1, 1, 1, 1}, {5, 3, 2, 1, 1}, {1, 2, 4, 5, 8}, {1, 4, 8, 3, 1}
P297	5	19, 29, 38, 50	{1, 1, 1, 1, 1}, {5, 3, 2, 1, 1}, {1, 2, 4, 5, 8}, {1, 4, 8, 3, 1}

This paper re-implements some recent and high-performing metaheuristic methods with proper adaptations. In addition, several variants of the proposed metaheuristic methods are also included to highlight the advantage of the improvements. All the tested algorithms are summarized in Table 4. In Table 4, the original simulated annealing algorithm or SA is selected to highlight the improvements. Artificial bee colony algorithm or ABC is included since it shows good performance in assembly line balancing problems (Tang, Li, and Zhang 2016) and its multi-objective edition has been successfully applied to MALB/S (Saif et al. 2014). Genetic algorithm or GA is the original algorithm for MALB/S (Kim, Kim, and Kim 2000a), and it is adopted to show the difference between with or without application of the co-evolution mechanism. Original co-evolutionary genetic algorithm or CoGA1 is applied to show whether the improvements on RCoGA is reasonable. Co-evolutionary genetic algorithm described in Section 4.3 without restart mechanism or CoGA2 is also employed to show the performance of restart mechanism.

Table 4 Summary and description of tested algorithms

Algorithms	Description
SA	No restart mechanism is applied.
RSA	This is the proposed restarted simulated annealing in Section 4.2.
ABC	One of the neighborhood operations in Section 4.4 is randomly selected for one individual in employee bee phase and onlooker phase. A scout is applied to replace the worst individual with a neighbor of a randomly selected individual generated by employing one of the neighborhood operations in Section 4.4 when no better global best solution is further achieved.
GA	The neighborhood structures in Section 4 are shared and tournament selection is applied. Elite strategy is applied by cloning the best individual to replace one of the offspring.
CoGA1	The same neighborhood structures in Section 4 are applied and tournament selection is applied. An individual in a sub-swarm is evaluated by combining with the best individuals of other two sub-swarms. No elite strategy or restart mechanism is applied.
CoGA2	It is the same to co-evolutionary genetic algorithm in Section 4.3 except for not applying the restart mechanism.
RCoGA	This is the proposed restarted co-evolutionary genetic algorithm in Section 4.3.

It is important to determine a proper termination criterion and this paper sets the termination criterion as a maximum elapsed CPU time following Li, Tang, and Zhang (2016a) and Li, Tang, and Zhang (2016b). The maximum elapsed CPU time is set to be equal to $Nt \times Nt \times \tau$ milliseconds, where it is tested at four levels (τ=10,20,30,40). This expression distributes more computational time to large-size problems and four termination criteria are able to analyze the performance of the algorithms from short CPU time to large CPU time. All the tested methods are coded in C++ and are executed on a cluster of personal computers with Microsoft Visual Studio 2012. All the computers are equipped with Intel(R) Core2(TM) CPU 2.33GHZ and 3.036 GB RMA.

5.2 Computational evaluation

This section presents the comparative study on the algorithms. As parameters play underlying role in the final performance of algorithms, this research calibrates the parameters of all tested algorithms at first using the full factorial design similar to the ones reported by Li, Tang, and Zhang (2016b) and Tang, Li, and Zhang (2016). A test problem with 111 tasks, 13 workstations and 4 models is utilized for the calibration and each configuration solves this large-size case for 10 times. The termination criterion is an elapsed CPU time of $Nt \times Nt \times 10$ milliseconds. The relative percentage increase (RPI) is employed as the response variable to measure the obtained results. RPI is calculated using Equation 16, where Fit_{some} is the objective function obtained by a configuration and Fit_{best} is the best fitness yielded by all the combinations.

$$Relative\ \ Percentage\ \ Increase\ \ (RPI) = 100 \times (Fit_{some} - Fit_{best})/Fit_{best} \tag{16}$$

The multifactor analysis of variance (ANOVA) technique (Montgomery 2008) is performed by considering the three parameters as factors as reported in Tang, Li, and Zhang (2016) after checking the fulfillment of three hypotheses, independence of residuals, homogeneity of the variance and normality of residuals. Detailed calibration results are omitted for space reason, but they are available upon request.

After calibrating the parameters, each algorithm solves each case 10 times and the best results within ten times' independent running for small-size problem are first compared with the optimal solution achieved by CPLEX solver in Table 5. Nk means the number of workstations and CPU refers to the computational time in seconds. The results of algorithms in Table 5 are obtained under the termination of an elapsed CPU time of $Nt \times Nt \times 10$ milliseconds. This table only shows the results for the P11 problem due to the high computational time when utilizing CPLEX solver for other problems. It is observed that all the optimal solutions for P11 are found by the two proposed algorithms, but with much less computational time. These results suggest that metaheuristic algorithms are more suitable for solving this problem.

Table 5 Best result comparison among algorithms and CPLEX solver

Problem	Nk	Demand in MPS	CPLEX		Algorithms							
			OPT	CPU (s)	SA	RSA	ABC	GA	COGA1	COGA2	RCOGA	CPU (s)
P11	2	{1,2}	1437	2	1437	1437	1437	1437	1437	1437	1437	1.2
P11	2	{3,1}	1778	2.1	1778	1778	1778	1778	1778	1778	1778	1.2
P11	3	{1,2}	870	6.1	870	870	870	870	870	870	870	1.2
P11	3	{3,1}	1114	8.8	1114	1114	1114	1114	1114	1114	1114	1.2
P11	4	{1,2}	647	23.2	647	647	647	647	647	647	647	1.2
P11	4	{3,1}	729	13.9	729	729	729	729	729	729	729	1.2
P11	5	{1,2}	553	88	553	553	553	553	553	553	553	1.2
P11	5	{3,1}	660	145.8	660	660	660	663	660	660	660	1.2

To further evaluate the two proposed algorithms, the average RPI values of all the cases are reported in Table 6. This table provides the average RPI values of problems, each containing several cases. For instance, each cell for P297 reports the average result of 16 cases generated by four workstation numbers and four different demands of the product models. From this table, the RSA is the best performer with the smallest average RPI of 5.58 under the termination of an elapsed CPU time of $Nt \times Nt \times 10$ milliseconds. Regarding the remaining three termination criteria, RSA is also the best performer with average RPI values of 2.84. 2.18 and 1.83 respectively. Among the remaining

algorithms, SA is the second best performer and RCoGA is the third best performer regarding all the four termination criteria.

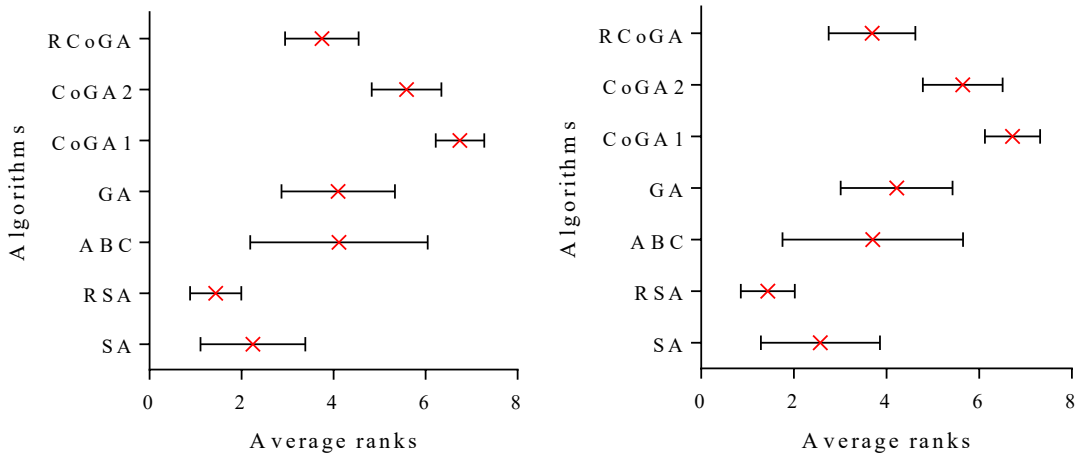
Table 6 Average RPI comparison among algorithms under four termination criteria

Problem	Workstation number	Average relative percentage increase							CPU time(s)
		SA	RSA	ABC	GA	CoGA1	CoGA2	RCoGA	
$\tau = 10$									
P11	2, 3, 4, 5	0.06	0.04	0.00	1.42	2.20	4.15	1.33	1.2
P25	3, 4, 6, 9	1.86	1.29	1.04	3.33	3.72	3.79	1.85	6.3
P35	4, 5, 7, 12	3.36	2.30	1.61	5.22	5.47	5.97	3.16	12.3
P53	5, 7, 10, 14	4.27	3.70	2.57	6.08	6.97	7.15	4.86	28.1
P70	7, 10, 14, 19	4.16	3.40	4.47	6.25	7.80	7.42	5.11	49.0
P89	8, 12, 16,21	4.05	4.07	4.60	6.08	8.06	6.68	5.46	79.2
P111	9, 13, 17, 22	4.42	4.35	9.38	7.19	11.90	9.44	8.06	123.2
P148	10, 14, 21, 29	7.18	7.33	16.45	10.16	21.12	15.35	11.89	219.0
P297	19, 29, 38, 50	15.28	15.47	21.28	14.12	27.84	19.20	15.69	882.1
Average RPI of all cases		5.81	**5.58**	8.43	7.46	12.42	9.93	7.44	
$\tau = 20$									-
P11	2, 3, 4, 5	0.06	0.03	0.00	1.42	2.20	3.91	0.82	2.4
P25	3, 4, 6, 9	1.86	0.91	0.82	3.23	3.69	3.73	1.56	12.5
P35	4, 5, 7, 12	3.36	1.61	1.40	4.99	5.39	5.72	2.35	24.5
P53	5, 7, 10, 14	4.27	2.92	2.00	5.82	6.83	7.05	3.74	56.2
P70	7, 10, 14, 19	4.15	2.63	3.16	5.82	7.18	7.17	3.81	98.0
P89	8, 12, 16,21	4.02	2.94	3.13	5.70	7.26	6.35	4.04	158.4
P111	9, 13, 17, 22	3.74	3.61	6.07	5.99	9.15	7.34	6.50	246.4
P148	10, 14, 21, 29	3.54	3.45	11.07	7.61	14.90	10.44	8.37	438.1
P297	19, 29, 38, 50	4.91	4.65	15.87	10.10	22.15	13.83	11.02	1764.2
Average RPI of all cases		3.57	**2.84**	5.97	6.16	10.08	7.98	5.43	-
$\tau = 30$									
P11	2, 3, 4, 5	0.06	0.03	0.00	1.42	2.20	3.91	0.63	3.6
P25	3, 4, 6, 9	1.86	0.74	0.72	2.94	3.66	3.69	1.45	18.8
P35	4, 5, 7, 12	3.36	1.24	1.29	4.88	5.30	5.57	2.11	36.8
P53	5, 7, 10, 14	4.27	2.56	1.85	5.80	6.78	7.03	3.23	84.3
P70	7, 10, 14, 19	4.15	2.18	2.60	5.58	6.96	7.07	3.17	147.0
P89	8, 12, 16,21	4.02	2.35	2.55	5.50	6.93	6.15	3.50	237.6
P111	9, 13, 17, 22	3.64	3.09	4.70	5.55	8.09	6.64	5.59	369.6
P148	10, 14, 21, 29	2.81	2.69	8.60	6.47	11.82	8.34	7.12	657.1
P297	19, 29, 38, 50	3.09	2.87	13.10	8.36	18.92	11.03	8.99	2646.3
Average RPI of all cases		3.17	**2.18**	4.84	5.58	8.91	7.10	4.59	-
$\tau = 40$									
P11	2, 3, 4, 5	0.06	0.03	0.00	1.42	2.20	3.91	0.63	4.8
P25	3, 4, 6, 9	1.86	0.67	0.60	2.89	3.59	3.69	1.21	25.0
P35	4, 5, 7, 12	3.36	0.99	1.23	4.83	5.21	5.52	1.89	49.0
P53	5, 7, 10, 14	4.27	2.28	1.77	5.72	6.77	6.97	2.92	112.4
P70	7, 10, 14, 19	4.15	1.83	2.26	5.48	6.84	7.00	2.80	196.0
P89	8, 12, 16,21	4.02	2.09	2.27	5.38	6.72	6.07	3.05	316.8
P111	9, 13, 17, 22	3.61	2.65	3.95	5.25	7.40	6.25	4.98	492.8
P148	10, 14, 21, 29	2.47	2.36	7.13	5.81	10.24	7.09	6.41	876.2
P297	19, 29, 38, 50	2.32	2.06	11.24	7.25	16.69	9.31	7.91	3528.4
Average RPI of all cases		2.99	**1.83**	4.15	5.23	8.19	6.57	4.07	-

RSA benefits from the restart mechanism by replacing the current temperature, and thus it has a stronger capacity to escape from local optima and outperforms the compared SA under almost all conditions. RCoGA benefits from two aspects: the application of the best solution for swarm evaluation and the utilization of restart mechanism. The application of best solution clones the best individual to offspring leads to fast convergence and strong search in the search space next to the best solution. The advantage of the application of the best solution for population evaluation is further proved by the superiority of CoGA2 over the CoGA1. Again, the restart mechanism avoids trapping RCoGA in local optima. Surprisingly, the GA outperforms CoGA1 and CoGA2. The reason is that the proposed GA employs an elite strategy. GA has two advantages: strong local search on the best individual and remained ability of exploring large search space achieved by modifying several vectors simultaneously. CoGA1 lacks the strong local search on the best individual since the best individual might not be preserved in the search space, and CoGA1 lacks exploration capacity and is trapped into local optima. Another interesting conclusion is related to ABC and GA algorithms. ABC

outperforms GA for small-size problems, whereas the GA outperforms ABC for large-size problems. The reason is that the GA utilizes crossover operation for population evolution and the applied ABC only propose local search methods and have no interchanges among the individuals.

To check whether the observed difference is statistically significant, non-parameter Friedman rank test (Friedman 1937) is applied since the normality of residuals is slightly violated. The Friedman rank test has been applied by Li, Tang, and Zhang (2016b) and Nilakantan et al. (2017) for comparing the performance of several algorithms. For Friedman rank test in this research, the best performer for each case is given a rank of 1 and the worst performer is given a rank of 7 since a total 7 algorithms are compared. Friedman rank analysis shows that the P-value is much lower than 0.01 and very close to 0.0 for each termination criterion, indicating that there is a significant statistical difference among the average ranks of the seven algorithms. This section presents the means plot of the algorithms in Figure.7. Figure.7 (a) and Figure. 7(b) illustrates the means plot with 95% minimal significant difference confidence intervals under elapsed CPU times of $Nt \times Nt \times 20$ milliseconds and $Nt \times Nt \times 40$ milliseconds.



(a) Average ranks under $\tau = 20$ (b) Average ranks under $\tau = 40$

Figure.7 Means plot of the ranks of the average RPI values with 95% confidence level

The statistical analysis results coincide with that presented in Table 6, and the RSA, SA and RCoGA are the three best performers. The span in RPI for the 10 problem instances appears to be the smallest. An interesting finding is that the gap between RCoGA and ABC is reduced with more computational time. All the computational results suggest that the improvements on SA and CoGA are reasonable and the RSA and RCoGA are effective for this new RMALB/S problem.

6. Conclusion and future research

This research presents the first method to address balancing and sequencing of robotic mixed-model assembly line (RMALB/S) problems simultaneously. This new RMALB/S contains three sub-problems: task assignment, robot allocation and model sequencing. A MILP model is presented with the objective of minimizing the makespan. CPLEX solver is applied to obtain optimal solutions for small-size problems. This model might receive interests from managers or decision-makers in industry, where robots are allocated to assemble kinds of products. This model helps managers to make the best of the available robots, reduce waste and ultimately improve the productivity and reduces cost.

To tackle large-size problems, this research proposes two metaheuristic algorithms, restarted simulated annealing (RSA) and restarted co-evolutionary genetic algorithm (RCoGA) with improvements. RSA replaces the current temperature with a new temperature to increase exploration capacity and RCoGA utilizes a restart mechanism to generate new population by modifying several vectors all together to avoid being trapped in local optima. To evaluate the proposed algorithms, a set of benchmark problems containing 104 cases are generated and solved. The computational results demonstrate the superiority of the improvements on the original simulated annealing algorithm and co-evolutionary genetic algorithm. Statistical analysis results prove that the proposed RSA and RCoGA are effective for solving RMALB/S problems and RSA is the best performer among the seven tested algorithms.

In future, constraint such as robot setup times can be incorporated to make the problem more realistic. More data from real-world factories are quite important to further enhance the proposed model. Meanwhile, one can consider other objectives such as cost and energy consumption

minimization. Multi-objective optimization algorithms are also interesting to be developed for RMALB/S problems. The model can be extended to other layouts of assembly lines, including U-type assembly lines and two-sided assembly lines. The proposed algorithms can also be extended to other scheduling problems that have several sub-problems to be addressed.

Acknowledgment
This research work is funded by the National Natural Science Foundation of China (Grant No. 51275366) (Qiuhua Tang).

References

Aghajani, Mojtaba, Reza Ghodsi, and Babak Javadi. 2014. "Balancing of robotic mixed-model two-sided assembly line with robot setup times." Review of. *The International Journal of Advanced Manufacturing Technology* 74 (5-8):1005-16.

Akpinar, Sener, and G Mirac Bayhan. 2014. "Performance evaluation of ant colony optimization-based solution strategies on the mixed-model assembly line balancing problem." Review of. *Engineering Optimization* 46 (6):842-62.

Battini, Daria, Maurizio Faccio, Alessandro Persona, and Fabio Sgarbossa. 2009. "Balancing–sequencing procedure for a mixed model assembly system in case of finite buffer capacity." Review of. *The International Journal of Advanced Manufacturing Technology* 44 (3):345-59. doi: 10.1007/s00170-008-1823-8.

Çil, Zeynel Abidin, Süleyman Mete, and Kürşad Ağpak. 2017. "Analysis of the type II robotic mixed-model assembly line balancing problem." Review of. *Engineering Optimization* 49 (6):990-1009.

Faccio, Maurizio, Mauro Gamberi, and Marco Bortolini. 2016. "Hierarchical approach for paced mixed-model assembly line balancing and sequencing with jolly operators." Review of. *International Journal of Production Research* 54 (3):761-77.

Friedman, Milton. 1937. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance." Review of. *Journal of the American Statistical Association* 32 (200):675-701. doi: 10.1080/01621459.1937.10503522.

Gao, Jie, Linyan Sun, Lihua Wang, and Mitsuo Gen. 2009. "An efficient approach for type II robotic assembly line balancing problems." Review of. *Computers & Industrial Engineering* 56 (3):1065-80. doi: http://dx.doi.org/10.1016/j.cie.2008.09.027.

Hamzadayi, Alper, and Gokalp Yildiz. 2012. "A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints." Review of. *Computers & Industrial Engineering* 62 (1):206-15. doi: http://dx.doi.org/10.1016/j.cie.2011.09.008.

———. 2013. "A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines." Review of. *Computers & Industrial Engineering* 66 (4):1070-84. doi: http://dx.doi.org/10.1016/j.cie.2013.08.008.

Kara, Yakup, Ugur Ozcan, and Ahmet Peker. 2007a. "An approach for balancing and sequencing mixed-model JIT U-lines." Review of. *The International Journal of Advanced Manufacturing Technology* 32 (11-12):1218-31.

———. 2007b. "Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives." Review of. *Applied Mathematics and Computation* 184 (2):566-88.

Kim, Yeo Keun, Jae Yun Kim, and Yeongho Kim. 2000a. "A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines." Review of. *Applied Intelligence* 13 (3):247-58.

———. 2006. "An endosymbiotic evolutionary algorithm for the integration of balancing and

sequencing in mixed-model U-lines." Review of. *European Journal of Operational Research* 168 (3):838-52.

Kim, Yeo Keun, Sun Jin Kim, and Jae Yun Kim. 2000b. "Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm." Review of. *Production Planning & Control* 11 (8):754-64.

Kucukkoc, Ibrahim, and David Z. Zhang. 2014a. "Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines." Review of. *International Journal of Production Economics* 158:314-33. doi: http://dx.doi.org/10.1016/j.ijpe.2014.08.010.

———. 2014b. "Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines." Review of. *International Journal of Production Research* 52 (12):3665-87. doi: 10.1080/00207543.2013.879618.

———. 2016. "Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines." Review of. *The International Journal of Advanced Manufacturing Technology* 82 (1):265-85. doi: 10.1007/s00170-015-7320-y.

Levitin, Gregory, Jacob Rubinovitz, and Boris Shnits. 2006. "A genetic algorithm for robotic assembly line balancing." Review of. *European Journal of Operational Research* 168 (3):811-25. doi: http://dx.doi.org/10.1016/j.ejor.2004.07.030.

Li, Z., M. N. Janardhanan, Q. Tang, and P. Nielsen. 2016. "Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem." Review of. *Advances in Mechanical Engineering* 8 (9):14. doi: 10.1177/1687814016667907.

Li, Zixiang, Qiuhua Tang, and LiPing Zhang. 2016a. "Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm." Review of. *Journal of Cleaner Production* 135:508-22. doi: http://dx.doi.org/10.1016/j.jclepro.2016.06.131.

———. 2016b. "Minimizing the Cycle Time in Two-Sided Assembly Lines with Assignment Restrictions: Improvements and a Simple Algorithm." Review of. *Mathematical Problems in Engineering* 2016 (Article ID 4536426):1-15. doi: 10.1155/2016/4536426.

Liu, Qiong, Wen-xi Wang, Ke-ren Zhu, Chao-yong Zhang, and Yun-qing Rao. 2014. "Advanced scatter search approach and its application in a sequencing problem of mixed-model assembly lines in a case company." Review of. *Engineering Optimization* 46 (11):1485-500.

Manavizadeh, Neda, Masoud Rabbani, and Farzad Radmehr. 2015. "A new multi-objective approach in order to balancing and sequencing U-shaped mixed model assembly line problem: a proposed heuristic algorithm." Review of. *The International Journal of Advanced Manufacturing Technology* 79 (1):415-25. doi: 10.1007/s00170-015-6841-8.

Montgomery, Douglas C. 2008. *Design and analysis of experiments*: John Wiley & Sons.

Mosadegh, H., M. Zandieh, and S. M. T. Fatemi Ghomi. 2012. "Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines." Review of. *Applied Soft Computing* 12 (4):1359-70. doi: http://dx.doi.org/10.1016/j.asoc.2011.11.027.

Mukund Nilakantan, J., George Q. Huang, and S. G. Ponnambalam. 2015. "An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems." Review of. *Journal of Cleaner Production* 90:311-25. doi: http://dx.doi.org/10.1016/j.jclepro.2014.11.041.

Mukund Nilakantan, J., and S. G. Ponnambalam. 2016. "Robotic U-shaped assembly line balancing using particle swarm optimization." Review of. *Engineering Optimization* 48 (2):231-52. doi: 10.1080/0305215X.2014.998664.

Mukund Nilakantan, J., S. G. Ponnambalam, N. Jawahar, and G. Kanagaraj. 2015. "Bio-inspired search

algorithms to solve robotic assembly line balancing problems." Review of. *Neural Computing and Applications* 26 (6):1379-93. doi: 10.1007/s00521-014-1811-x.

Nilakantan, J Mukund, Zixiang Li, Qiuhua Tang, and Peter Nielsen. 2017. "Multi-objective co-operative co-evolutionary algorithm for minimizing carbon footprint and maximizing line efficiency in robotic assembly line systems." Review of. Journal of Cleaner Production 156:124-36.

Özcan, Uğur, Hakan Çerçioğlu, Hadi Gökçen, and Bilal Toklu. 2010. "Balancing and sequencing of parallel mixed-model assembly lines." Review of. International Journal of Production Research 48 (17):5089-113. doi: 10.1080/00207540903055735.

Öztürk, Cemalettin, Semra Tunalı, Brahim Hnich, and M. Arslan Örnek. 2013. "Balancing and scheduling of flexible mixed model assembly lines." Review of. Constraints 18 (3):434-69. doi: 10.1007/s10601-013-9142-6.

Rabbani, Masoud, Zahra Mousavi, and Hamed Farrokhi-Asl. 2016. "Multi-objective metaheuristics for solving a type II robotic mixed-model assembly line balancing problem." Review of. Journal of Industrial and Production Engineering 33 (7):472-84. doi: 10.1080/21681015.2015.1126656.

Rabbani, Masoud, Shadi Sadri, Neda Manavizadeh, and Hamed Rafiei. 2015. "A novel bi-level hierarchy towards available-to-promise in mixed-model assembly line sequencing problems." Review of. Engineering Optimization 47 (7):947-62.

Rubinovitz, Jacob, and Joseph Bukchin. 1991. Design and balancing of robotic assembly lines: Society of Manufacturing Engineers.

Rubinovitz, Jacob, Joseph Bukchin, and Ehud Lenz. 1993. "RALB – A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines." Review of. CIRP Annals - Manufacturing Technology 42 (1):497-500. doi: http://dx.doi.org/10.1016/S0007-8506(07)62494-9.

Ruiz, Rubén, Concepción Maroto, and Javier Alcaraz. 2006. "Two new robust genetic algorithms for the flowshop scheduling problem." Review of. Omega 34 (5):461-76.

Ruiz, Rubén, and Thomas Stützle. 2007. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem." Review of. European Journal of Operational Research 177 (3):2033-49.

Saif, Ullah, Zailin Guan, Weiqi Liu, Baoxi Wang, and Chaoyong Zhang. 2014. "Multi-objective artificial bee colony algorithm for simultaneous sequencing and balancing of mixed model assembly line." Review of. The International Journal of Advanced Manufacturing Technology 75 (9-12):1809-27.

Sivasankaran, P, and P Shahabudeen. 2014. "Literature review of assembly line balancing problems." Review of. The International Journal of Advanced Manufacturing Technology 73 (9-12):1665-94.

Tang, Qiuhua, Zixiang Li, and Liping Zhang. 2016. "An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-II." Review of. Computers & Industrial Engineering 97:146-56. doi: http://dx.doi.org/10.1016/j.cie.2016.05.004.

Yoosefelahi, A., M. Aminnayeri, H. Mosadegh, and H. Davari Ardakani. 2012. "Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model." Review of. Journal of Manufacturing Systems 31 (2):139-51. doi: http://dx.doi.org/10.1016/j.jmsy.2011.10.002.