Analysis of the type II robotic mixed-model assembly line balancing problem

Zeynel Abidin Çil, Süleyman Mete and Kürşad Ağpak

Department of Manufacturing Engineering, University of Batman, Batman, Turkey; Department of Industrial Engineering, Gaziantep University, Gaziantep, Turkey

ABSTRACT

In recent years, there has been an increasing trend towards using robots in production systems. Robots are used in different areas such as packaging, transportation, loading/unloading and especially assembly lines. One important step in taking advantage of robots on the assembly line is considering them while balancing the line. On the other hand, market conditions have increased the importance of mixed-model assembly lines. Therefore, in this article, the robotic mixed-model assembly line balancing problem is studied. The aim of this study is to develop a new efficient heuristic algorithm based on beam search in order to minimize the sum of cycle times over all models. In addition, mathematical models of the problem are presented for comparison. The proposed heuristic is tested on benchmark problems and compared with the optimal solutions. The results show that the algorithm is very competitive and is a promising tool for further research.

1. Introduction

An assembly line system is an arrangement of machines, employees and equipment where the product being assembled passes successively from operation to operation until it is completed. Assembly lines can be automated, manually operated or of mixed design. The increase in variety of demands has resulted in a need for more flexible and efficient systems. In this regard, robots can enhance the flexibility and efficiency of the system. An assembly line that uses robots is called a robotic assembly line (RAL). Figure 1 illustrates an example of an RAL. Such lines are used especially in the automotive industry. The advantages of using robots are summarized by Nilakantan and Ponnambalam (2016) as improving productivity, quality of product, manufacturing flexibility and safety, and reducing the need for skilled labour. A general-purpose assembly robot can perform different tasks using certain tools, but different types of robot can complete a task in different performance times. The allocation of suitable robots in the station is critical for RAL. Therefore, the robotic assembly line balancing (RALB) problem has two subproblems: assigning tasks to the stations and allocating robots to the stations.

The mathematical formulation of the assembly line balancing (ALB) problem was introduced by Salveson (1955). Since then, various optimization models and heuristic/metaheuristic methods have...
been developed to solve the problem (Baybars 1986; Scholl and Becker 2006; Battaia and Dolgui 2013). ALB problems can be classified into two groups, named type I and type II, based on the objective function. This classification is also used for RALB problems by Gao et al. (2009). The aim of the type I robotic assembly line balancing problem (RALBP-I) is to minimize the number of stations under the determined cycle time, whereas the type II robotic assembly line balancing problem (RALBP-II) aims to minimize the cycle time for a fixed number of stations.

The RALB problem was first described by Rubinovitz, Bukchin, and Lenz (1993). They define the problem in detail and discuss the advantages of RALs in production environments. The frontier search method, based on the branch-and-bound algorithm, has been developed to solve RALBP-I. Besides the classical RAL problem, Kim and Park (1995) consider assigning parts and tools to stations. Space and joint assignment constraints are also taken into account in defining the problem. A mathematical model and cutting plane algorithm are proposed to solve the problem with the objective of minimizing the total number of robot cells. A genetic algorithm (GA) (Levitin, Rubinovitz, and Shnits 2006), an improved GA with combined local search algorithm (Gao et al. 2009) and a particle swarm optimization (PSO) algorithm (Nilakantan and Ponnambalam 2012) have been developed to solve RALBP-II on straight lines. Finally, for the same problem, Nilakantan et al. (2015) present a PSO algorithm and a hybrid cuckoo search (CS–PSO algorithm and compare them with the GA of Gao et al. (2009). The CS-PSO algorithm outperforms the other two algorithms, PSO and GA, based on the results of solved test problems. Yoosefelahi et al. (2012) study the RALB problem with three objectives: the minimization of cycle time, robot set-up cost and total robot cost. They present a new mixed-integer linear programming model and evolution strategies to solve the problem. Daoud et al. (2014) address an industrial application of RALB. The system includes certain pick-and-place robots. Several metaheuristic algorithms are proposed to maximize the line efficiency of the system and to discover the best and most balanced distribution of components and location points for each robot. Nilakantan, Huang, and Ponnambalam (2015) consider the minimization of cycle time and total energy consumption together. The primary objective is to minimize cycle time and the secondary objective is to minimize total energy consumption. A PSO algorithm is proposed to solve the multi-objective problem. Nilakantan and Ponnambalam (2016) consider a U-shaped RALB problem for the first time in the literature. A mathematical model is presented to define the problem and a PSO algorithm is suggested for solving the problem. Nilakantan, Ponnambalam, and Jawahar (2016) also study the U-shaped RALB problem to minimize to total energy consumption. They propose differential evolution and PSO algorithms to solve the problem. In terms of total energy consumption, PSO outperforms differential evolution for almost all problems. Nilakantan et al. (2016) consider straight and U-line RALB problem with a cost-based approach. The objective of the study is to minimize cost and cycle time. A differential evolution algorithm is proposed to solve the problem.

All of the above-mentioned articles are considered for the single-model RALB problem. A single-model assembly line is used to produce a high volume of a specific product. This type of line does not allow flexibility for customized products or variation in demand for product models. Under today’s demanding market conditions, an increase in the variety of products has resulted in new and more
complex problems. Economic opportunities can be gained by manufacturing more than one type of product on the same assembly line. Mixed-model assembly lines allow the manufacture of products that are in a group of similar models of items. These models can be produced in different numbers according to customer demand. A lot of articles in the literature have focused on the mixed-model ALB problem (Gökçen and Erel 1998; Simaria and Vilarinho 2004; Kara, Ozcan, and Peker 2007; Simaria and Vilarinho 2009; Akpınar and Bayhan 2011; Battaiá et al. 2015; Buyukozkan et al. 2016). On the other hand, to the authors’ knowledge, only two articles have considered the robotic mixed-model assembly line balancing (RMALB) problem. In the first article, Aghajani, Ghodsi, and Javadi (2014) present a two-sided RMALB problem. Robot set-up and sequence-dependent set-up times are taken into account simultaneously. A mathematical model is developed to minimize the sum of cycle times over all models. In addition, a simulated annealing algorithm is proposed for large-sized test problems. In the second article, Rabbani, Mousavi, and Farrokhi-Asl (2016) study a U-type RMALB problem with objective of minimizing robot purchasing costs, robot set-up costs, sequence-dependent set-up costs and cycle times. Tasks on the line are handled by two groups: (1) the special tasks for one model, and (2) the common tasks for several models. In addition, it is assumed that two or more robots can perform the work at the same station. A summary of the literature on RALB problems is given in Table 1. These studies are classified according to the number of models, line type, objectives and solution technique.

Table 1. Summary of the literature on the robotic assembly line balancing problem.

| Study | No. of models | Line type | Objectives | Heuristic technique |
|-------|---------------|-----------|------------|---------------------|
| Rubinovitz, Bukchin, and Lenz (1993) | Single | Straight | Min. no. of stations | Frontier search method |
| Kim and Park (1995) | Single | Straight | Min. no. of stations | Cutting plane algorithm |
| Levitin, Rubinovitz, and Shnits (2006) | Single | Straight | Min. cycle time | Genetic algorithm |
| Gao et al. (2009) | Single | Straight | Min. cycle time | Hybrid genetic algorithm |
| Nilakantan and Ponnambalam (2012) | Single | Straight | Min. cycle time | Particle swarm optimization |
| Yoosefelahi et al. (2012) | Single | Straight | Min. cycle time, robot set-up cost and robot cost | Evaluation algorithm |
| Aghajani, Ghodsi, and Javadi (2014) | Mixed | Two-sided | Min. sum of cycle times | Simulated annealing |
| Daoud et al. (2014) | Single | Straight | Max. line efficiency | Ant colony algorithm, particle swarm optimization and genetic algorithm |
| Nilakantan, Huang, and Ponnambalam (2015) | Single | Straight | Min. cycle time and total energy consumption | Particle swarm optimization |
| Nilakantan et al. (2015) | Single | Straight | Min. cycle time | Cuckoo search–particle swarm optimization |
| Nilakantan and Ponnambalam (2016) | Single | U-line | Min. cycle time | Particle swarm optimization |
| Rabbani, Mousavi, and Farrokhi-Asl (2016) | Mixed | U-line | Min. sum of cycle times, robot cost, robot set-up cost, sequence-dependent set-up cost | Genetic algorithm and particle swarm optimization |
| Nilakantan, Ponnambalam, and Jawahar (2016) | Single | U-line | Min. total energy consumption | Evaluation algorithm |
| Nilakantan et al. (2016) | Single | Straight and U-line | Min. cost and cycle time | Differential evolution algorithm |
and Aghajani, Ghodsi, and Javadi (2014) are modified to define the RMALB-II problem mathematically. Secondly, the modified formulations are compared with each other to show their computational efficiency. Thirdly, since the complexity of the problem also increases with the number of models and tasks, a metaheuristic/heuristic approach is essential to solve large-sized test problems. Hence, a beam search (BS)-based approach is developed to solve RMALB-II problem instances. Lastly, several versions of the BS algorithm are developed using different priority rules at the local search stage. The aim of the proposed different versions is to evaluate the effects of the local search strategy on the performance of the BS algorithm. As far as the authors know, this is the first study to apply a BS approach to the RMALB-II problem.

The rest of the article is arranged as follows. In Section 2, the problem definition and mathematical models of the RMALB-II problem are given. The proposed BS algorithm is presented in Section 3. In Section 4, performances of the algorithms and models are tested on the newly generated benchmark problems and the results of the heuristic are compared with the optimal solution found with mixed-integer programming models. In Section 5, conclusions and future research directions are offered.

2. Problem definition

In classical ALB problems, a product has a group of particular tasks that have predetermined precedence relations and task times. The main purpose is to minimize or maximize the objective function by way of assigning tasks to stations (Scholl, Boysen, and Fliedner 2013). On an RAL, each task can be performed by a group of robots in particular times under precedence constraints to manufacture a product. The operation time of a task can be different for each robot. The product is transferred from station to station for the performance of tasks, using a conveyor belt. At each station, only a robot is responsible for completing the task. While the task is assigned to a station, the set of tasks at that station should not exceed the cycle time and precedence relations must be obeyed. Assumptions made for solving the RMALB-II problem are given as follows (Levitin, Rubinovitz, and Shnits 2006; Yoosefela hi et al. 2012):

- Processing times are constant and known. The times may vary according to the type of robot.
- Precedence diagrams are known and combined into a single precedence diagram.
- Division of tasks is not allowed.
- If there is a precedence relation between task \( i \) and task \( k \), task \( k \) must not be assigned to a station earlier than the station that is to perform task \( i \).
- Only a robot can perform the task at each station.
- All robots are offered without any limitations.
- The set-up times and material transportation times are negligible.
- The number of models is more than one (\( m > 1 \))
2.1. Notation

(i, k) Task (i = 1, . . . , I, k = 1, . . . , K)
(r) Robot (r = 1, . . . , R)
(m) Models (m = 1, . . . , M)
(j) Station (j = 1, . . . , J)
(t_{imr}) Operation time of task i for model m and robot r
(C_m) Cycle time of model m
(P) Set of pairs of tasks (i, k) in which task i is the immediate predecessor of task k
(P_a) Set of pairs of tasks (i, k) in which task i is an element of all predecessors of task k
(B) A large number
(A_{jr}) 1, if robot r is assigned to station j; 0, otherwise.

Model 1 variables
(x_{ijr}) 1, if task i is assigned to station j and performed by robot r; 0, otherwise.

Model 2 variables
(Z_{ik}) 1, if task i is assigned before task k in the same station; 0, if task k is assigned before task i in the same station.
(Y_{ij}) 1, if task i is assigned to station j; 0, otherwise.

The first mixed-integer linear programming model of the problem is a modified model of Yooselahi et al. (2012) and is for single-model RALB-II problems. The model is presented as follows.

Mathematical Model 1:

Objective

\[ \text{Min Obj} = \sum_{m=1}^{M} C_m \]  

Constraints

\[ \sum_{j=1}^{J} \sum_{r=1}^{R} x_{ijr} = 1 \quad \text{for } i = 1, \ldots, I \]  

\[ \sum_{j=1}^{J} \sum_{r=1}^{R} j * x_{ijr} \leq \sum_{j=1}^{J} \sum_{r=1}^{R} k * x_{kjr} \quad \text{for all } (i, k) \in P \]  

\[ \sum_{j=1}^{J} \sum_{r=1}^{R} t_{imr} * x_{ijr} \leq C_m \quad \text{for } j = 1, \ldots, J, \text{ and } m = 1, \ldots, M \]  

\[ \sum_{r=1}^{R} A_{jr} = 1 \quad \text{for } j = 1, \ldots, J \]  

\[ \sum_{j=1}^{J} \sum_{r=1}^{R} A_{jr} \leq J \]  

\[ \sum_{i=1}^{I} x_{ijr} \leq B * A_{jr} \quad \text{for } j = 1, \ldots, J, r = 1, \ldots, R \]  

\[ x_{ijr}, A_{jr} \in \{0,1\} \]  

The objective function (1) is defined to minimize the sum of cycle times over all models. The assignment Constraint (2) ensures that each task can be assigned to only one station and can be
performed by one robot. Constraint (3) provides that if there is a precedence relation between tasks, the successor task cannot be assigned before the predecessor task. Constraint (4) ensures that the total processing time in each station cannot exceed the cycle time for each model. Constraint (5) means that a robot must be assigned to a station. Constraint (6) provides that the total number of robots used cannot be greater than the predetermined stations. Constraint (7) ensures that if a task is assigned to station \(j\) and performed by robot \(r\), then robot \(r\) is assigned to station \(j\). Constraint (8) indicates binary variables.

The second mixed-integer linear programming model is derived from the model of Aghajani, Ghodsi, and Javadi (2014) and is for two-sided RMALB problems. Also, Constraint (14) is added as a new constraint in the model. The complete model is presented as follows.

**Mathematical Model 2:**

**Objective**

\[
\text{Min Obj} = \sum_{m=1}^{M} C_m
\]  
(9)

**Constraints**

\[
\sum_{j=1}^{J} Y_{ij} = 1 \quad \text{for } i = 1, \ldots, I
\]  
(10)

\[
\sum_{j=1}^{j} j.Y_{ij} \leq \sum_{j=1}^{j} j.Y_{kj} \quad \text{for all } (i, k) \in P
\]  
(11)

\[t_{fim} \leq C_m \quad \text{for } i = 1, \ldots, I \text{ and } m = 1, \ldots, M
\]  
(12)

\[t_{fkm} - t_{fim} + B * (1 - Y_{ij}) + B * (1 - Y_{kj}) \geq \sum_{r=1}^{R} t_{kmr} * A_{jr}
\]
for \( j = 1, \ldots, J, m = 1, \ldots, M, \text{ and } (i, k) \in P \)
(13)

\[t_{fim} + B * (1 - Y_{ij}) + B * (1 - A_{jr}) \geq t_{imr} * A_{jr}
\]
for \( i = 1, \ldots, I, \text{ and } m = 1, \ldots, M, j = 1, \ldots, J, \text{ and } r = 1, \ldots, R \)
(14)

\[t_{fkm} - t_{fim} + B * (1 - Y_{ij}) + B * (1 - Y_{kj}) + B * (1 - Z_{ik}) \geq \sum_{r=1}^{R} t_{kmr} * A_{jr}
\]
for \( j = 1, \ldots, J, m = 1, \ldots, M, \text{ and } (i, k) \notin P_a, i < k \)
(15)

\[t_{fim} - t_{fkm} + B * (1 - Y_{ij}) + B * (1 - Y_{kj}) + B * (Z_{ik}) \geq \sum_{r=1}^{R} t_{imr} * A_{jr}
\]
for \( j = 1, \ldots, J, \text{ and } m = 1, \ldots, M, \text{ and } (i, k) \notin P_a, i < k \)
(16)

\[\sum_{r=1}^{R} A_{jr} = 1 \quad \text{for } j = 1, \ldots, J
\]  
(17)

\[Y_{ij}, A_{jr} \in \{0, 1\}
\]  
(18)

The objective function is defined in Equation (9) to minimize the sum of cycle times over all models. Constraint (10) is the assignment constraint that ensures that each task is assigned to a station.
Constraint (11) provides the precedence relations between tasks. Constraint (12) ensures that the finish time of each task cannot exceed the cycle time for each model. If task \( k \) is the immediate successor of task \( i \) and they are assigned to the same station, Constraint (13) becomes active and ensures that the start time of task \( k \) must be greater than or equal to the finish time of task \( i \) for each model. Constraint (14) becomes active when robot \( r \) and task \( i \) are assigned to the same station and it provides that the finish time of task \( i \) should be greater than the processing time of task \( i \) for each model. If there is no precedence relation between tasks \( i \) and \( k \), and these are assigned to the same station, Constraints (15) and (16) become active. In this case, tasks can be done in any sequence. If \( Z_{ik} \) takes a value of 1, constraint (15) becomes \( t_{fkm} - t_{fim} \geq \sum_{r=1}^{R} t_{kmr} \ast A_{jr} \) and task \( i \) finishes before the start of task \( k \); otherwise, Constraint (16) becomes \( t_{fim} - t_{fkm} \geq \sum_{r=1}^{R} t_{imr} \ast A_{jr} \) and task \( k \) is done first. Constraint (17) ensures that only a robot can be assigned to a station. Constraint (18) indicates binary variables.

The two models are compared in Section 4. Owing to the NP-hard structure of the problem, it is expected that limited-size problems can be solved with the mathematical models in an acceptable time (Nilakantan et al. 2015). Therefore, heuristic algorithms are necessary for large-sized problems. In this study, the BS algorithm is presented in the next section.

### 3. Beam search algorithm

The BS algorithm is used in many types of problem in the literature, from artificial intelligence to optimization problems. The algorithm was first defined by Lowerre (1976) and applied artificial intelligence. Moreover, this technique is used in several types of ALB and scheduling problems in the literature, including stochastic ALB (Erel, Sabuncuoğlu, and Sekerci 2005), ALB with time and space constraints (Blum, Bautista, and Pereira 2006), simple ALB (Blum 2008), assembly line scheduling (Sabuncuoglu, Gocgun, and Erel 2008) and assembly line worker assignment and balancing (ALWAB) problems (Blum and Miralles 2011; Borba and Ritt 2014). Several enhancement tools and hybridization of local search methods are used to improve the performance of the BS. For example, the BS algorithm has been combined with ant colony optimization, increasing its performance efficiently and effectively (Blum, Bautista, and Pereira 2006), and a backtracking method has been developed for the multi-objective flow-shop problem (Honda 2003). Certain improvement methods which are related to assembly lines are summarized in Table 2.

In this article, the BS algorithm is selected as a solution approach for several reasons. First, BS is an adaption of the branch-and-bound method in terms of structure. Branch-and-bound methods check all solutions with a backtracking technique. In contrast, the BS algorithm searches a solution just on determined points in a forward direction to speed up the solution search. Secondly, the iterative structure of the algorithm can be advantageous because the assignment of tasks and robots is considered together in the problem. Last, the algorithm provides efficient results in simple ALB and ALWAB problems. Thus, the results motivate the application of the BS algorithm to RMALB

### Table 2. Literature on beam search algorithms.

| Study                  | Improvement method                             | Application area                |
|------------------------|-----------------------------------------------|---------------------------------|
| Leu (1997)             | Filtered                                      | Assembly line sequencing        |
| Erel, Sabuncuoğlu, and Sekerci (2005) | –                                             | Assembly line balancing         |
| McMullen and Tarasewich (2005) | –                                             | Assembly line sequencing        |
| Erel, Göçgun, and Sabuncuoğlu (2007) | Enhanced with exchange of information and backtracking method | Assembly line sequencing        |
| Sabuncuoğlu, Göçgun, and Erel (2008) | Enhanced with backtracking                    | Assembly line scheduling        |
| Blum (2008)            | Hybridized with ant colony optimization        | Assembly line balancing         |
| Yavuz (2010)           | Enhanced with dynamic programming              | Production smoothing problem    |
| Blum and Miralles (2011)| Filtered                                      | Assembly line balancing         |
| Baldi et al. (2014)    | –                                             | Bin packing problem             |
| Golle, Rothlauf, and Boysen (2015) | –                                             | Car sequencing                  |
problems. Considering all these reasons, the BS algorithm is one of the best heuristic approaches to solve the problem.

BS completes a solution from top to bottom, selecting the best point within each level in a beam tree. The algorithm can hold at most $\beta$ (beam width) partial solutions. At each level of the beam tree, partial solutions are extended with a local search strategy. Then, the partial solution is selected to transfer to the next stage according to the fitness function at the most beam extension ($\beta_{ext}$) feasible solution. The algorithm completes its search when all of the tasks have been assigned. The speed and intelligence of the algorithm are its strengths and certain critical parameters can affect these strengths. For example, $\beta_{ext}$ and beam width, $\beta$, parameters can affect directly the central processing unit (CPU) time. The fastest solution can be found when these parameters take a value of 1, but it leads to the loss of the best solution to the problem. On the other hand, extension of the method of partial solutions can be crucial to select the best beam owing to its variability (Yavuz 2010). In this study, six variants of the method are presented (see Section 3.1). The pseudo-code of the proposed algorithm scheme is given in Algorithm 1.

The precedence diagram, $P$, the predetermined station, $W$, task time, $T$, beam width, $\beta$, and the maximal number of feasible solutions, $\beta_{ext}$, are needed as inputs to solve the problem. First, the initial cycle time values are determined using Equation (19) (Levitin, Rubinovitz, and Shnits 2006). If the duration of any task is more than the initial cycle time, the duration of the task is accepted as the initial cycle time.

$$C_m = \sum_{i=1}^{I} \min_{i \in R} (T_{imr}) \quad \forall m$$ (19)

**Algorithm 1: The proposed algorithm scheme (PAS)**

1. **Input:** $(P, W, T, \beta, \beta_{ext})$
2. $C_m \leftarrow$ DetermineInitialCycleTime $(T, W)$
3. $[C_m] \leftarrow$ PriorityBasedHeuristic $(P, T, W, C_m)$
4. for each model
5.      $C_m = C_m - 1$
6.      while timelimit $\leq$ 300 second do
7.          $ws \leftarrow$ BeamSearchAlgorithm $(P, W, T, \beta, \beta_{ext}, C_m)$
8.          if $ws > W$ do
9.              break
10.         else do
11.              $C_m = C_m - 1$
12.          end if
13.         end while
14.     $C_m = C_m + 1$
15. end for
16. **output:** $C_m$

Then, a priority-based heuristic algorithm, PriorityBasedHeuristic $(P, T, W, C_m)$, is used to find a good upper bound for cycle time in a short time. The maximum total followers method is selected as a heuristic because it provides a feasible solution in a very short time, less than 1 s for all of the problems in the pre-test. Moreover, it offers good results in a simple type II ALB problem (Scholl and Voβ 1996). This method selects the task from the assignable tasks list according to the total number of followers. The steps of the method are given below:

Step 1: Input data, $P, T, W, C_m$.
Step 2: Start assignment process.
Step 3: Determine assignable list based on P and go to Step 4.
Step 4: If assignable list is empty, go to Step 7; otherwise go to Step 5.
Step 5: Select a task from assignable list according to total number of maximum followers. If certain tasks have equal numbers of maximum followers, a task is selected randomly between these tasks.
   Step 5.1: Assign selected task to station. If remaining station time is less than task time for all robots, go to Step 6
   Step 5.2: Compute remaining time of station for all robots.
   Step 5.3: Return to Step 3.
Step 6: Open a new station and return to Step 5.1.
Step 7: If the number of stations, ws, is equal to predetermined number of the stations, W, go to Step 9, else go Step 8.
Step 8: Increase cycle time, \( C_m = C_m + 1 \), return to Step 2.
Step 9: Output cycle time, \( C_m \).

After finding a feasible solution with the priority-based algorithm, BS with \( \beta \) and \( \beta_{ext} \) searches new feasible solutions in a smaller cycle time. Although higher parameter values obviously take more computational time, they have a higher chance of obtaining feasible solutions in a smaller cycle time (Blum and Miralles 2011). Hence, the optimization of \( \beta \) and \( \beta_{ext} \) parameters is crucial. The pseudocode of the BS algorithm is given in Algorithm 2.

**Algorithm 2: The proposed beam search algorithm (BSA)**

1. **Input:** \((P, W, C_m, T, \beta, \beta_{ext})\)
2. \(PS = \emptyset\)
3. \(\text{BExtending} = \emptyset, \text{BExtending} = \emptyset, \text{GlobObj} = \emptyset\)
4. \(l = 0\)
5. \(PS \leftarrow \text{GenerateRootNodes}(P, l)\)
6. \(l = l + 1;\)
7. \(\text{aslist} = \text{alltask}\)
8. **while** True **do**
9. \([tn, PS] \leftarrow \text{GenerateDescendantNodes}(PS, P, l)\)
10. **if** descendant nodes are generated for each nodes current \(l\) **do**
11. **if** \(tn \geq \beta\)
12. **for** \(i = 1, \ldots, tn\)
13. \([\text{BExtending}_i, FV_i, \text{AsR}_w] \leftarrow \text{ExtendPartialSolution}(PS, T, l, P)\)
14. **if** \(FV_i\) is equal to \(W\) **do**
15. **return** output \(ws = FV_i\)
16. **end if**
17. **end for**
18. \([PS] \leftarrow \text{ChoosingBestSolutions}(FV_i, \text{BExtending}_i, \beta_{ext})\)
19. \(\text{aslist} \leftarrow \text{Update}(\text{aslist}, PS)\)
20. **if** \(\text{aslist} = \emptyset\)
21. \(\text{BeamComp} = PS\)
22. \(ws = \min(FV_i)\)
23. **break**
24. **else**
25. \(l = l + 1\)
26. **end if**
27. **else**
28. \(l = l + 1\)
First of all, the root nodes are produced for level, $l = 0$, according to the precedence relations in Algorithm 2. Then, descendant nodes, $PS$, are produced for each beam in $l = l + 1$. An example of the procedure of generating descendant nodes, based on the structure in the study by Sabuncuoglu, Gocgun, and Erel (2008), is given in Figure 3.

The beam tree is generated in Figure 3(b) according to the precedence diagram in Figure 3(a). The grey nodes show selected nodes in each level. The branch 1-3-2-0-4-5 denotes that tasks 1, 2 and 3 are assigned to the first station and tasks 4 and 5 are assigned to the second station in the order shown. The task 0 nodes illustrate that the current station is closed. If the total number of nodes, $m$, is greater than the beam width, $\beta$, at any level, each partial solution is extended in function $\text{ExtendPartialSolution}(PS, T, l, P)$. If the number of stations of extended solutions, $FVi$, is equal to the predetermined number of stations, $W$, the function is finished for the current cycle time. The extension and solution selection procedures are given in the next sections. At the end of each stage, the beams are selected up to $\beta_{ext}$ in function $\text{ChoosingBestSolutions}(FVi, BExtendingi, \beta_{ext})$ to transfer to the next stage. The algorithm can work for up to 300 s for each model but if the algorithm cannot find any feasible solution for any cycle time, the algorithm is stopped.

### 3.1. Extending partial solution

Several priority-based heuristic algorithms are available in the literature (Talbot, Patterson, and Gehrlein 1986; Scholl and Voß 1996; Borba and Ritt 2014; Otto and Otto 2014). Six priority rules (Table 3) in the local search stage, namely, maximum follower (MF), shortest task time (STT), descending minimum positional weights (Max$PW^-$), combination of maximum follower and longest task times (MF&L), equal priority or random search (RS) and mixing of the best three methods with equal probabilities [mixing priority rules (MPR)], are proposed for the complete solution.
Table 3. Priority rules for the robotic mixed-model assembly line balancing problem.

| Task priority rule | References |
|--------------------|------------|
| Maximum follower method (MF) | Talbot, Patterson, and Gehlein (1986) |
| Shortest task time (STT) | – |
| Equal priority (random search) (RS) | – |
| Descending minimum positional weights (MaxPW−) | Borba and Ritt (2014) |
| Combination of maximum follower and longest task time (MF&L) | Otto and Otto (2014) |
| Mixing priority rules (MPR) | – |

The aim is to evaluate the effects of the local search strategy on the performance of the BS algorithm.

MF: The MF method provides priority according to the total number of followers. If the task has a higher total number of followers, it takes higher priority. In other words, if the duration of tasks is less than the remaining time of any robot for each model, it is taken from the assignable list under precedence relations, and then the task is selected from the assignable list by considering the total number of maximum followers. If certain tasks have equal total numbers of followers, a task is selected randomly from these tasks.

STT: In this method, the task is selected according to Equation (20). The aim of using Equation (20) is to find a task with the minimum possible time loading at the station. ST is calculated for each task in the assignable list and the task with the minimum ST is assigned to the station. If certain tasks have the same ST value, a task is selected randomly from among these tasks.

\[ ST_i = \min_r \left( \max_m \left( \sum_{k \in AL} t_{kmr} + t_{imr} \right) \right) \]  

RS: In this method, it is assumed that each task has equal priority to be assigned to a station. The task is chosen randomly from the assignable list. The method can enable searching in different areas in the solution space owing to randomization.

MaxPW−: This method is adapted as \( pw_i^- = t_i^- + \sum_{h \in Pa} t_h^- \), where \( t_i^- = \min_r(\min_m t_{imr}) \) for the RAMALB problem.

MF&L: The maximum combination of MF and the longest task time method with equal weight is selected as a priority rule where \( MF&L_i = MF_i + L_i \).

MPR: The best three priority rules are used with equal probability to use the advantages of the methods and differentiate between search areas in the solution space. These priority rules will be determined after the performance of other rules has been tested on the problem. A random integer between 1 and 3 is generated. According to the random number, one of the priority rules is applied to assign the task to the station.

3.2. Robot assignment procedure

The robot assignment procedure, AsRw, is employed simultaneously during extension. First, unassigned tasks are assigned to the current station until there is no assignable task for any robots in the current cycle times, \( C_m \). In other words, because there is no limit to assigning the same robot to different stations, the processing time of the station is considered for all robots and then the most suitable robot is selected to increase line efficiency. For example, in Figure 4, a feasible solution is illustrated to explain the robot assignment procedure in detail. The solution is given for cycle times of 30 s and 25 s for Models 1 and 2, respectively, and it includes two stations and two robots. With regard to station 1,
R1 dominates R2 in terms of line efficiency. Even though it is feasible for both robots to be assigned to these task groups in station 1, R1 can perform tasks in a smaller processing time for both models. For the second station, assigning R2 leads to infeasible solution because the total processing time of Model 1 is greater than the current cycle time. Therefore, R1 must also be assigned to the second station.

3.3. Solution selection and fitness function

The solutions are ordered according to the fitness function and the best solutions are chosen to transfer to the next step by $\beta_{ext}$ solutions. The number of stations in a given cycle time is selected as a fitness function value. If different solutions are found with the same result, the solution is selected randomly.

4. Computational experiments

The presentation of the computational experiments is divided into three subsections. First, the procedure of test problem generation is explained in detail. Then, the proposed mathematical models are compared in terms of solution efficiency. This section closes with an analysis of the performance of different versions of the BS algorithm on the generated test problems.

4.1. Test problem generation

For the combined precedence diagram, eight test problems with from 25 to 297 tasks are selected from the problem set, which is available at http://www.assembly-line-balancing.de/. The selected test bed is generally used in the RALB problem literature, whereas the times based on the number of models and robots are generated for this study.

The generation of the test bed is considered in terms of several parameters (Table 4). The mixed model lines allow more than one model to be produced on the same line; therefore, in data generation, two different models are considered. The allocation of robots to stations is another important issue.

| 1 | 3 | 2 | 4 | 5 |
|---|---|---|---|---|
| Ws1 | Ws2 |

Figure 4. Robot assignment procedure.
in RALB problems. The increase in the number of robot types may make it difficult to discover better solutions. In this case, five different types of robot are considered.

Different robots have different features, so they can perform a task with different operation times. First of all, random task times are produced for the first robot independently, which are determined as base times. Then, task times of the other robots are produced randomly and compared to the first robot’s times according to variability levels. Three variability levels, i.e. low (±10%), medium (±20%) and high (±30%), are defined. Hence, the maximum performance difference between robots is assumed to be 60%. For example, if a task time of 50 s is produced randomly for the first robot, the task time for the other robots can be between 45 and 55 s, 40 and 60 s and 35 and 65 s for low, medium and high variability levels, respectively.

Also, the ratio between minimum and maximum task time is used to measure the homogeneity and heterogeneity of the tasks in the problem. The selection of the small ratio value shows the homogeneity of the tasks in the classical assembly line because labour-intensive tasks have no extreme difference between them (Amen 2001). Nevertheless, an RAL can contain different type of tasks; for example, while some tasks are simply pick and place, other tasks need to be done precisely and slowly. Therefore, a high ratio, which is 20, is selected in this study while generating task times.

In this computational study, four different predetermined numbers of stations and three different time variability levels (low, medium and high) are considered for eight problem sets. These parameters result in $4 \times 3^8 = 96$ test problems. However, the results of the mathematical models are compared only for the small and medium-sized problems, from 25 to 53 tasks (three different problem sets), because the models do not ensure optimal solutions for the larger sized problems. Therefore, 36 of out 96 test problems are used to compare the models. All computational experiments are conducted on a personal computer with a two Intel® Xeon® 4 Core 2.40 GHz processor with 8 GB of memory.

### 4.2. Comparison of mathematical models

In this section, the performance of the proposed models is evaluated using newly generated benchmark problems. The test-bed generation method was mentioned in the previous section. A GAMS/CPLEX-12.3 solver is used for solving the mathematical models.

Table 5 demonstrates the total numbers of constraints and variables in the models. It is clearly seen that Model 2 includes many more constraints than Model 1. However, the number of variables is not significantly different.

Figures 5 and 6 illustrate the numbers of constraints and variables in the solver from 25 tasks to 53 tasks in the test problems for the models. Model 2 contains more constraints for all problems,

| Model  | No. of constraints | No. of total variables | No. of binary variables |
|--------|--------------------|------------------------|-------------------------|
| Model 1 | $l + |P| + J(M + R + 2)$ | $J.R.(l + 1) + M$ | $J.R.(l + 1)$ |
| Model 2 | $l.(M + 1) + J.M.(|P| + J.R)$ | $J.(l + R) + (l - 1).(l - 2)/2 + |P| + J$ | $- |P_a| + M.(l + 1)$ |

Table 4. Range of parameters.

| Parameter                        | Range           |
|----------------------------------|-----------------|
| Number of models                 | 2               |
| Number of robots                 | 5               |
| Minimum and maximum task time    | 5–100           |
| Time variation between robots    | 20%, 40%, 60%   |
although Model 2 usually has fewer variables than Model 1 for all problems. Model 1 is much more sensitive to the predetermined number of stations, a trend that can be seen in Figure 6.

Results of the models are presented in Table 6. According to Table 6, Model 1 outperforms Model 2 for all test problems in terms of results and CPU times. While Model 1 provides the optimal solution to more than half of the problems within 3600 s, Model 2 could not find any proven optimal solutions. Model 1 gives the optimal solution for seven, eight and six out of 12 problems for low, medium and high variability levels on time, respectively, whereas Model 2 only finds three, five and three integer solutions that are known to be optimal from Model 1 for low, medium and high variability levels. With regard to gaps, Model 1 has a lower value of gaps than Model 2. Proven optimal solutions of Model 1 are considered for evaluating the performance of BS.

4.3. Comparison of versions of beam search

In this section, the performance of the proposed algorithm is evaluated using newly generated benchmark problems for low, medium and high variability levels.

Preliminary experiments are needed to determine proper parameter values. Five pairs of $\beta$ and $\beta_{ext}$ parameters are tested on the selected problems from small size to large size (25 tasks, 53 tasks and 111 tasks). These are tested on all the versions of the algorithm. Figures 7–9 illustrate the test results. According to these figures, the proposed algorithm ensures the best solution with $\beta$ and $\beta_{ext}$ values of 20 and 20, respectively. The algorithms are coded in MATLAB® and 10 instances are run for 96 problems and each version. The time limit for each run is determined as 300 s for each model (Table 7).
Table 6. Comparison of the models.

| Problem          | No. of tasks | Station | Total C | C1    | C2    | CPU (s) | Gap (%) | Total C | C1    | C2    | CPU (s) | Gap (%) |
|------------------|--------------|---------|---------|-------|-------|---------|---------|---------|-------|-------|---------|---------|
|                  |              |         |         |       |       |         |         |         |       |       |         |         |
| **Low time variability** |              |         |         |       |       |         |         |         |       |       |         |         |
| 25               | 3            |         | 760     | 389   | 371   | 5       | –       | 760     | 389   | 371   | +3600   | 17.6    |
|                  | 4            |         | 578     | 294   | 284   | 11      | –       | 578     | 294   | 284   | +3600   | 20.9    |
|                  | 6            |         | 403     | 203   | 200   | 880     | –       | 403     | 203   | 200   | +3600   | 27.5    |
|                  | 9            |         | 278     | 147   | 131   | 3091    | –       | 282     | 145   | 137   | +3600   | 37.2    |
| 35               | 8            |         | 891     | 438   | 453   | 96      | –       | 896     | 441   | 455   | +3600   | 44.1    |
|                  | 5            |         | 712     | 354   | 358   | 477     | –       | 715     | 354   | 361   | +3600   | 46.9    |
|                  | 7            |         | 523     | 253   | 270   | +3600   | 5.2     | 525     | 260   | 265   | +3600   | 48.4    |
|                  | 9            |         | 396     | 150   | 156   | +3600   | 7.3     | 327     | 160   | 167   | +3600   | 88.8    |
| 53               | 5            |         | 972     | 495   | 477   | 270     | –       | 977     | 499   | 478   | +3600   | 37.2    |
|                  | 7            |         | 698     | 342   | 356   | +3600   | 2.5     | 713     | 339   | 374   | +3600   | 47.2    |
|                  | 10           |         | 496     | 245   | 251   | +3600   | 7.8     | 503     | 242   | 261   | +3600   | 61.5    |
|                  | 14           |         | 370     | 184   | 164   | +3600   | 13.2    | 399     | 215   | 184   | +3600   | 90.1    |
| **Medium time variability** |              |         |         |       |       |         |         |         |       |       |         |         |
| 25               | 3            |         | 792     | 403   | 389   | 5       | –       | 792     | 403   | 389   | +3600   | 9.5     |
|                  | 4            |         | 589     | 305   | 284   | 9       | –       | 589     | 305   | 284   | +3600   | 10.7    |
|                  | 6            |         | 424     | 217   | 207   | 147     | –       | 424     | 217   | 207   | +3600   | 16.2    |
| 35               | 4            |         | 747     | 374   | 373   | 28      | –       | 747     | 374   | 373   | +3600   | 30.7    |
|                  | 5            |         | 604     | 304   | 300   | 163     | –       | 613     | 318   | 295   | +3600   | 33.7    |
|                  | 7            |         | 429     | 220   | 209   | 3350    | –       | 435     | 226   | 209   | +3600   | 42.5    |
| 53               | 5            |         | 901     | 465   | 436   | 28      | –       | 906     | 452   | 454   | +3600   | 35.1    |
|                  | 7            |         | 647     | 328   | 319   | 2991    | –       | 666     | 333   | 333   | +3600   | 39.0    |
|                  | 10           |         | 489     | 248   | 241   | 3140    | 15      | 386     | 248   | 238   | +3600   | 46.9    |
|                  | 14           |         | 342     | 178   | 164   | +3600   | 14.6    | 363     | 188   | 175   | +3600   | 71.3    |
| **High time variability** |              |         |         |       |       |         |         |         |       |       |         |         |
| 25               | 3            |         | 724     | 386   | 338   | 5       | –       | 724     | 383   | 341   | +3600   | 42.5    |
|                  | 4            |         | 537     | 282   | 255   | 12      | –       | 540     | 282   | 258   | +3600   | 17.2    |
|                  | 6            |         | 349     | 186   | 163   | 59      | –       | 358     | 189   | 169   | +3600   | 11.8    |
|                  | 9            |         | 244     | 129   | 115   | +3600   | 3.6     | 244     | 129   | 115   | +3600   | 34.1    |
| 35               | 4            |         | 764     | 369   | 395   | 20      | –       | 785     | 380   | 405   | +3600   | 30.3    |
|                  | 5            |         | 611     | 296   | 315   | 208     | –       | 611     | 296   | 315   | +3600   | 30.1    |
|                  | 7            |         | 443     | 211   | 232   | +3600   | 2.3     | 458     | 229   | 229   | +3600   | 36.4    |
|                  | 12           |         | 267     | 147   | 120   | +3600   | 17.2    | 276     | 147   | 129   | +3600   | 52.5    |
| 53               | 5            |         | 967     | 484   | 483   | 205     | –       | 972     | 500   | 473   | +3600   | 34.4    |
|                  | 7            |         | 705     | 370   | 335   | +3600   | 3       | 705     | 370   | 335   | +3600   | 38.0    |
|                  | 10           |         | 489     | 256   | 233   | +3600   | 12.3    | 505     | 265   | 240   | +3600   | 47.7    |
|                  | 14           |         | 365     | 210   | 155   | +3600   | 19.7    | 383     | 190   | 193   | +3600   | 90.7    |

*a* Integer solution (optimality not proven).

Figure 7. Parameter optimization for small size. BW-Bext = beam width-max. number of feasible solution; CPU = central processing unit time; RS = random search; MF = maximum follower; STT = shortest task time; Mixed = mixing priority rules; MF&L = combination of maximum follower and longest task times.
Figure 8. Parameter optimization for medium size. BW-Bext = beam width-max. number of feasible solution; CPU = central processing unit time; RS = random search; MF = maximum follower; STT = shortest task time; Mixed = mixing priority rules; MF&L = combination of maximum follower and longest task times.

Figure 9. Parameter optimization for large size. BW-Bext = beam width-max. number of feasible solution; CPU = central processing unit time; RS = random search; MF = maximum follower; STT = shortest task time; Mixed = mixing priority rules; MF&L = combination of maximum follower and longest task times.

Table 7. Parameters of beam search.

| Parameter                  | Value       |
|----------------------------|-------------|
| Beam width ($\beta$)      | 20          |
| Beam extension ($\beta_{ext}$) | 20          |
| Max. computational time    | 300 s for each model |

All of the obtained solutions are given in Tables A1–A3 in the appendices (see supplementary data). Only the minimum total cycle times, each model cycle time and average CPU times are reported for each test problem. The first column of the tables shows the number of the tasks in the problem. Each test problem is analysed for four different predetermined numbers of stations shown in the second column. Optimal solutions in the third column are obtained from the mathematical model for certain test problems. Total and CPU (s) represent total cycle time and computational times in seconds. C1 and C2 symbolize each product’s model cycle time values. The tables clearly illustrate that for the time variability and size of the problem, BS with MPR usually dominates the other versions. In terms of computational times, there is no significant difference between versions.

The results obtained for each method are summarized in Tables 8–10. The following parameters are utilized to evaluate the proposed algorithms:
Av. CPU (s): average computation time in seconds  
#opt: number of found (proven) optimal solutions  
#best: number of best results found among the versions  
Av. diff. #best (%): average percentage differences between obtained solutions and best results.

Table 8 illustrates the results obtained for the low time variability set. The proposed versions are compared with the optimal solution and each other in order to compare their effectiveness. According to Table 8, for low variability optimal results are known for only seven problems. BS with MPR dominates the other versions in terms of proven optimal solution and finds the best results. This MPR version is obtained by mixing MF, RS and MF&L priority rules. This version offers four out of seven proven optimal solutions and the best results for 26 out of 32 problems. On the other hand, BS with STT, MaxPW−, RS, MF and MF&L provide the best results for two, three, seven, seven and eight, respectively. In terms of average CPU time, there is no substantial difference between the versions. Moreover, average differences from the best solution are about 1% for all versions except for BS with STT and BS with MaxPW−.

Tables 9 and 10 show the results obtained for medium and high time variability levels. According to Table 9 and 10, BS with MPR outperforms other versions for medium variability and BS with RS dominates for high time variability in terms of finding optimal solutions. While the MPR version

**Table 8. Summary of results of all versions for low time variability.**

| Low time variability | BS with MF | BS with STT | BS with RS | BS with MaxPW− | BS with MF&L | BS with MPR |
|----------------------|-----------|------------|-----------|----------------|--------------|------------|
| Av. CPU (s)          | 277       | 304        | 282       | 248            | 274          | 265        |
| #opt                 | 3(7)      | 2(7)       | 3(7)      | 3(7)           | 3(7)         | 4(7)       |
| #best                | 7         | 2          | 7         | 3              | 8            | 26         |
| Av. difference from #best | < 1      | 2.6        | 1.1       | 2.6            | 1.1          | < 1        |
| Total problem        | 32        | 32         | 32        | 32             | 32           | 32         |

**Table 9. Summary of results of all versions for medium time variability.**

| Medium time variability | BS with MF | BS with STT | BS with RS | BS with MaxPW− | BS with MF&L | BS with MPR |
|-------------------------|-----------|------------|-----------|----------------|--------------|------------|
| Av. CPU (s)             | 285       | 290        | 259       | 266            | 274          | 299        |
| #opt                    | 2(8)      | 1(8)       | 3(8)      | 1(8)           | 1(8)         | 5(8)       |
| #best                   | 9         | 1          | 9         | 1              | 5            | 23         |
| Av. diff. #best (%)     | 1.4       | 2.8        | < 1       | 3.03           | 1.2          | < 1        |
| Total problem           | 32        | 32         | 32        | 32             | 32           | 32         |

**Table 10. Summary of results of all versions for high time variability.**

| High time variability   | BS with MF | BS with STT | BS with RS | BS with MaxPW− | BS with MF&L | BS with MPR |
|-------------------------|-----------|------------|-----------|----------------|--------------|------------|
| Av. CPU (s)             | 303       | 289        | 258       | 241            | 305          | 300        |
| #opt                    | 1(6)      | –          | 3(6)      | –              | 1(6)         | 2(6)       |
| #best                   | 9         | –          | 6         | –              | 7            | 20         |
| Av. diff. #best (%)     | < 1       | 2.7        | 1.3       | 3.1            | 1.02         | < 1        |
| Total problem           | 32        | 32         | 32        | 32             | 32           | 32         |

BS = beam search; MF = maximum follower; STT = shortest task time; RS = random search; MaxPW− = descending minimum positional weights; MF&L = combination of maximum follower and longest task time; MPR = mixing priority rules; CPU = central processing unit time.
yields optimal solution for five out of eight and two out of six problems for medium and high variability levels, the RS version provides optimal solutions for three out of eight and three out of eight problems for medium and high variability levels, respectively. On the other hand, BS with MPR version outperforms the other versions in terms of the number of best results found among the versions. It provides the best results almost twice as often as the other versions.

The performance of BS with STT and MaxPW is very unsatisfactory in terms of solution quality because it is observed that there is no significant effect on the computed initial solution quality. Also, average differences from the best result are higher than the other versions. In addition, solution quality is not considerably affected, although CPU time has increased. This demonstrates the complexity of the problem and highlights the respectable performance of the other versions.

Considering the performance of BS with RS, randomization provides better solutions, especially in small-sized problems, owing to the solution space. The chance of finding the best solution is high through randomization when the problem size is small. Furthermore, the average computational time is smaller compared with other versions owing to the assignment rule. Therefore, it presents acceptable solution quality in reasonable computational time.

The performance of BS with MF competes with MF&L for proven optimal solutions and finding the best solutions. While the MF version provides better performance in small and medium-sized test problems, the MF&L yields better results for larger test problems. Moreover, the average difference from the best result is very small, showing that when it could not find the best solution, it could find solutions very close to the best one.

With respect to BS with MPR, it performs very well in terms of finding the best results and proven optimal solution. The purpose of MPR is to use the advantages of each method. It outperforms other algorithms for all time variabilities. With the increase in the $\beta$ and $\beta_{\text{ext}}$ parameters, the performance of the method is better but the computation takes a long time. In addition, the average difference from the best result is a maximum of 1% when it does not find the best solution. Overall, the average difference is less than 1% and it is smaller than the other methods for all time variability levels.

With regard to managerial implications of the proposed approach, and computational advantages, the optimal and feasible solutions obtained by BS could have several consequences for managers. First, it provides feedback to engineers, so that the line design could lead to higher efficiencies in the assembly process. Secondly, it helps decision makers to reduce cycle time and increase line efficiency. Finally, it could achieve economies of scale, division of robots and better balance.

5. Conclusions and future research directions

According to the International Federation of Robotics (2015), approximately 1.3 million industrial robots will enter operation in the next 2 years. They may be used instead of human labour, especially in assembly systems. Therefore, in this article, the RMALB problem is considered. Mixed-integer linear programming models proposed by Yoosefelahi et al. (2012) and Aghajani, Ghodsi, and Javadi (2014) are modified to define the RMALB-II problem mathematically. In addition, the modified mathematical models are compared with each other to demonstrate the performance of the mathematical models. The modified model of Yoosefelahi et al. (2012) reaches an optimal solution in less computational time for the small and medium-sized test problem. A BS-based approach is developed to solve the problem for minimizing the sum of cycle times over all models. Several versions of the BS algorithm are developed, using different priority rules at the local search stage. The aim of using the proposed different versions is to evaluate the effect of the local search strategy on the performance of the BS algorithm. The performance of the algorithm and models is compared using newly generated test problems. The results of the proposed algorithm show that it is very competitive and is a promising tool for further research on the RMALB-II problem.

The limitations of this study are as follows: (1) the mathematical models provide proven optimal solutions up to the medium-sized (53 tasks) problem; and (2) there is no effective lower bound to compare the performance of the algorithms. In future, sequence-dependent set-up and tool-changing
times should be taken into account in the problem. The proposed approach can be applied to other types of RMALB problem, such as parallel, two-sided and U-line set-ups. Finally, balancing and model sequencing problems could be considered simultaneously in future research.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**ORCiD**

Zeynel Abidin Çil [http://orcid.org/0000-0002-7270-9321](http://orcid.org/0000-0002-7270-9321)

Süleyman Mete [http://orcid.org/0000-0001-7631-5584](http://orcid.org/0000-0001-7631-5584)

**References**

Aghajani, M., R. Ghodsi, and B. Javadi. 2014. “Balancing of Robotic Mixed-Model Two-Sided Assembly Line with Robot Setup Times.” *The International Journal of Advanced Manufacturing Technology* 74: 1005–1016.

Akpınar, S., and G. M. Bayhan. 2011. “A Hybrid Genetic Algorithm for Mixed Model Assembly Line Balancing Problem with Parallel Workstations and Zoning Constraints.” *Engineering Applications of Artificial Intelligence* 24: 449–457.

Amen, M. 2001. “Heuristic Methods for Cost-Oriented Assembly Line Balancing: A Comparison on Solution Quality and Computing Time.” *International Journal of Production Economics* 69: 255–264.

Baldi, M., T. Crainic, G. Perboli, and R. Tadei. 2014. “Branch-and-Price and Beam Search Algorithms for the Variable Cost and Size Bin Packing Problem with Optional Items.” *Annals of Operations Research* 222 (1): 125–141.

Battaia, O., X. Delorme, A. Dolgui, J. Hagemann, A. Horlemann, S. Kovalev, and S. Malyutin. 2015. “Workforce Minimization for a Mixed-Model Assembly Line in the Automotive Industry.” *International Journal of Production Economics* 170: 489–500.

Battaia, O., and A. Dolgui. 2013. “A Taxonomy of Line Balancing Problems and their Solution Approaches.” *International Journal of Production Economics* 142: 259–277.

Baybars, I. 1986. “A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem.” *Management Science* 32: 909–932.

Blum, C. 2008. “Beam-ACO for Simple Assembly Line Balancing.” *INFORMS Journal on Computing* 20 (4): 618–627.

Blum, C., J. Bautista, and J. Pereira. 2006. “Beam-ACO Applied to Assembly Line Balancing.” *Lecture Notes in Computer Science* 4150: 96–107.

Blum, C., and C. Miralles. 2011. “On Solving the Assembly Line Worker Assignment and Balancing Problem via Beam Search.” *Computers and Operations Research* 43: 1411–1426.

Borba, L., and M. Ritt. 2014. “A Heuristic and a Branch-and-Bound Algorithm for the Assembly Line Worker Assignment and Balancing Problem.” *Computers & Operations Research* 45: 87–96.

Buyukozkan, K., I. Kucukkoc, S. I. Satoglu, and D. Z. Zhang. 2016. “Lexicographic Bottleneck Mixed-Model Assembly Line Balancing Problem: Artificial Bee Colony and Tabu Search Approaches with Optimised Parameters.” *Expert Systems with Applications* 50: 151–166.

Daoud, S., H. Chehade, F. Yalaoui, and L. Amodeo. 2014. “Solving a Robotic Assembly Line Balancing Problem using Efficient Hybrid Methods.” *Journal of Heuristics* 20: 235–259.

Erel, E., Y. Göçgun, and I. Sabuncuoglu. 2007. “Mixed-Model Assembly Line Sequencing using Beam Search.” *International Journal of Production Research* 45 (22): 5265–5284.

Erel, E., I. Sabuncuoglu, and H. Sekerci. 2005. “Stochastic Assembly Line Balancing using Beam Search.” *International Journal of Production Research* 43: 1411–1426.

Gao, J., L. Sun, L. Wang, and M. Gen. 2009. “An Efficient Approach for Type II Robotic Assembly Line Balancing Problems.” *Computers & Industrial Engineering* 56: 1065–1080.

Gökçen, H., and E. Erel. 1998. “Binary Integer Formulation for Mixed-Model Assembly Line Balancing Problem.” *Computers & Industrial Engineering* 34 (2): 451–461.

Golle, U., F. Rothlauf, and N. Boysen. 2015. “Iterative Beam Search for Car Sequencing.” *Annals of Operations Research* 226: 239–254.

Honda, N. 2003. “Backtrack Beam Search for Multi objective Scheduling Problem.” *Advances in Soft Computing* 21: 147–152.

International Federation of Robotics. 2015. “World Robotic 2015 Industrial Robots.” [http://www.ifr.org/industrial-robots/statistics/](http://www.ifr.org/industrial-robots/statistics/).

Kara, Y., U. Ozcan, and A. Peker. 2007. “An Approach for Balancing and Sequencing Mixed-Model JIT U-Lines.” *The International Journal of Advanced Manufacturing Technology* 32: 1218–1231.
Kim, H., and S. Park. 1995. “A Strong Cutting Plane Algorithm for the Robotic Assembly Line Balancing Problem.” *International Journal of Production Research* 33 (8): 2311–2323.

Leu, Y. 1997. “Using Beam Search Techniques for Sequencing Mixed-Model Assembly Lines.” *Annals of Operations Research* 70: 379–397.

Levitin, G., J. Rubinovitz, and B. Shnits. 2006. “A Genetic Algorithm for Robotic Assembly Line Balancing.” *European Journal of Operational Research* 168: 811–825.

Lowerre, B. T. 1976. “The HARPY Speech Recognition System.” Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA.

McMullen, P. R., and P. Tarasewich. 2005. “A Beam Search Heuristic Method for Mixed-Model Scheduling with Setups.” *International Journal of Production Economics* 96: 273–283.

Nilakantan, J. M., G. Huang, and S. G. Ponnambalam. 2015. “An Investigation on Minimizing Cycle Time and Total Energy Consumption in Robotic Assembly Line Systems.” *Journal of Cleaner Production* 90: 311–325.

Nilakantan, J. M., I. Nielsen, S. G. Ponnambalam, and S. Venkataramanaiah. 2016. “Differential Evolution Algorithm for Solving RALB Problem using Cost- and Time-Based Models.” *The International Journal of Advanced Manufacturing Technology*. DOI 10.1007/s00170-016-9086-2.

Nilakantan, M., and S. G. Ponnambalam. 2012. “An Efficient PSO for Type II Robotic Assembly Line Balancing Problem.” *IEEE International Conference on Automation Science and Engineering (CASE).* doi:10.1109/CoASE.2012.6386398.

Nilakantan, M., and S. G. Ponnambalam. 2016. “Robotic U-Shaped Assembly Line Balancing using Particle Swarm Optimization.” *Engineering Optimization* 48 (2): 231–252.

Rubban, M., Z. Mousavi, and H. Farrokhi-Asl. 2016. “Multi-objective Metaheuristic for Solving a Type II Robotic Mixed-Model Assembly Line Balancing Problem.” *Journal of Industrial and Production Engineering*. http://dx.doi.org/10.1080/21681015.2015.1126656.

Rubinovitz, J., J. Bukchin, and E. Lenz. 1993. “RALB—A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines.” *CIRP Annals—Manufacturing Technology* 42: 497–500.

Sabuncuoglu, I., Y. Gocgun, and E. Erel. 2008. “Backtracking and Exchange of Information: Methods to Enhance a Beam Search Algorithm for Assembly Line Scheduling.” *European Journal of Operational Research* 186: 915–930.

Salveson, M. E. 1955. “The Assembly Line Balancing Problem.” *Journal of Industrial Engineering* 6 (3): 18–25.

Scholl, A., and C. Becker. 2006. “State-of-the-Art Exact and Heuristic Solution Procedures for Simple Assembly Line Balancing.” *European Journal of Operational Research* 168: 666–693.

Scholl, A., N. Boysen, and M. Fliedner. 2013. “The Assembly Line Balancing and Scheduling Problem with Sequence-Dependent Setup Times: Problem Extension, Model Formulation and Efficient Heuristics.” *OR Spectrum* 35: 291–320.

Scholl, A., and S. Voß. 1996. “Simple Assembly Line Balancing-Heuristic Approaches.” *Journal of Heuristics* 2: 17–244.

Simaria, A. S., and P. M. Vilarinho. 2004. “A Genetic Algorithm Based Approach to the Mixed-Model Assembly Line Balancing Problem of Type II.” *Computers & Industrial Engineering* 47: 391–407.

Simaria, A. S., and P. M. Vilarinho. 2009. “2-ANTBAL: An Ant Colony Optimisation Algorithm for Balancing Two-Sided Assembly Lines.” *Computers & Industrial Engineering* 56: 489–506.

Talbot, F. B., J. H. Patterson, and W. V. Gehrlein. 1986. “A Comparative Evaluation of Heuristic Line Balancing Techniques.” *Management Science* 32: 430–454.

Yavuz, M. 2010. “An Iterated Beam Search Algorithm for the Multi-level Production Smoothing Problem with Workload Smoothing Goal.” *International Journal of Production Research* 48 (20): 6189–6202.

Yoosufalehi, A., M. Aminnayeri, H. Mosadegh, and H. D. Ardakani. 2012. “Type II Robotic Assembly Line Balancing Problem: An Evolution Strategies Algorithm for a Multi-objective Model.” *Journal of Manufacturing Systems* 31: 139–151.