Branch, bound and remember algorithm for two-sided assembly line balancing problem

Zixiang Li , Ibrahim Kucukkoc , Zikai Zhang

PII:	S0377-2217(20)30053-9
DOI:	https://doi.org/10.1016/j.ejor.2020.01.032
Reference:	EOR 16286

To appear in: *European Journal of Operational Research*

Received date:	22 July 2019
Accepted date:	15 January 2020

Please cite this article as: Zixiang Li , Ibrahim Kucukkoc , Zikai Zhang , Branch, bound and remember algorithm for two-sided assembly line balancing problem, *European Journal of Operational Research* (2020), doi: https://doi.org/10.1016/j.ejor.2020.01.032

Highlights

- Branch, bound and remember algorithm is improved for two-sided assembly lines.
- Modified Hoffman heuristic is first applied to achieve a high-quality upper bound.
- New dominance rules and lower bounds are developed for state-of-the-art results.
- Proposed methods outperform the current best exact method and metaheuristic.
- Optimal solutions are achieved for all the benchmarks tested.

Branch, bound and remember algorithm for two-sided assembly line balancing problem

Zixiang Li[1,2], Ibrahim Kucukkoc [3*], Zikai Zhang [2]

[1] Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, China

[2] Engineering Research Center for Metallurgical Automation and Measurement Technology of Ministry of Education, Wuhan University of Science and Technology, China

[3] Industrial Engineering Department, Balikesir University, Cagis Campus, Balikesir 10145, Turkey

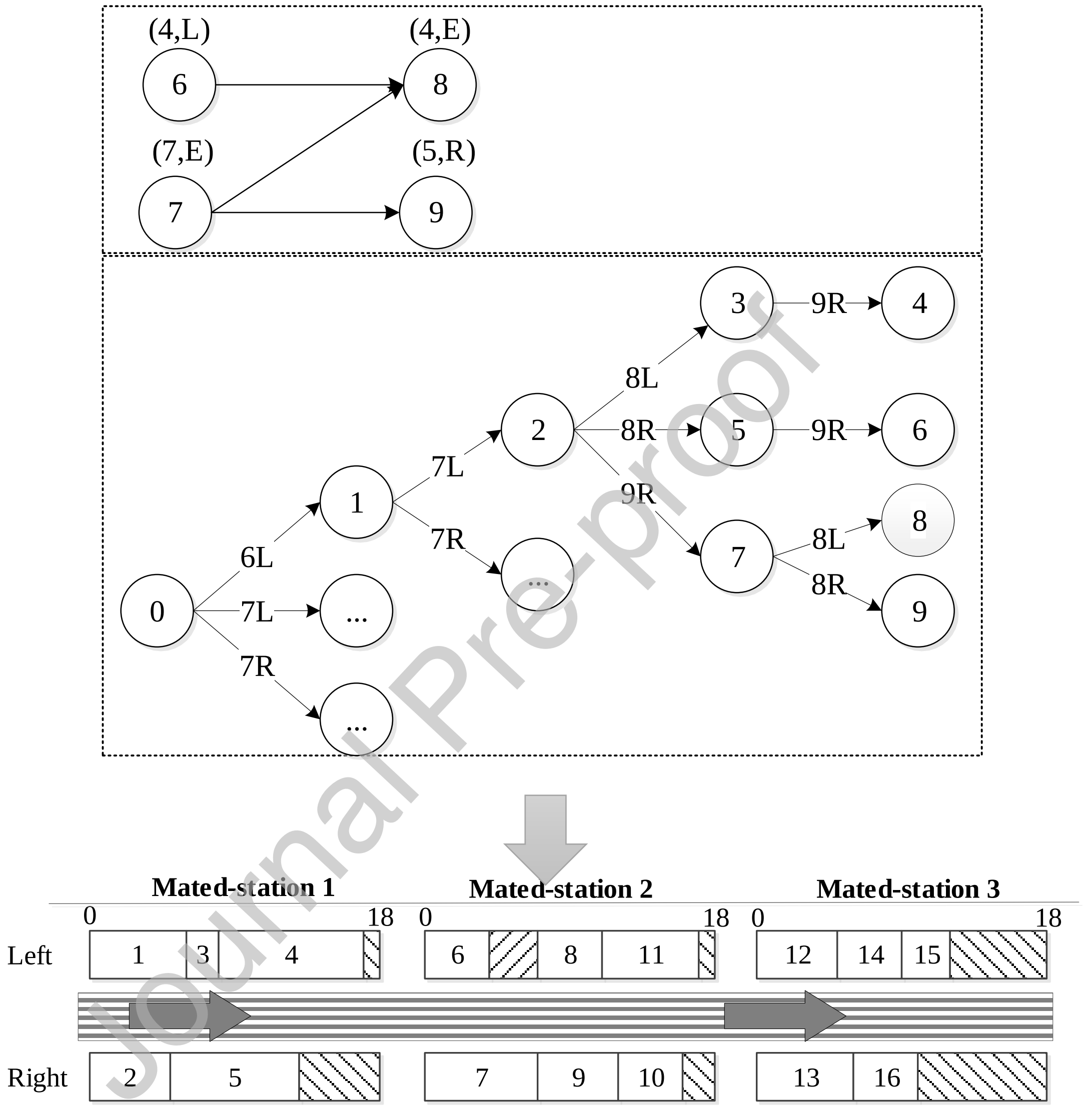
Emails: zixiangliwust@gmail.com (Z-X Li); ikucukkoc@balikesir.edu.tr (I. Kucukkoc); zhangzikai0703@gmail.com (Z-K Zhang);

*Corresponding author. Tel: +902666121194 (Ext. 126407)

Abstract: This research presents a new branch, bound and remember (BBR) algorithm to minimize the number of mated-stations in two-sided assembly lines. The proposed methodology modifies the Hoffman heuristic to achieve high-quality upper bounds, and employs two new dominance rules, referred to as memory-based maximal load rule and memory-based extended Jackson rule, to prune the sub-problems. The BBR algorithm also employs several other improvements to enhance the performance, including renumbering the tasks and new lower bounds. Computational results demonstrate that BBR achieves the optimal solutions for all the tested instances within 1.0 second on average, including two optimal solutions for the first time. Comparative study shows that BBR outperforms the current best exact method (branch and bound algorithm) and the current best heuristic algorithm (iterated greedy search algorithm). As a consequence, the proposed BBR can be regarded as the state-of-the-art method for TALBP.

Keywords: Combinatorial optimization; Assembly line balancing; Two-sided assembly line; Branch and bound; Branch, bound and remember

Graphical Abstract



1. Introduction

Assembly lines are intensively utilized in mass production due to higher production efficiency. An assembly line constitutes a set of connected workstations, and the assembly tasks are divided and operated by the workers on workstations (Michels, Lopes, Sikora, & Magatão, 2019; Mosadegh, Fatemi Ghomi, & Süer, 2020). As a variant of the simple assembly line, two-sided assembly lines are built to assemble large-size products, e.g. cars, trucks and motorcycles (Bartholdi, 1993). A two-sided assembly line comprises a set of mated-stations, and there are two facing workstations or two sides inside one mated-station. Two workers allocated to the two facing workstations operate the tasks in parallel. To improve the assembly efficiency, two-sided assembly line balancing problem (TALBP) attracts increasing attention from both academics and practitioners (Abdullah Make, Ab. Rashid, & Razali, 2017).

1.1 Related work

The applied methodologies for TALBP can be categorized into three types: exact methods, heuristic methods and metaheuristic methods (Li, Kucukkoc, & Nilakantan, 2017). The researches on metaheuristic methods comprise of the majority of the published researches. These metaheuristics include genetic algorithms (Delice, Kızılkaya Aydoğan, & Özcan, 2016; Kim, Kim, & Kim, 2000; Kim, Song, & Kim, 2009; Kucukkoc & Zhang, 2015a), ant colony optimization algorithms (Baykasoglu & Dereli, 2008; Kucukkoc & Zhang, 2015b; Simaria & Vilarinho, 2009), tabu search algorithms (Özcan, Gökçen, & Toklu, 2010; Özcan & Toklu, 2008), simulated annealing algorithms (Khorasanian, Hejazi, & Moslehi, 2013; Özcan &

Toklu, 2009), particle swarm optimization algorithms (Chutima & Chimklai, 2012; Delice, Kızılkaya Aydoğan, Özcan, & İlkay, 2017), bee algorithms and artificial bee colony algorithms (Tang, Li, & Zhang, 2016; Tapkan, Özbakır, & Baykasoğlu, 2016; Özbakır & Tapkan, 2011), iterated greedy search (IG) algorithms (Li, Tang, & Zhang, 2016; Li, Tang, & Zhang, 2017), and artificial fish swarm optimization algorithm (Zhong, Deng, & Xu, 2019), to cite just a few. Among these methods, IG algorithm produces the state-of-the-art results for TALBP when utilizing the best combination of encoding, decoding and objective function (Li, Kucukkoc, et al., 2017). A detailed comparative study of these metaheuristics refers to a recent comprehensive review paper by Li, Kucukkoc, et al. (2017). With regard to exact methods, Hu, Wu, et al. (2008) introduce a station-oriented enumerative algorithm to address small-size instances. Wu, Jin et al. (2008) employ a branch and bound (B&B) algorithm where the instances with up to 148 tasks are solved optimally. Xiaofeng, Erfei, et al. (2010) also employ a B&B algorithm where the largest-size instances with 205 tasks are solved. Nevertheless, this method might consume a large amount of running time, and the optimal solutions of the two cases are not achieved.

Regarding the exact methods, there are many exact methods on the simple assembly line balancing problem (SALBP) (Johnson, 1988; Morrison, Sewell, & Jacobson, 2014; Nourie & Venta, 1991; Pape, 2015; Scholl & Klein, 1997, 1999; Sewell & Jacobson, 2012; Vilà & Pereira, 2013), to cite just a few. Among these methods, the branch, bound and remember (BBR) algorithm (Morrison et al., 2014; Sewell & Jacobson, 2012) might be regarded as the state-of-the-art method as it solves all the well-known Scholl’s 269 instances optimally within very short running time. Due to the effectiveness of the BBR method, they have been extended to the variants of SALBP, including U-shaped assembly line balancing problem (Li, Kucukkoc, & Zhang, 2018; Yolmeh & Salehi, 2017), robust assembly line balancing problem (Pereira & Álvarez-Miranda, 2018), assembly line worker assignment and balancing problem (Pereira, 2018; Vilà & Pereira, 2014) and robotic assembly line balancing problem (Borba, Ritt, & Miralles, 2018), among others.

Among the aforementioned literature, there are only three papers working on exact methods for TALBP. However, only one is capable of addressing the largest-size instances with the cost of a large amount of running time. Furthermore, there is no research to extend and improve the BBR to address TALBP.

1.2 Contributions of the work

This research focuses on developing new exact methods to produce better results for TALBP. Specifically, this research extends and improves the BBR to solve TALBP with the objective of minimizing mated-station number. As will be presented in Section 2, the task sequence within one station does not matter as long as the precedence constraint is satisfied for SALBP. For TALBP, on the contrary, a different task sequence might result in different idle times. Thus, the task sequence within one station must be optimized. Moreover, direction constraints make the TALBP even more complicated, and hence the published exact methods for SALBP cannot solve the considered problem with minor modifications.

The proposed BBR method employs a modified Hoffman heuristic to construct a high-quality upper bound and cyclic best-first search strategy (CBFS) in the search process. As the TALBP is greatly different from SALBP due to the sequence-induced idle times and direction constraints, this proposed BBR utilizes an improved task enumeration procedure, new dominance rules and new lower bounds. This is the first time to apply Hoffman heuristic and BBR algorithm to TALBP. The computational study demonstrates the superiority of these improvements and shows that proposed BBR outperforms the B&B algorithm (Xiaofeng et al., 2010) and IG algorithm (which might be the current best heuristic algorithm (Li, Kucukkoc, et al., 2017)), by achieving more optimal solutions or consuming less computational time.

The remainder of this research is organized as follows. Section 2 describes the considered TALBP and Section 3 provides a detailed description of the proposed BBR methodology. Subsequently, the computational study is conducted in Section 4, where the proposed method is evaluated. Finally, the conclusions and several future research venues are presented in Section 5.

2. Problem description

TALBP can be, without loss of generality, described as a set of tasks I ($I = \{1,2,\dots,i,\dots,nt\}$) are allocated to a set of mated-stations J ($J = \{1,2,\dots,j,\dots,nm\}$) with one or several optimization criteria. There are three types of essential constraints needed to be considered: precedence constraint, cycle time constraint and direction constraint. Figure 1 and Fig. 2 illustrate an example taken from (Xiaofeng et al., 2010) with 16 tasks to highlight the features of TALBP, where Fig. 1 presents the precedence diagram and Fig. 2 shows the detailed task assignment. In Fig. 1, the nodes denote tasks, labels above nodes refer to the operation times and preferred directions, and arrows refer to the precedence relations, e.g. the arrow between task h and task i indicates that task i is an immediate successor of task h. The tasks in TALBP are portioned into three categories: L-type tasks with left-side direction (L in precedence diagram), R-type tasks with right-side direction (R in precedence diagram) and E-type tasks with either-side direction (E in precedence diagram).

Regarding the precedence constraint, the predecessors of a task should be allocated to the former mated-station or the same mated-station. When they are allocated to the same mated-station, the successor cannot begin until its predecessors have been completed. For instance, task 8 cannot begin when task 6 is completed as the predecessor task 7 of task 8 is not completed (see Fig. 2). The idle time resulting from the precedence constraint and the utilization of two sides is referred to as sequence-induced idle time. This can be reduced by optimizing the task sequence in each workstation (Li, Kucukkoc, et al., 2017), and hence this task sequence in one workstation cannot be ignored. As regard to cycle time constraint, all the tasks in each mated-station must be completed within the given cycle time (CT), e.g. all the tasks in Fig. 2 are finished within 18 time units. For the direction constraint, the L-type (R-type) tasks must be allocated to the left side (right side), e.g. task 3, task 6 and task 12 are allocated to left side, and E-type tasks can be allocated to either side.

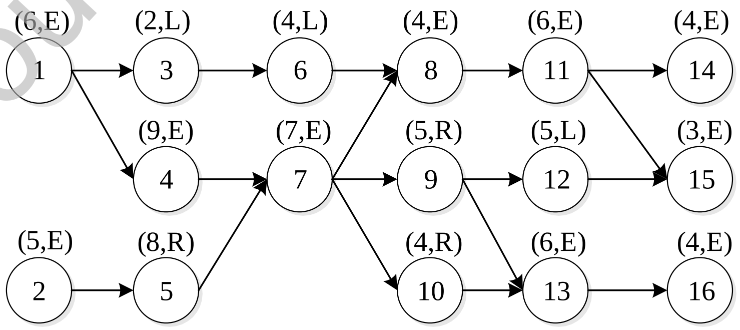


Fig. 1 Precedence diagram with 16 tasks, taken from (Xiaofeng et al., 2010)

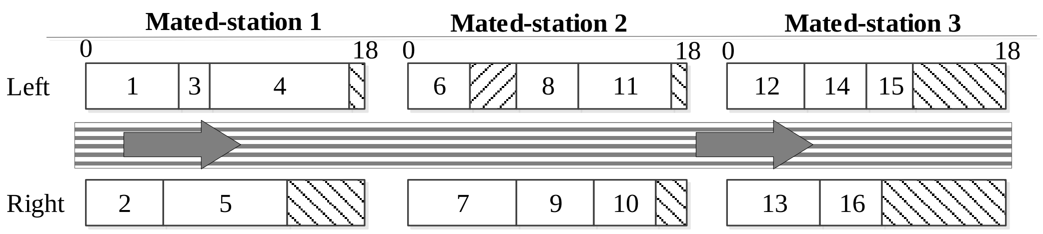


Fig. 2 Detailed task assignment with a cycle time of 18 time units

3. Proposed branch, bound and remember algorithm

BBR algorithm is an exact method developed by Sewell & Jacobson (2012), and the main characteristic of this methodology is utilizing memory-based dominance rule by storing all the explored sub-problems. This method produces the competing results for SALBP (Battaïa & Dolgui, 2013; Li, Kucukkoc, & Tang, 2019; Morrison et al., 2014; Sewell & Jacobson, 2012). Owing to the superiority of the BBR algorithm, this research puts the first attempt to extend this method to TALBP. Nevertheless, the difference between SALBP and TALBP is quite large due to the sequence-induced idle times and direction constraint. For instance, the task sequence in one workstation for SALBP does not matter as long as the precedence constraint is satisfied. The different task sequences in one workstation for TALBP, on the contrary, might result in different sequence-induced idle times, and thus the task sequence cannot be ignored. Due to these differences, task enumeration procedure and the dominance rules in SALBP need a big adjustment or even are not applicable. Hence, the proposed BBR method utilizes new dominance rules, new lower bounds and an improved task enumeration procedure.

The main procedure of the proposed BBR follows that in Sewell & Jacobson (2012), and is presented as follows. In this procedure, UB_{NM} is the upper bound on mated-station number, and LB_{NM} is the lower bound on mated-station number. This procedure consists of three phases, where Phase I utilizes the modified Hoffman heuristic to obtain high-quality UB_{NM}. Phase II utilizes CBFS to attempt to find the optimal solution. If Phase II fails to prove the optimality of the obtained solution, Phase III is conducted using breadth-first search strategy (BrFS).

```
Algorithm 1: Procedure of BBR algorithm for TALBP
% Start of Phase I
Step 1: Calculate the $LB_{NM}$ at the root.
Step 2: Obtain $UB_{NM}$ using modified Hoffman heuristic.
Step 3: If $UB_{NM} = LB_{NM}$, terminate this procedure; otherwise, conduct Step 4.
% Start of Phase II
Step 4: Execute CBFS and update $UB_{NM}$ when smaller $UB_{NM}$ is obtained. If the optimal
solution is achieved or the termination criterion is met, terminate this procedure.
% Start of Phase III
Step 5: Execute BrFS and update $UB_{NM}$ when smaller $UB_{NM}$ is obtained. If the optimal
solution is achieved or the termination criterion is met, terminate this procedure.
```

The procedure of executing search strategy CBFS or BrFS is illustrated in Algorithm 2. Here, a partial solution is referred to as $\wp = (A, U, M_1, M_2, \cdots, M_{nj})$, where nj is the utilized mated-station number in $\wp$, M_j denotes the set of tasks allocated to mated-station j, A is the set of allocated tasks ($A = \bigcup_{j=1}^{nj} M_j$) and U is the set of unallocated tasks ($U = I \backslash A$). Notice that one sub-problem Y is stored in memory for exploration at a later iteration after applying the $LB1_{NM}$, $LB2_{NM}$, $LB3_{NM}$ and all the dominance rules. $LB4_{NM}$ is applied only for the sub-problem in memory as $LB4_{NM}$ costs a lot of computation time following (Sewell & Jacobson, 2012) and (Morrison et al., 2014) (please check the codes on the webpage "*https://assembly-line-balancing.de*"). For a sub-problem Y, if $UB_{NM} \leq LB4_{NM}$, the sub-problem Y is not tested anymore whereas it is preserved in memory for memory-based rules.

```
Algorithm 2: The procedure of executing the search strategy
1   While $UB_{NM} > LB_{NM}$ and termination criterion not met
2       Select an unexplored and non-dominated partial solution $\wp$ $(A, U, M_1, M_2, \cdots, M_{nj})$
        with the utilized search strategy
3       While the task enumeration tree for partial solution $\wp$ is not empty
```

```
4    Obtain a new partial solution $Y\left(A', U', M_1, M_2, \cdots, M_{nj}, M_{nj+1}\right)$ at next depth with branching method
5    $UB_{NM} \leftarrow nj+1$ when $Y$ is complete solution and $UB_{NM} > nj+1$
6    Apply the maximal load rule and delete $Y$ if it is dominated
7    Calculate $LB1_{NM}$, $LB2_{NM}$ and $LB3_{NM}$ of sub-problem $Y$ and delete $Y$ when $UB_{NM} \leq max\{LB1_{NM}, LB2_{NM}, LB3_{NM}\}$
8    Apply the extended Jackson rule and no-successors rule and delete $Y$ if it is dominated
9    Apply the memory-based dominance rule, memory-based maximal load rule and memory-based extended Jackson rule and delete $Y$ if it is dominated
10   Save sub-problem $Y$ for exploration and usage in memory-based dominance rules
11   Calculate the $LB4_{NM}$ of sub-problem $Y$ and mark $Y$ as explored if $UB_{NM} \leq LB4_{NM}$
12     Endwhile
13   Endwhile
```

3.1 Branching

There are two popular branching methods: task-oriented branching (Johnson, 1988; Nourie & Venta, 1991) and station-oriented branching (Hoffmann, 1992; Scholl & Klein, 1997, 1999) when utilizing B&B algorithm to address SALBP. In the task-oriented branching method, the task sequences are enumerated and subproblems are generated by allocating tasks to the current workstation if there is plenty of remaining time, or to the next workstation when there is not enough remaining time. Regarding the station-oriented branching, the task assignments in workstations are enumerated and subproblems are created by allocating a set of tasks together or a complete load to the next workstation. Regarding the performances of the two branching methods, the station-oriented branching shows superior performance over task-oriented branching (Scholl & Klein, 1997, 1999; Sewell & Jacobson, 2012) in solving the SALBP.

Nevertheless, the published B&B algorithms (Wu et al., 2008; Xiaofeng et al., 2010) in solving the TALBP utilize only task-oriented branching and there are no applications of station-oriented branching. Inspired by the high performance of station-oriented branching in solving the SALBP, the proposed BBR method also employs the station-oriented branching. As the two-sided assembly line utilizes mated-stations and there are connections among tasks allocated to two sides inside one mated-station, this research proposes a modified station-oriented branching method, referred to as *mated-station-oriented branching*. Namely, sub-problems are created by allocating a set of tasks together or a complete load to the next mated-station.

3.1.1 Original task enumeration procedure

The original task enumeration procedure is presented as follows to obtain all the possible full loads for one mated-station. For one mated-station, the assignable tasks in depth 1 are generated and allocated at first, then the assignable tasks in depth 2 are generated and allocated and this procedure is terminated when no assignable task exists. Figure 3 presents the enumerated tree of an illustrated instance, which has four tasks and is taken from the precedence diagram in Fig. 1. This enumeration procedure is similar to that in (Wu et al., 2008), where the E-type tasks should be tested on both left side and right side. For instance, task 7 should be considered with 7L and 7R, respectively, when branching. Clearly, this enumeration tree ensures that all feasible task sequences are enumerated and tested. When obtaining the full loads for one mated-station, a branch is terminated when no unallocated tasks can be completed within the remaining time; the assigned tasks up to that point then form a full load. Clearly, the enumeration tree to generate full loads for one mated-station is much smaller than the full enumeration tree utilized in task-oriented branching by (Wu et al., 2008).

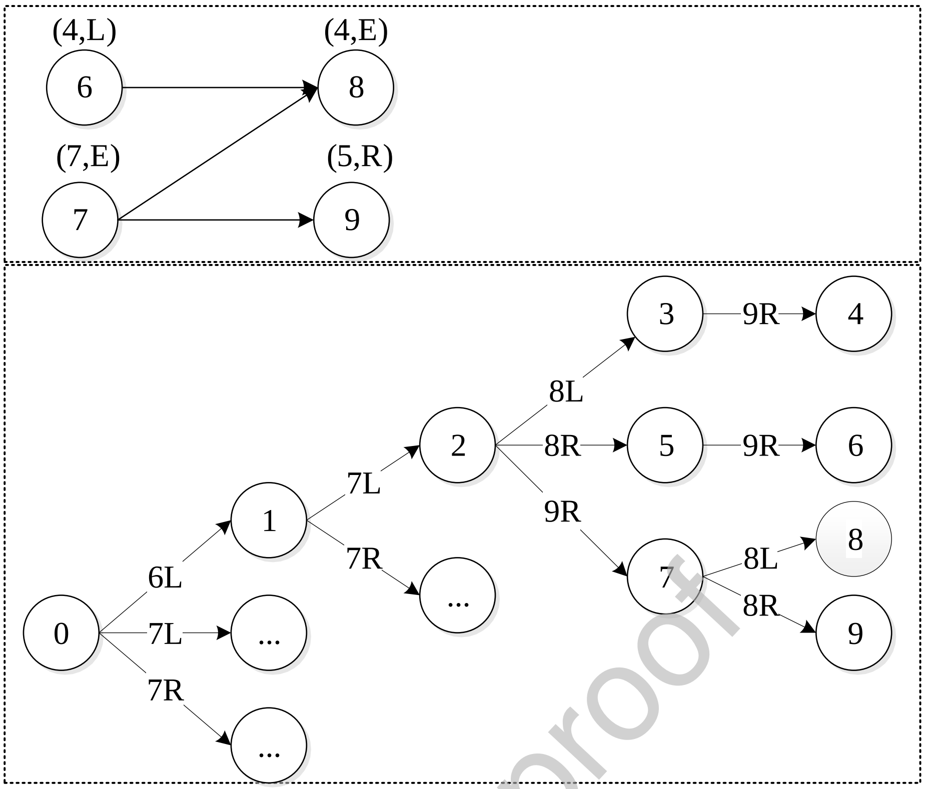


Fig. 3 The enumerated tree of the illustrated instance

3.1.2 Improved task enumeration procedure

The original task enumeration procedure above has two possible drawbacks in solving large-size instances: 1) there are many “poor” full loads with a lot of idle times and a lot of running time is wasted to prune these “poor” full loads; 2) there are many different task sequences for the tasks allocated to one mated-station and a tremendous amount of time is consumed in achieving all the full loads for one sub-problem. Hence, this study presents an improved task enumeration procedure to speed up the search process. The proposed improvements are proposed as follows to achieve the most promising full loads as soon as possible.

% Improvement 1

Step 1: The candidates (potential branching decisions) are divided into two sets: (i) Set

A contains L-type tasks with left side, R-type tasks with right side and E-type tasks with the side with more remaining time or left side when both sides have the same remaining time. (ii) Set B contains the remaining candidates.

% Improvement 2

Step 2: The candidates in Set A are further divided into two sets: (i) Set C contains the tasks with the smallest sequence-induced idle time when they are allocated to the corresponding workstation. (ii) Set D contains the tasks with more sequence-induced idle times.

% Improvement 3

Step 3: The candidates in Set C are renumbered, where tasks with larger ranked positional weight (Helgeson & Birnie, 1961) and operation time are enumerated first.

During enumeration, the candidates (potential branching decisions) in Set C are first enumerated, and the remaining candidates (in Set B and Set D) are enumerated only when all the candidates in Set C in each depth have been enumerated. Step 1 aims at quickly obtaining loads with balanced workloads on both sides of a mated-station. Step 2 tries to achieve high-quality loads with less sequence-induced idle time as soon as possible and Step 3 tries to allocate the "hard" tasks at first to have more tasks assignable. In short, all these improvements are in favor of achieving a set of high-quality mated-station loads as soon as possible. Additionally, the proposed improved task enumeration procedure terminates when the number of full loads reaches to a predetermined number (set to 10,000) when solving the large-size instances following (Sewell & Jacobson, 2012), (Morrison et al., 2014) and (Li et al., 2018). Notice that, if there is no limit of the number of achieved full loads, the improved task enumeration procedure is capable of achieving all the possible station loads.

In fact, if the three improvements are not utilized, there would be a lot of "poor" full loads and the BBR method would not obtain high-quality solutions for some instances. Also, if there is no limit on the number of full loads, the BBR method would be very slow and unable to obtain high-quality solutions within an acceptable amount of computation time. As you will see in Section 4.1, this improved task enumeration procedure enhances the performance of the BBR method by a significant margin in solving the large-size instances. The detailed applications of the two task enumeration procedures are clarified as follows.

1) Firstly, Phase I performs the improved task enumeration procedure with a limit of 10,000 station loads to obtain a high-quality upper bound.
2) Subsequently, Phase II conducts the improved task enumeration procedure with a limit of 10,000 station loads to obtain a set of high-quality full loads as soon as possible.
3) Finally, Phase III performs the original task enumeration procedure (or improved task enumeration procedure) without limitation on the station loads to achieve all the possible station loads.

Hence, the proposed BBR algorithm is an exact method which is capable of achieving the optimal solutions and verifying the optimality of the achieved solutions when solving both small-size and large-size instances.

3.2 Upper bound and lower bounds

The proposed BBR proposes a modified Hoffman heuristic (MHH) to achieve UB_{NM} before branching. MHH utilizes the same procedure as the Hoffman heuristic (Hoffmann, 1963), where the solution is built for one mated-station at a time. MHH first generates all the possible or a predetermined number of mated-station loads for the first mated-station, and then selects the best one as the task assignment to the current mated-station. Then MHH generates a set of full loads and selects the best one for the second mated-station, and this procedure is terminated when all the tasks are allocated. The mated-station number of the achieved solution is regarded as UB_{NM}. Following Sewell & Jacobson (2012), the proposed MHH selects a load with the maximum value of $\sum_{i \in CM}(t_i + \alpha \cdot w_i + \beta \cdot |S(i)| - \gamma - \gamma \cdot D_i)$. In this expression, CM is the set of tasks that can be allocated to the current new mated-station, $S(i)$ ($S_a(i)$) refers to the set of immediate (all) successors of task i, w_i is the positional weight of task i ($w_i = t_i + \sum_{h \in S_a(i)} t_h$), D_i is set to 1 when task i is E-type task and 0 when task i is L-type or R-type task. The rationale of this expression is clarified as follows. The t_i, w_i and $|S(i)|$ encourage the task with the larger operation time, the larger positional weight and the larger number of successors. The term $-\gamma$ and term $-\gamma \cdot D_i$ encourage the task with a larger operation time and with left direction or right direction respectively, aiming at making the remained tasks easier to pack.

There are three parameters, namely α, β and γ. The values of these parameters are set based on published papers: $\alpha \in \{0, 0.005, 0.01, 0.015, 0.02\}$, $\beta \in \{0, 0.005, 0.01, 0.015, 0.02\}$ and $\gamma \in \{0, 0.01, 0.02, 0.03\}$. The proposed MHH is run using each combination of these values, and the best one is regarded as the final solution by MHH. To avoid much time in generating mated-station load, the number of mated-station loads is limited to 1,000. In addition, the improvements introduced in Section 3.1 are also applied to achieve the most promising full loads as soon as possible.

The lower bounds in SALBP are not applicable to TALBP, and hence this research modifies lower bounds in SALBP to suit TALBP. This research builds the lower bounds for TALBP based on three standard lower bounds (Scholl & Klein, 1997), LB1, LB2 and LB3, and a tighter lower bound, BPLB, by solving the bin packing problem (Sewell & Jacobson, 2012). Here, BPLB is achieved by a separate branch-and-bound solver to solve the bin packing problem and computation time for this branch-and-bound solver is limited to 1 second (see (Sewell & Jacobson, 2012) and (Morrison et al., 2014)). The three standard lower bounds have been widely applied, and they are clarified using expressions (1-4), where T is a set of tasks. BPLB is developed by Sewell & Jacobson (2012) by relaxing SALBP into the bin packing problem and the solution by solving the bin packing problem is regarded as the BPLB.

$$\text{LB1} = \lceil \textstyle\sum_{i \in T} t_i / CT \rceil \tag{1}$$

$$\text{LB2} = |\{i \in T | t_i > CT/2\}| + \left\lceil \frac{|\{i \in T | t_i = CT/2\}|}{2} \right\rceil \tag{2}$$

$$\text{LB3} = \left\lceil \sum_{i \in T} w_i \right\rceil \tag{3}$$

$$w_i = \begin{cases} 1 \ \text{ if } \ t_i > 2 \cdot CT/3 \\ 2/3 \ \text{ if } \ t_i = 2 \cdot CT/3 \\ 1/2 \ \text{ if } \ CT/3 < \ t_i < 2 \cdot CT/3 \\ 1/3 \ \text{ if } \ t_i = CT/3 \end{cases} \tag{4}$$

To apply these lower bounds to TALBP, this research first defines four operations, LB1(T), LB2(T), LB3(T) and BPLB(T), denoting the achieved LB1, LB2, LB3 and BPLB for task set T. Subsequently, there are four lower bounds on mated-station number: $LB1_{NM}$, $LB2_{NM}$, $LB3_{NM}$ and $LB4_{NM}$. These lower bounds are calculated with expressions (5-8), where I is the set of remained tasks, and AL or AR is the set of remained tasks with left or right direction. Recall that $LB1_{NM}$ is equivalent to the lower bound in Hu et al. (2008) and lower bound 1 in Wu et al. (2008). $LB3_{NM}$ is equivalent to the lower bound 2 in Wu et al. (2008). For a new sub-problem, $LB1_{NM}$, $LB2_{NM}$ and $LB3_{NM}$ are first applied. If the sub-problem achieves LB_{NM} equal to or larger than UB_{NM}, this sub-problem is pruned. If this new sub-problem cannot be pruned by these three lower bounds, $LB4_{NM}$ is applied as $LB4_{NM}$ consumes much more running time.

$$LB1_{NM} = \max\left\{\left\lceil\frac{\mathrm{LB1(I)}}{2}\right\rceil, \mathrm{LB1(AL)}, \mathrm{LB1(AR)}\right\} \tag{5}$$

$$LB2_{NM} = \max\left\{\left\lceil\frac{\mathrm{LB2(I)}}{2}\right\rceil, \mathrm{LB2(AL)}, \mathrm{LB2(AR)}\right\} \tag{6}$$

$$LB3_{NM} = \max\left\{\left\lceil\frac{\mathrm{LB3(I)}}{2}\right\rceil, \mathrm{LB3(AL)}, \mathrm{LB3(AR)}\right\} \tag{7}$$

$$LB4_{NM} = \max\left\{\left\lceil\frac{\mathrm{BPLB(I)}}{2}\right\rceil, \mathrm{BPLB(AL)}, \mathrm{BPLB(AR)}\right\} \tag{8}$$

3.3 Dominance rules

The maximal load rule and Jackson dominance rule cannot be directly applied to TALBP due to sequence-induced idle time. An illustrated example is presented in Fig. 4 to show why the general maximal load rule cannot be directly applied to TALBP. Let us assume that there is an unallocated E-type task j whose predecessors have been allocated. Task i is the predecessor of task j and there is enough time remaining to complete task j on the left side. For the SALBP, the partial solution is pruned when the remaining time is larger than or equal to the operation time of task j. Nevertheless, this partial solution cannot be pruned for TALBP as task i is the predecessor of task j and the remaining time on the right side is not enough to complete task j. Hence, the remaining capacities of both sides should be considered at the same time when applying the maximal load rule. Similarly, it is also necessary to consider the remaining capacities of both sides when applying the Jackson dominance rule.

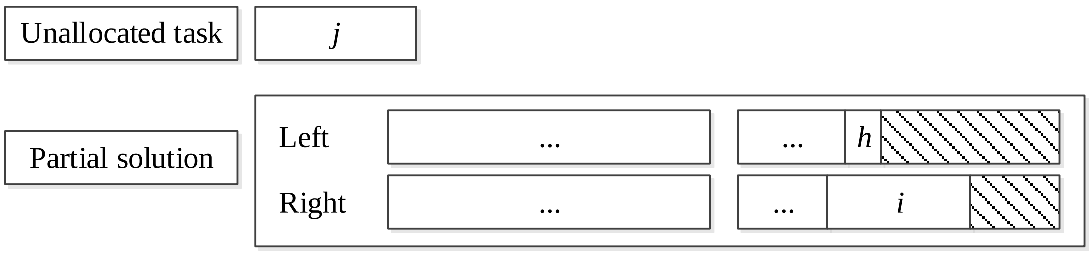


Fig. 4 An illustrated example of applying the maximal load rule

There are two possible situations for adapting the maximal load rule: 1) both sides have enough remaining time to complete task j; 2) only one side has enough remaining time to complete task j. For the first situation, the maximal load rule can be applied directly. For the second situation, a partial solution can be pruned satisfying two conditions: 1) only one station has enough remaining time to complete task j; 2) the sum of the operation time of task j and the maximum completion time of the predecessors of task j on this mated-station is not larger than the cycle time. Clearly, the second method needs to store the completion times of all tasks on this mated-station and also consumes a lot of time to check whether the second condition is satisfied.

The first method to handle the first situation is quick and easy to be implemented, but it might not be able to prune some sub-problems. Luckily, the memory-based dominance rule can remedy this drawback by storing all the explored sub-problems, and thus this paper employs two new dominance rules: memory-based maximal load rule and memory-based extended Jackson rule. Taking the memory-based maximal load rule as an example, for a sub-problem $\wp$

with assigned tasks A, if a stored sub-problem with the same mated-station number has a set of assigned tasks A' that satisfies $A \subseteq A'$, the sub-problem $\wp$ is pruned. Note that the memory-based maximal load rule may not take full advantage of potential pruning because it needs to check the sub-problems $\Upsilon\left(A', U', {M_1}', {M_2}', \cdots, {M_{nj}}'\right)$ that have already been generated in order to prune the sub-problem $\wp\left(A, U, M_1, M_2, \cdots, M_{nj}\right)$. Namely, this is a tradeoff between more pruning and faster running time.

The applied dominance rules are presented as follows, and they are developed based on the dominance rules for SALBP (Morrison et al., 2014; Sewell & Jacobson, 2012). Recall that, the dominance rules are applied once a full station load is achieved during the task enumeration procedure. When applying these dominance rules, the maximal load rule, the extended Jackson rule and the no-successor rule are first applied. If this sub-problem cannot be pruned, the memory-based rules are applied successively.

Maximal load rule: A partial solution is pruned if 1) this partial solution contains a station load $\bigcup_{j=1}^{nj} M_j$, and 2) there is an unallocated task whose predecessors have been allocated to the former j mated-stations, 3) the remaining times on both sides of the last mated-station are larger than or equal to the operation time of this task.

Extended Jackson rule: A partial solution is pruned if 1) the set of tasks allocated to the last mated-station contains a task i and there is task h such that $h \in \left\{c \middle| c \in I - \left(P_a(i) \cup S_a(i)\right)\right\}$, $t_i \leq t_h$ and $S_a(i) \subseteq S_a(h)$, 2) task i and task h have the same preferred direction, 3) task h can replace task i without violating precedence constraint, 4) both the remaining times in both sides are larger than or equal to $t_h - t_i$. Recall that $P_a(i)$ and $S_a(i)$ denote the set of all predecessors or successors of task i, respectively.

No-successor rule: A partial solution is pruned if 1) the set of tasks allocated to the last mated-station has no successors, and 2) there exists an unallocated task which has one or several successors.

Memory-based dominance rule: A partial solution is pruned if this partial solution has the same assigned tasks and no smaller mated-station number as a previously-explored sub-problem.

Memory-based maximal load rule: A partial solution is pruned if 1) its set of allocated tasks is a subset of the allocated tasks in a previously explored subproblem; 2) the number of its utilized mated-stations is not less than that of this previously-explored sub-problem.

Memory-based extended Jackson rule: A partial solution is pruned if 1) this partial solution contains the same allocated tasks as a previously explored sub-problem except task i in this partial solution and task h in the previously explored sub-problem, 2) $h \in \{c | c \in I - (P_a(i) \cup S_a(i))\}$, $t_i \leq t_h$ and $S_a(i) \subseteq S_a(h)$, 3) task i and task h have the same preferred direction or task i is an E-type task.

3.4 Search strategy

Search strategy has a great impact on the speed of a B&B algorithm by determining the sequence of sub-problems being explored. If the most promising sub-problem is first explored, new upper bound might be achieved very fast and many subproblems with lower bounds no smaller than the upper bounds can be pruned. There are mainly four search strategies in B&B algorithms: depth-first search strategy (DFS), best-first search strategy (BFS), breadth-first search strategy (BrFS) and cyclic best-first search strategy (CBFS).

DFS explores one sub-problem at depth 1, one at depth 2 and finally one at the deepest level of the search tree. Afterwards, DFS returns to the top of the tree after exhausting all sub-problems within that sub-tree. DFS has a fast speed to achieve a complete solution, and among the methods utilizing different types of DFS (Hoffmann, 1992; Johnson, 1988; Scholl & Klein, 1997, 1999), SALOME (Scholl & Klein, 1997, 1999) produces the best results. BFS selects the best sub-problem regarding the selection criterion from a set of unexplored sub-problems. To achieve a complete solution quickly, the selection criterion can be designed to select the sub-problem with the maximum number of utilized workstations. BrFS is the slowest strategy to achieve a complete solution, and sometimes it might even hardly achieve a complete solution before the termination criterion is met. This method first generates all the sub-problems at depth 1, and then explores the sub-problems and generates all the subproblems at depth 2, and so on. The procedure is terminated after generating all the solutions at the deepest depth. CBFS is a recently developed strategy by hybridizing DFS and BFS (Sewell & Jacobson, 2012). CBFS first selects and explores one best sub-problem at depth 1, and subsequently selects and explores one best sub-problem at the next depth until one best sub-problem at the deepest depth is explored. It then returns to the highest level and this cycle is repeated until a termination criterion is satisfied. The main difference between DFS and CBFS is that CBFS selects the most promising sub-problem to be explored at each depth.

This research proposes CBFS as the search strategy as BBR with CBFS has produced the state-of-the-art results for SALBP (Sewell & Jacobson, 2012). The main procedure of CBFS strategy is presented in Algorithm 3. For a sub-problem $\wp = (A, U, M_1, M_2, \cdots, M_{nj})$, the sub-problem selection criterion is set as $b(\wp) = LB_{NM}(U) + TI/nj - \lambda/|U|$ following (Sewell & Jacobson, 2012), where TI is the total idle in the former nj mated-station and λ

is an input parameter (set to 0.02). In this expression, $LB_{NM}(U)$ encourages the sub-problem with smaller lower bounds, TI/nj encourages full loads and $-\lambda/|U|$ encourages tasks with larger operation times. Among a set of sub-problems at each depth, CBFS selects the one with the minimum value of $b(*)$.

```
Algorithm 3: Procedure of CBFS strategy
$k = 0$;
While (there exists unexplored sub-problem at any depth)
    $k = k + 1$; $k = k\%(UB_{NM} - 1)$;
    If (there is no unexplored sub-problem at depth $k$)
        Continue;
    Else
        Select a sub-problem $\wp$ at depth $k$ with the minimum value of $b(*)$;
        Generate and store the children of the selected sub-problem $\wp$ at depth $k + 1$;
    Endif
Endwhile
```

4. Computational study

To evaluate the proposed improvements and the proposed BBR algorithm, this section first tests these improvements on the performance of BBR algorithm, and subsequently compares the BBR algorithm with the B&B algorithm of (Xiaofeng et al., 2010) and the current best heuristic algorithm, IG algorithm (Li, Tang, et al., 2017).

All the well-known benchmark problems for TALBP are solved, including the instances in (Li, Tang, et al., 2017) and the instances in (Xiaofeng et al., 2010). These tested benchmarks consist of four small-size problems (P9, P12, P16 and P24) and three large-size problems (P65, P148 and P205). The precedence diagrams of P9, P12 and P24 are taken from (Kim et al., 2000); the

precedence diagrams of P16, P65 and P205 are taken from (Lee, Kim, & Kim, 2001) and the precedence diagram of P148 is taken from (Bartholdi, 1993), where the operation times of task 79 and task 108 are changed as suggested by (Lee et al., 2001). As each problem has several different cycle times, there are a total number of 59 instances. Recall that the benchmark instances in SALBP are different from that in TALBP as there are direction constraints in TALBP. The algorithms are coded in C++ on the platform of Microsoft Visual Studio 2015 and run on a personal computer equipped with Intel(R) Core (TM) i7-4790S 3.20GHZ CPU and 8.00GB RAM.

4.1 Evaluation of the improvements

This section tests the improvements presented in Section 3, and the proposed algorithm is compared to other algorithms where these improvements are selectively disabled (the original branch, bound and remember algorithms). These compared algorithms are presented as follows.

1) BBR-oldEnu: The original task enumeration procedure in Fig. 3 is utilized in Phase I and Phase II, and the original task enumeration procedure in Phase II terminates after achieving all the possible mated-station loads.

2) BBR-limOldEnu: The original task enumeration procedure in Fig. 3 is utilized in Phase I and Phase II, and the original task enumeration procedure in Phase II terminates when the number of full loads reaches a predetermined number (set to 10,000).

3) BBR-noRenumber: The tasks are enumerated in the original order in Phase II when utilizing the improved task enumeration procedure.

4) BBR-noMHH: The MHH or Phase I is not utilized to achieve initial upper bounds.

5) BBR-noBPLB: The BPLB is not utilized as the lower bound in Phase II.

6) BBR-noMem: Memory-based dominance rule, memory-based maximal load rule and

memory-based extended Jackson rule are not utilized in Phase II.

7) BBR-BFS: BFS is utilized as the search strategy in Phase II.

To evaluate the performance on different instances, the relative percentage deviation or RPD is employed using the expression (9). In this expression, UB_{NM} is the achieved upper bound on mated-station number, and LB_{NM} is the lower bound on mated-station number, where LB_{NM} is updated during the search process. If RPD is equal to 0.0, the achieved upper bound is the optimal solution.

$$RPD = 100 \cdot (UB_{NM} - LB_{NM})/LB_{NM} \tag{9}$$

All the algorithms are tested under three maximum running times (10 s, 50 s and 100 s), where BBR terminates when the optimal solution is achieved or computational time reaches the given maximum time. Table 1 illustrates the number of instances solved optimally (#OPT), detailed average RPD values and average running times. From this table, it is observed that proposed BBR is the fastest method which achieves all the optimal solutions when the maximum time is equal to 10s, 50s or 100s. The BBR-oldEnu and the BBR-limOldEnu are the two worst performers, and they cannot achieve new optimal solutions with increased running time. This finding demonstrates the superiority of the improved task enumeration procedure over the original task enumeration procedure. In fact, this superiority is because the improved task enumeration procedure is capable to reduce the sequence-induced idle times effectively. It is also observed that BBR-noMem cannot achieve all the optimal solutions with increased running time, demonstrating the superiority of the proposed memory-based rules. As for other improvements, they show slightly better performance. For instance, if BPLB is not utilized, BBR will cost more running time to achieve all the optimal solutions. Notice that BPLB can obtain tighter lower bounds with the cost of a larger computation time, and the benefit is saving the time by pruning more partial solutions. If the consumed time to calculate the BPLB is larger than the time saved by pruning more partial solutions, the utilization of BPLB will result in a larger computation time; otherwise, the utilization of BPLB will reduce the computation time. Whether BPLB will help reduce the computation time might differ from one instance to another as observed in preliminary experiments. Still, as BPLB is capable of obtaining tighter lower bounds and might verify the optimality of some special instances, the BPLB is utilized in this study. In summary, this comparative study demonstrates the superiority of the proposed improvements, and the improved task enumeration procedure and the memory-based rules among these improvements are the most important factors which affect the performance of the BBR algorithm by a significant margin.

Table 1 Results by BBR algorithms

Maximum time		10s			50s			100s	
Evaluation index	#OPT	RPD	Time	#OPT	RPD	Time	#OPT	RPD	Time
BBR-oldEnu	30	9.31	5.57	30	9.31	25.18	30	9.31	49.93
BBR-limOldEnu	35	7.20	5.04	35	6.80	21.19	35	6.56	41.49
BBR-noRenumber	57	0.43	0.59	58	0.19	1.51	58	0.19	2.31
BBR-noMHH	58	0.85	2.06	58	0.85	2.70	58	0.85	3.62
BBR-noBPLB	59	0.00	0.67	59	0.00	0.67	59	0.00	0.67
BBR-noMem	55	1.05	0.91	56	0.77	3.29	56	0.77	5.84
BBR-BFS	57	0.43	0.67	59	0.00	1.19	59	0.00	1.19
Proposed BBR	59	0.00	**0.64**	59	0.00	**0.64**	59	0.00	**0.64**

**Best in bold.*

4.2 Comparative study

This section presents the comparative study among BBR, B&B and IG algorithms. Notice that IG is the best heuristic algorithm when utilizing the best combination of the decoding scheme and objective function, but this algorithm might show poor performance when utilizing low-quality decoding scheme and objective function (Li, Kucukkoc, et al., 2017). Hence, the proposed IG utilizes the best combination of the decoding scheme and objective function presented in (Li, Tang, et al., 2017). The tested BBR terminates when the optimal solution is achieved or computational time reaches 500 s. IG algorithm runs each instance for 10 times with the termination criterion of achieving the mated-station number equal to the lower bound at the root or running time reaches 500 s. The results of the B&B algorithm, which was coded and run on a Pentium-4 2.66 GHz personal computer equipped with 2 GB of memory, are directly taken from the literature (Xiaofeng et al., 2010).

The detailed results are exhibited in Table 2, where LB_{NM}-*root* reports the lower bound at root,

OPT refers to the optimal mated-station number and UB_{NM} and LB_{NM} are the upper bound and lower bound by the proposed BBR method. NM-Avg and Time-Avg are the average mated-station number and the average running time with a time unit of second by IG algorithm in ten times' running. Among these instances, Phase II is capable of verifying the optimality when solving P9, P12, P16 and P24 by testing all the station loads where the branching limit is not reached. Nevertheless, the branching limit is reached for P65, P148 and P205. Hence, Phase II tries to prove the optimality of the obtained solutions by checking LB_{NM} at the root when solving P65, P148 and P205.

Table 2 Computational results by algorithms

Problem	CT	LB_{NM}-*root*	OPT	B&B algorithm			IG		MHH		BBR		
				UB_{NM}	LB_{NM}	Time	NM-Avg	Time-Avg	NM	Time	UB_{NM}	LB_{NM}	Time
P9	3	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	4	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	5	2	2	-	-	-	2	0.00	2	0.00	2	2	0.00
	6	2	2	-	-	-	2	0.00	2	0.00	2	2	0.00
	7	2	2	-	-	-	2	0.00	2	0.00	2	2	0.00
P12	4	4	4	4	4	0.02	4	0.00	4	0.00	4	4	0.00
	5	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	6	3	3	3	3	0.02	3	0.00	3	0.00	3	3	0.00
	7	2	2	-	-	-	2	0.00	2	0.00	2	2	0.00
	8	2	2	-	-	-	2	0.00	2	0.00	2	2	0.00
	9	2	2	-	-	-	2	0.00	2	0.00	2	2	0.00
P16	15	3	4	-	-	-	4	500.00	4	0.01	4	4	0.85
	16	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	18	3	3	3	3	0.02	3	0.00	3	0.00	3	3	0.00
	19	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	20	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	21	2	3	-	-	-	3	500.00	3	0.00	3	3	0.85
	22	2	2	2	2	0.02	2	0.00	2	0.00	2	2	0.00
P24	18	4	4	-	-	-	4	0.01	4	0.00	4	4	0.00
	20	4	4	-	-	-	4	0.00	4	0.00	4	4	0.00
	24	3	3	-	-	-	3	0.10	4	0.27	3	3	1.19
	25	3	3	3	3	0.14	3	0.01	3	0.00	3	3	0.00
	30	3	3	-	-	-	3	0.00	3	0.00	3	3	0.00
	35	2	2	-	-	-	2	0.01	2	0.00	2	2	0.00
	40	2	2	2	2	0.12	2	0.01	2	0.00	2	2	0.00
P65	326	8	8	-	-	-	8	0.25	9	1.39	8	8	2.80
	381	7	7	7	7	0.34	7	0.23	8	1.41	7	7	0.02
	435	6	6	-	-	-	6	0.24	7	1.37	6	6	2.79
	490	6	6	6	6	17.2	6	0.27	6	0.01	6	6	0.01
	512	5	5	6	5	173551	5	14.54	6	1.16	**5**	**5**	8.96
	544	5	5	5	5	45	5	0.24	5	0.01	5	5	0.01
P148	204	13	13	13	13	30.2	13	6.18	13	0.05	13	13	0.05
	228	12	12	12	12	13257	12	6.58	12	0.05	12	12	0.06
	255	11	11	11	11	10.06	11	6.95	11	0.05	11	11	0.05
	306	9	9	-	-	-	9	6.45	9	0.04	9	9	0.04
	357	8	8	8	8	48.13	8	7.01	8	0.03	8	8	0.04
	378	7	7	8	7	184464	7	6.71	7	0.03	**7**	**7**	0.04
	408	7	7	7	7	36.4	7	6.71	7	0.03	7	7	0.04
	454	6	6	6	6	8065	6	6.62	6	0.04	6	6	0.04
	459	6	6	-	-	-	6	6.74	6	0.03	6	6	0.04
	510	6	6	6	6	18.43	6	7.77	6	0.03	6	6	0.04
P205	1133	11	11	-	-	-	11	18.75	11	0.82	11	11	0.19
	1275	10	10	10	10	17.87	10	19.09	10	0.04	10	10	0.04
	1322	9	9	-	-	-	9	68.59	10	3.57	9	9	9.05
	1455	9	9	9	9	16.08	9	18.79	9	0.03	9	9	0.03
	1510	8	8	-	-	-	8	18.32	9	3.00	8	8	0.29
	1650	8	8	8	8	27683	8	18.56	8	0.03	8	8	0.03
	1699	7	7	-	-	-	7	18.35	8	2.81	7	7	9.75
	1888	7	7	-	-	-	7	17.66	7	0.03	7	7	0.03
	1920	7	7	7	7	44.3	7	18.70	7	0.03	7	7	0.03
	2077	6	6	-	-	-	6	17.85	6	0.02	6	6	0.02
	2100	6	6	6	6	19753	6	18.21	6	0.05	6	6	0.03
	2266	6	6	-	-	-	6	18.73	6	0.03	6	6	0.04
	2300	6	6	6	6	61.19	6	20.00	6	0.03	6	6	0.03
	2454	5	5	-	-	-	5	19.37	5	1.77	5	5	0.02
	2500	5	5	5	5	24591	5	18.87	5	0.02	5	5	0.02
	2643	5	5	-	-	-	5	19.05	5	0.04	5	5	0.03
	2800	5	5	5	5	4.2	5	18.18	5	0.03	5	5	0.04
	2832	5	5	-	-	-	5	18.42	5	0.03	5	5	0.04

It is observed that BBR achieves all the optimal solutions for all the tested cases. B&B algorithm, on the contrary, cannot find the optimal solutions for two cases, P65 with a cycle time of 512 and P148 with a cycle time of 378. Regarding the running time, the proposed BBR method can solve all the instances within 10s optimally and the average running time by BBR is only 0.64s. Clearly, BBR is an effective and efficient methodology by outperforming the B&B algorithm in achieving a greater number of optimal solutions.

IG is also capable of achieving all the optimal solutions in each run, but it cannot prove the optimality of two cases, P16 with a cycle time of 15 and P16 with a cycle time of 21. The overall average time by IG algorithm in solving all the instances is 24.90 s while the overall average time by BBR is only 0.64s. BBR shows clear superiority in search speed by consuming less running time. As time limit might overly penalize IG by artificially inflating its average running time, this study also compares the average times by the IG algorithm and BBR method in solving the instances where the optimal mated-station number is equal to the LB_{NM} at root (P16-15 and P16-21 are not included). The average time by IG algorithm in solving these instances is 8.23s where the average time by BBR is only 0.87s. Again, BBR shows clear superiority in search speed. As for MHH, it achieves 53 optimal solutions out of 59 instances or 89.83% optimal solutions with a running time of 0.21s on average.

Notice that Phase III was not needed in solving these instances as Phase II was capable of proving the optimality of all the obtained solutions. However, in preliminary experiments, it was observed that Phase III was capable of obtaining the optimal solutions and verifying the optimality when Phase II was not conducted. The reason behind the utilization of Phase III is that Phase II might not verify the optimality of the obtained solutions as the improved task enumeration procedure in Phase II terminates upon reaching a predetermined number of station loads. The original task enumeration procedure (or improved task enumeration procedure) in Phase III, on the contrary, achieves all the station loads and hence ensures that the proposed BBR method is an exact method in theory.

In summary, the proposed BBR outperforms B&B in both solution quality and speed, and BBR outperforms IG algorithm in search speed. BBR is capable of finding all the optimal solutions within 1.0 s on average and is the state-of-the-art methodology for TALBP with mated-station minimization criterion.

5. Conclusion and future research

This research presents a new BBR algorithm to solve the TALBP with the mated-station number minimization criterion. The proposed algorithm utilizes the modified Hoffman heuristic to obtain a high-quality initial solution as the upper bound. Due to the specific characteristic of TALBP, the task enumeration procedure and dominance rules in the simple assembly line balancing problem (SALBP) are ineffective or not applicable to TALBP. Hence, this paper proposes an improved task enumeration procedure, modifies the dominance rules for SALBP and develops two new dominance rules: memory-based maximal load rule and memory-based extended Jackson rule. This algorithm also employs several other improvements, including renumbering the tasks, new lower bounds and a new criterion to select the most promising sub-problem. This algorithm is compared with the current best exact method, the B&B algorithm (Xiaofeng et al., 2010), and the current best heuristic algorithm, IG algorithm (Li, Tang, et al., 2017). Computational results demonstrate the superiority of these improvements and show that the BBR method outperforms the compared ones by achieving more optimal solutions within less computational time. The BBR method can find and verify optimal solutions for all tested instances in 1.0 second on average, and it can be regarded as the state-of-the-art methodology for TALBP with the mated-station number minimization criterion.

Future research might develop more dominance rules based on the characteristics of the TALBP. As the minimization of the workstation number is also important in industry, it is necessary to extend the proposed BBR algorithm to minimize both the mated-station number and the workstation number simultaneously. As the considered problem is the basic edition of TALBP, it is suggested to utilize the proposed BBR to address other variants of TALBP. As current exact methods mainly consider one optimization criterion, it is quite interesting to extend the proposed BBR or hybridize it with other metaheuristics to address multi-objective TALBP.

Acknowledgment

The authors are grateful for the anonymous referees for their constructive comments to improve this paper. This project is partially supported by National Natural Science Foundation of China under grant 61803287 and China Postdoctoral Science Foundation under grant 2018M642928.

References

Abdullah Make, M. R., Ab. Rashid, M. F. F., & Razali, M. M. (2017). A review of two-sided assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology, 89*(5-8), 1743-1763. doi:10.1007/s00170-016-9158-3

Bartholdi, J. J. (1993). Balancing two-sided assembly lines: a case study. *International Journal of Production Research, 31*(10), 2447-2461. doi:10.1080/00207549308956868

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, *142*(2), 259-277.

Baykasoglu, A., & Dereli, T. (2008). Two-sided assembly line balancing using an ant-colony-based heuristic. *The International Journal of Advanced Manufacturing Technology, 36*(5), 582-588. doi:10.1007/s00170-006-0861-3

Borba, L., Ritt, M., & Miralles, C. (2018). Exact and heuristic methods for solving the Robotic Assembly Line Balancing Problem. *European Journal of Operational Research, 270*(1), 146-156. doi:10.1016/j.ejor.2018.03.011

Chutima, P., & Chimklai, P. (2012). Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers & Industrial Engineering, 62*(1), 39-55. doi:http://dx.doi.org/10.1016/j.cie.2011.08.015

Delice, Y., Kızılkaya Aydoğan, E., & Özcan, U. (2016). Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach. *International Journal of Production Research, 54*(11), 3429-3451. doi:10.1080/00207543.2016.1140918

Delice, Y., Kızılkaya Aydoğan, E., Özcan, U., & İlkay, M. S. (2017). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing, 28*(1), 23-36. doi:10.1007/s10845-014-0959-7

Helgeson, W., & Birnie, D. (1961). Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering, 12*(6), 394-398.

Hoffmann, T. R. (1963). Assembly Line Balancing with a Precedence Matrix. *Management Science, 9*(4), 551-562. doi:10.1287/mnsc.9.4.551

Hoffmann, T. R. (1992). Eureka: A Hybrid System for Assembly Line Balancing. *Management Science*, *38*(1), 39-47. doi:10.1287/mnsc.38.1.39

Hu, X., Wu, E., & Jin, Y. (2008). A station-oriented enumerative algorithm for two-sided assembly line balancing. *European Journal of Operational Research, 186*(1), 435-440. doi:http://dx.doi.org/10.1016/j.ejor.2007.01.022

Johnson, R. V. (1988). Optimally Balancing Large Assembly Lines with "Fable". *Management Science, 34*(2), 240-253. doi:10.1287/mnsc.34.2.240

Khorasanian, D., Hejazi, S. R., & Moslehi, G. (2013). Two-sided assembly line balancing considering the relationships between tasks. *Computers & Industrial Engineering, 66*(4), 1096-1105. doi:http://dx.doi.org/10.1016/j.cie.2013.08.006

Kim, Y. K., Kim, Y., & Kim, Y. J. (2000). Two-sided assembly line balancing: A genetic algorithm approach. *Production Planning & Control, 11*(1), 44-53. doi:10.1080/095372800232478

Kim, Y. K., Song, W. S., & Kim, J. H. (2009). A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers & Operations Research, 36*(3), 853-865.

doi:http://dx.doi.org/10.1016/j.cor.2007.11.003

Kucukkoc, I., & Zhang, D. Z. (2015a). A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem. *Production Planning & Control, 26*(11), 874-894. doi:10.1080/09537287.2014.994685

Kucukkoc, I., & Zhang, D. Z. (2015b). Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. *Computers & Industrial Engineering, 84,* 56-69. doi:http://dx.doi.org/10.1016/j.cie.2014.12.037

Lee, T. O., Kim, Y., & Kim, Y. K. (2001). Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering, 40*(3), 273-292. doi:http://dx.doi.org/10.1016/S0360-8352(01)00029-8

Li, Z., Kucukkoc, I., & Nilakantan, J. M. (2017). Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. *Computers & Operations Research, 84*, 146-161. doi:https://doi.org/10.1016/j.cor.2017.03.002

Li, Z., Kucukkoc, I., & Tang, Q. (2019). A comparative study of exact methods for the simple assembly line balancing problem. *Soft Computing*. doi:10.1007/s00500-019-04609-9

Li, Z., Kucukkoc, I., & Zhang, Z. (2018). Branch, bound and remember algorithm for U-shaped assembly line balancing problem. *Computers & Industrial Engineering, 124*, 24-35. doi:10.1016/j.cie.2018.06.037

Li, Z., Tang, Q., & Zhang, L. (2016). Minimizing the Cycle Time in Two-Sided Assembly Lines with Assignment Restrictions: Improvements and a Simple Algorithm. *Mathematical Problems in Engineering, 2016*, 1-15. doi:10.1155/2016/4536426

Li, Z., Tang, Q., & Zhang, L. (2017). Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study. *Computers & Operations Research, 79*, 78-93. doi:10.1016/j.cor.2016.10.006

Michels, A. S., Lopes, T. C., Sikora, C. G. S., & Magatão, L. (2019). A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem. *European Journal of Operational Research, 278*(3), 796-808. doi:https://doi.org/10.1016/j.ejor.2019.05.001

Morrison, D. R., Sewell, E. C., & Jacobson, S. H. (2014). An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset. *European Journal of Operational Research, 236*(2), 403-409. doi:http://dx.doi.org/10.1016/j.ejor.2013.11.033

Mosadegh, H., Fatemi Ghomi, S. M. T., & Süer, G. A. (2020). Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and Q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research, 282*(2), 530-544. doi:https://doi.org/10.1016/j.ejor.2019.09.021

Nourie, F. J., & Venta, E. R. (1991). Finding optimal line balances with OptPack. *Operations Research Letters, 10*(3), 165-171. doi:https://doi.org/10.1016/0167-6377(91)90034-M

Pape, T. (2015). Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *European Journal of Operational Research, 240*(1), 32-42. doi:http://dx.doi.org/10.1016/j.ejor.2014.06.023

Pereira, J. (2018). The robust (minmax regret) assembly line worker assignment and balancing problem. *Computers & Operations Research, 93*, 27-40. doi:https://doi.org/10.1016/j.cor.2018.01.009

Pereira, J., & Álvarez-Miranda, E. (2018). An exact approach for the robust assembly line balancing problem. *Omega, 78*, 85-98. doi:10.1016/j.omega.2017.08.020

Scholl, A., & Klein, R. (1997). SALOME: A Bidirectional Branch-and-Bound Procedure for Assembly Line Balancing. *INFORMS Journal on Computing, 9*(4), 319-334. doi:10.1287/ijoc.9.4.319

Scholl, A., & Klein, R. (1999). Balancing assembly lines effectively – A computational comparison. *European Journal of Operational Research, 114*(1), 50-58. doi:http://dx.doi.org/10.1016/S0377-2217(98)00173-8

Sewell, E. C., & Jacobson, S. H. (2012). A Branch, Bound, and Remember Algorithm for the Simple Assembly Line Balancing Problem. *INFORMS Journal on Computing, 24*(3), 433-442. doi:10.1287/ijoc.1110.0462

Simaria, A. S., & Vilarinho, P. M. (2009). 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering, 56*(2), 489-506. doi:http://dx.doi.org/10.1016/j.cie.2007.10.007

Tang, Q., Li, Z., & Zhang, L. (2016). An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-II. *Computers & Industrial Engineering, 97*, 146-156. doi:http://dx.doi.org/10.1016/j.cie.2016.05.004

Tapkan, P., Özbakır, L., & Baykasoğlu, A. (2016). Bee algorithms for parallel two-sided assembly line balancing problem with walking times. *Applied Soft Computing, 39*, 275-291. doi:http://dx.doi.org/10.1016/j.asoc.2015.11.017

Vilà, M., & Pereira, J. (2013). An enumeration procedure for the assembly line balancing problem based on branching by non-decreasing idle time. *European Journal of Operational Research, 229*(1), 106-113. doi:http://dx.doi.org/10.1016/j.ejor.2013.03.003

Vilà, M., & Pereira, J. (2014). A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers & Operations Research, 44*, 105-114. doi:http://dx.doi.org/10.1016/j.cor.2013.10.016

Wu, E.-F., Jin, Y., Bao, J.-S., & Hu, X.-F. (2008). A branch-and-bound algorithm for two-sided assembly line balancing. *The International Journal of Advanced Manufacturing Technology, 39*(9), 1009-1015. doi:10.1007/s00170-007-1286-3

Xiaofeng, H., Erfei, W., Jinsong, B., & Ye, J. (2010). A branch-and-bound algorithm to minimize the line length of a two-sided assembly line. *European Journal of Operational Research, 206*(3), 703-707. doi:http://dx.doi.org/10.1016/j.ejor.2010.02.034

Yolmeh, A., & Salehi, N. (2017). A branch, price and remember algorithm for the U shaped assembly line balancing problem. *arXiv preprint arXiv:1708.04127*.

Zhong, Y., Deng, Z., & Xu, K. (2019). An effective artificial fish swarm optimization algorithm for two-sided assembly line balancing problems. *Computers & Industrial Engineering, 138*, 106121. doi:10.1016/j.cie.2019.106121

Özbakır, L., & Tapkan, P. (2011). Bee colony intelligence in zone constrained two-sided assembly line balancing problem. *Expert Systems with Applications, 38*(9), 11947-11957. doi:http://dx.doi.org/10.1016/j.eswa.2011.03.089

Özcan, U., Gökçen, H., & Toklu, B. (2010). Balancing parallel two-sided assembly lines. *International Journal of Production Research, 48*(16), 4767-4784. doi:10.1080/00207540903074991

Özcan, U., & Toklu, B. (2008). A tabu search algorithm for two-sided assembly line balancing. *The International Journal of Advanced Manufacturing Technology, 43*(7), 822-829. doi:10.1007/s00170-008-1753-5

Özcan, U., & Toklu, B. (2009). Balancing of mixed-model two-sided assembly lines. *Computers & Industrial Engineering, 57*(1), 217-227. doi:http://dx.doi.org/10.1016/j.cie.2008.11.012