# *A comparative study of exact methods for the simple assembly line balancing problem*

**Zixiang Li, Ibrahim Kucukkoc & Qiuhua Tang**


**Soft Computing**
A Fusion of Foundations,
Methodologies and Applications

ISSN 1432-7643

Soft Comput
DOI 10.1007/s00500-019-04609-9

Springer

Soft Computing
https://doi.org/10.1007/s00500-019-04609-9

METHODOLOGIES AND APPLICATION

# A comparative study of exact methods for the simple assembly line balancing problem

Zixiang Li[1,2] · Ibrahim Kucukkoc[3] · Qiuhua Tang[1]



**Abstract**
Exact methods have shown advanced and promising performance in solving the simple assembly line balancing problem, known as NP-hard. This research investigates the impact of various structural parameters on the performance of exact methods, including *branching methods*, *search direction*, *method to achieve upper bounds*, *utilized lower bounds*, *utilized dominance rules* and *search strategy*. In accordance with the structural parameter evaluation, *utilized dominance rules* and *search strategy* have shown the most important effect on the exact methods' performance. This research also improves and re-implements three well-known exact methods [i.e., SALOME, bounded dynamic programming (BDP) heuristic and branch, bound and remember (BBR) algorithm] using effective parameters. Computational study demonstrates that the utilization of high-performance structural parameters enhances the performance of exact methods by a significant margin. The re-implemented BBR method with proper parameters shows clear superiority over all the published exact methods and might be regarded as the state-of-the-art exact methodology.

**Keywords** Assembly line balancing · Combinatorial optimization · Heuristics · Branch and bound

## 1 Introduction

Assembly lines, composed of a set of workstations, are widely utilized in modern industry to assemble standardized products. The workstations in such lines are connected via a transportation system, e.g., a conveyer belt, and the product is assembled from the former workstations to the subsequent ones. Tasks in the latter workstations cannot be operated unless all the tasks in the former workstations are completed. To improve the efficiency of the assembly lines, assembly line balancing problems attract increasing attentions from both industry and academia (Scholl and Becker 2006; Boysen et al. 2007; Battaïa and Dolgui 2013). The basic edition of the assembly line balancing problem is the simple assembly line balancing problem (SALBP), which is known as NP-hard (Scholl and Becker 2006). As far as the optimization criterion is concerned, the SALBPs can be divided into three types as follows:

1. SALBP-I: Type I simple assembly line balancing problem to minimize the number of workstations given the cycle time;
2. SALBP-II: Type II simple assembly line balancing problem to minimize the cycle time given the number of workstations;

Communicated by V. Loia.

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s00500-019-04609-9) contains supplementary material, which is available to authorized users.

✉ Ibrahim Kucukkoc
ikucukkoc@balikesir.edu.tr

Zixiang Li
zixiangliwust@gmail.com

Qiuhua Tang
tangqiuhua@wust.edu.cn

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

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

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

Published online: 12 December 2019

3. SALBP-E: Type E simple assembly line balancing problem to maximize the line efficiency.

This study deals with the SALBP-I, where a set of tasks is allocated to workstations with the aim of minimizing the number of workstations. Each task has a positive operation time, and there are precedence relations among tasks, resulting in two main constraints needed to be fulfilled: cycle time constraint and precedence constraint. Cycle time constraint requires that the total operation time in each workstation is smaller than or equal to the cycle time given. Precedence constraint requires that the predecessors of a task must be allocated to the former workstations or start earlier when allocated to the same workstation.

### 1.1 Literature analysis

The applied methodologies to solve SALBP can be partitioned into three categories: heuristic methods, metaheuristic methods and exact methods (Battaïa and Dolgui 2013). Among the heuristic methods, Hoffman heuristic (Hoffmann 1963; Fleszar and Hindi 2003; Sternatz 2014) and beam search (Blum and Miralles 2011; Borba and Ritt 2014; Çil et al. 2017; Borba et al. 2018) produce quite effective performance in SALBP-I and its variants. As for metaheuristic methods, there are many studies: genetic algorithms (Sabuncuoglu et al. 2000), ant colony algorithms (Bautista and Pereira 2007; Blum 2008), tabu search algorithms (Scholl and Voß 1997; Lapierre et al. 2006) and many others. Detailed review on algorithms refers to recent review papers (Scholl and Becker 2006; Battaïa and Dolgui 2013). Among these methods, beam-ACO (Blum 2008) and re-implemented tabu search (Pape 2015) on the basis of Scholl and Voß (1997) are the best two performers observed in the computational study in Pape (2015).

Regarding the exact methods, there are many studies: FABLE (Johnson 1988), EUREKA (Hoffmann 1992) and OptPack (Nourie and Venta 1991) before the well-known SALOME (Scholl and Klein 1997, 1999). The comparative study in Scholl and Klein (1999) showed that SALOME outperforms the previous ones. After that, there are several exact methods outperforming SALOME, including the branch-and-bound algorithm (Liu et al. 2008), the bounded dynamic programming (BDP) heuristic (Bautista and Pereira 2009), the enumeration procedure (Vilà and Pereira 2013) and the branch, bound and remember (BBR) algorithm (Sewell and Jacobson 2012; Morrison et al. 2014). BBR is a new exact method combining branch-and-bound algorithm and dynamic programming method. It might be regarded as the current best performer for SALBP-I since it achieves the optimality for all the Scholl's 269 instances in short running times. This highly effective method has been later extended to the variants of SALBP, including U-shaped assembly lines (Yolmeh and Salehi 2017; Li et al. 2018), integrated worker assignment and line balancing problem (Vilà and Pereira 2014), robust worker assignment and line balancing problem (Pereira 2018), robotic assembly line balancing problem (Borba et al. 2018) and robust assembly line balancing problem (Pereira and Álvarez-Miranda 2018). There are also many researches on branch-and-bound methods for the variants of SALBP: assembly line balancing with station paralleling (Ege et al. 2009), two-sided assembly line balancing problem (Wu et al. 2008; Xiaofeng et al. 2010), multi-manned assembly line balancing problem (Kellegöz and Toklu 2012), integrated worker assignment and line balancing problem (Miralles et al. 2008; Borba and Ritt 2014) and U-shaped assembly line balancing problem with equipment requirements (Ogan and Azizoglu 2015).

Regarding the SALBP-II, the applied methods could also be divided into two categories: iterative solution approaches and direct solution approaches (see Scholl and Becker (2006) for a detailed review). The iterative solution approaches are based on the SALBP-I solution methods, and they solve the SALBP-II by iteratively solving a series of SALBP-I instances. Among the published methods, the iterative SALOME-2 (Klein and Scholl 1996) and iterative beam search (Blum 2010) produce the best performance. For SALBP-E, it is typically handled by iteratively solving SALBP-I or SALBP-II instances (Wei and Chao 2011; Esmaeilbeigi et al. 2015; Kucukkoc and Zhang 2015).

### 1.2 Motivations and contributions

There are multiple works on BBR methods for SALBP-I. However, the reasons (parameters and strategies) which lead to a better performance still need to be analyzed. As suggested by Morrison et al. (2014), the first motivation of this study is to carefully investigate the impact of various structural parameters, where the bidirectional search rule from SALOME is also tested as a structural parameter. Notice that this research studies the structural parameters following the procedure of BBR, since BBR is a hybrid method combining branch-and-bound algorithm and dynamic programming method. Variants of branch-and-bound algorithms and dynamic programming methods can be developed using different parameters following BBR's procedure. Another motivation comes from the fact that the re-implementation of some exact methods might cause quite different performances, leading to a possible confusion. For instance, the re-implemented SALOME by Pape (2015) produces superior performance than the original SALOME by Scholl and Klein (1997). The analysis of these structural parameters can help eliminate the possible confusion. Despite being solved by many exact methods, research on SALBP-I is needed as there are still many

unsolved challenging instances in the new dataset introduced in 2013 (Otto et al. 2013). Hence, another motivation is to develop a more effective method that is capable of producing the state-of-the-art results and update part of the current upper bounds. Additionally, the analysis of these structural parameters can also provide some guidelines when extending the BBR to other assembly line balancing problems.

This research presents a comparative study of exact methods for the SALBP-I: The cycle time is given and the goal is to minimize the number of stations. Hence, this study provides the first attempt to investigate the impact of various structural parameters on BBR algorithms' performance as suggested by Morrison et al. (2014). The branching method, the search direction, method to achieve upper bounds, utilized lower bounds, utilized dominance rules and search strategy are also tested as the parameters to reveal the reasons leading to superiority of BBR method. The well-known SALOME or BDP can be achieved using one of the combinations of these structural parameters. This research also re-implements three well-known exact methods (SALOME, BDP and BBR), and computational study demonstrates that the utilization of high-performing structural parameters significantly enhances the performance of SALOME, BDP and BBR by a significant margin. The re-implemented BBR with proper parameters shows clear superiority in comparison with all published exact methods and might be regarded as the state-of-the-art exact methodology. Finally, this research provides several possible guidelines to extend this method to the variants of SALBP.

The remainder of this research is organized as follows. Section 2 provides a detailed description of the BBR algorithm, and Sect. 3 tests the six structural parameters along with comparative studies. Section 4 compares the original exact methods and improved methods using better parameters and carries out a comprehensive study to identify the state-of-the-art methodology. Finally, Sect. 5 concludes this paper and provides several research venues.

## 2 Branch, bound and remember algorithm

BBR is a hybrid method combining the branch-and-bound algorithm and the dynamic programming method (Sewell and Jacobson 2012; Morrison et al. 2014; Li et al. 2018). The main feature of this method is that it stores all the subproblems searched and checks whether one subproblem is dominated by a subproblem in memory before branching. This method produces quite promising results in SALBP-I and other variants, and it quickly achieves the optimality for all the Scholl's 269 instances. Following subsections present a detailed description of the BBR method. Recall that BBR is selected to test the parameters as it hybridizes the branch-and-bound algorithm and dynamic programming method. The branch-and-bound algorithms and dynamic programming method might be achieved as the variants of BBR methods with different parameters.

### 2.1 Main procedure of BBR method

The general procedure of BBR is outlined as follows. LB1, LB2 and LB3 are the three well-known lower bounds (LB) in Scholl and Klein (1997), and BPLB is the bin packing lower bound achieved by branch-and-bound solver to solve the bin packing problem optimally (Sewell and Jacobson 2012). This algorithm consists of three main phases, and starts with obtaining an upper bound with high-performing heuristic methods, e.g., modified Hoffman heuristic. Subsequently, the cyclic best-first search in Phase II is conducted, using lower bounds and dominance rules, to attempt to find new better solutions and update the upper bound (UB). If Phase II fails to prove the optimality of the achieved solution, breadth-first search in Phase III is executed to attempt to verify the optimality using lower bounds and dominance rules.

Algorithm 1. The procedure of BBR method

| | | |
|---|---|---|
| Phase I | **1** | Determine the search direction |
| | | % Decide whether the problem is solved in the forward direction or in the backward direction |
| | **2** | Achieve UB with a high-performance heuristic |
| | | % Published BBR utilizes modified Hoffman heuristic |
| | **3** | Calculate the lower bound at the root or $LB_{root}$ |
| | | % $LB_{root}$ is the maximum value of LB1, LB2, LB3 and BPLB at the root in the published BBR, and it is also applicable to employ other lower bounds. |
| **Phase II** | **4** | **If** $UB > LB_{root}$ |
| | | Execute cyclic best-first search and update UB when necessary |
| | | % The published BBR employs cyclic best-first search, and it is also applicable to employ other search strategies, e.g., depth-first search strategy. |
| | | **Endif** |
| **Phase III** | **5** | **If** $UB > LB_{root}$ **and** termination criterion is not met |
| | | Execute breadth-first search and update UB when necessary |
| | | % The breadth-first search attempts to verify the optimality of the achieved solution, and only breadth-first search is allowed. |
| | | **Endif** |

### 2.2 Branch-and-bound search procedure

This section illustrates the procedure to execute cyclic best-first search or breadth-first search. In this research, a partial solution (subproblem) is denoted as $\wp = (A, U, S_1, S_2, \ldots, S_m)$, where $S_j$ is the set of tasks assigned to workstation $j$ ($j = 1, \ldots, m$), $A$ refers to the set of assigned tasks on the former $m$ workstations ($A = \bigcup_{j=1}^{m} S_j$) and $U$ is the set of unassigned tasks. In the procedure, a new non-dominated partial solution $X(A, U, S_1, S_2, \ldots, S_m)$ is first selected based on the utilized search strategy. Afterward, a number of offspring or new partial solutions $Y(A', U', S_1, S_2, \ldots, S_m, S_{m+1})$ in deeper depth are generated. Each new partial solution $X$ is stored only when it cannot be dominated by lower bounds and dominance rules.

Algorithm 2. Branch-and-bound search procedure

1 **While** there is a non-explored partial solution in memory and the termination criterion is not met
2 Select a new non-dominated partial solution $X(A, U, S_1, S_2, \ldots, S_m)$ according to the search strategy
3 **While** search tree is not empty or the number of generated station-loads is not larger than a given number (10,000 in Sewell and Jacobson (2012))
4 Generate a new partial solution $Y(A', U', S_1, S_2, \ldots, S_m, S_{m+1})$ in deeper depth based on subproblem $X$
5 If $Y$ is a complete solution, update UB when necessary
6 Delete $Y$ if $\max\{LB1, LB2, LB3, BPLB\} \geq UB$
% Delete the partial solution which cannot achieve better upper bound.
7 Delete $Y$ if it is dominated by one of the dominance rules (maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule)
% In the memory-based dominance rule, $Y$ is dominated when there is a subproblem $Z$ in memory with the same assigned task set and the same or smaller lower bound.
8 Store the subproblem $Y$
9 **Endwhile**
10 **Endwhile**
11 Output UB

In the BBR methods, different upper bounds, dominance rules and search strategies might result in quite different performances. In the following section, these structural parameters are tested, including branching method, search direction, the method to achieve upper bounds, utilized lower bounds, utilized dominance rules and search strategy in Phase II.

## 3 Structural parameter evaluation

This section tests the structural parameters on the BBR method's performance, where all the important parameters are included. Three sets of well-known and hard instances are solved: Scholl's 269 instances (Scholl and Klein 1997), Otto-100 with 100 tasks (Otto, Otto et al. 2013) and Otto-1000 with 1000 tasks (Otto, Otto et al. 2013), where Otto-100 has 525 instances and Otto-1000 has 525 instances, leading to a total number of 1319 instances. The information about these instances is available and provided in the cited papers. The BBR method terminates when the optimal solution is achieved and verified. It also terminates if the elapsed computation time reaches 900 s.

To evaluate the performance of BBR method on different instances, this research utilizes several evaluation criteria as follows. Here, the relative percentage deviation (RPD) for one instance is calculated with $\text{RPD} = 100 \cdot (\text{UB}_{\text{some}} - \text{LB})/\text{LB}$, where $\text{UB}_{\text{some}}$ is the achieved UB by one configuration and LB is the maximum value of lower bounds by all the tested methods and that reported in Pape (2015).

| | |
|---|---|
| #OPT | The number of instances for which an optimal solution is found |
| RPD-Avg | The average of the RPD values for all the tested instances |
| RPD-Max | The maximum of the RPD values for all the tested instances |
| RPD-Var | The variance of the RPD values for all the tested instances |
| CPU-Avg | The average running time for the instances solved |

During the parameter calibration, there are several settings for each parameter, leading to too many combinations of the parameters. Due to the hardness of the problem, it takes a lot of computation time to test all the combinations of these parameters. Hence, this study tests the parameters one by one from Sects. 3.1–3.6 with other parameters fixed to the values in the original BBR method in Morrison et al. (2014) with minor modifications. The based parameters remaining fixed are listed as follows.

| | |
|---|---|
| Branching methods | The station-oriented branching method with a maximum number of 10,000 station-loads by Morrison et al. (2014) |
| Search directions | Direction selection method by Morrison et al. (2014) |
| Upper bounds | Modified Hoffmann heuristic with 10,000 station-loads by Morrison et al. (2014) |

| | |
|---|---|
| Lower bounds | LB1, LB2, LB3, LB6 and BPLB (see Sect. 3.4 for a detailed description) |
| Dominance rules | Maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule by Morrison et al. (2014). |
| Search strategies | Modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\|U\|$ (see Sect. 3.6 for a detailed description) |

Subsequently, Sect. 3.7 discusses the interactions between parameters to achieve the best combinations of the parameters. The details on tested parameters and comparative study are presented in the following subsections.

### 3.1 Branching methods

There are two widely utilized branching methods: task-oriented branching (Johnson 1988; Nourie and Venta 1991) and station-oriented branching (Hoffmann 1992; Scholl and Klein 1997, 1999). Regarding task-oriented branching method, partial solutions are generated by allocating a task to the current station if all constraints are satisfied or this task is allocated to the latter station (a new workstation is open). For station-oriented branching, partial solutions are created by allocating a set of tasks or a complete station-load to the next station. Suppose a partial solution $\wp = (A, U, S_1, S_2, \ldots, S_m)$, a new partial solution is $\wp' = (A', U', S_1, S_2, \ldots, S_m, S_{m+1})$. As station-oriented branching method has shown clear superiority in the literature (Scholl and Klein 1997, 1999), this article studies only variants of station-oriented branching method: different maximum numbers of generated station-loads at each depth and different renumbering methods.

The tested branching methods are summarized in Table 1, where the suffix denotes the maximum number of generated station-loads. In this table, BM1 is the original branching method where the sequence in enumeration remains unchanged. BM2 renumbers tasks using task time where the tasks with larger operation times are enumerated first. BM3 renumbers tasks using positional weight where the tasks with larger positional weights are enumerated first. BM4 renumbers tasks using task time and positional weight where the tasks with larger task times and larger positional weights are enumerated first. These three renumbering methods (i.e., BM2, BM3 and BM4) are taken from a recent study by Li et al. (2018), and they are tested as they are simple and straightforward. There are also other renumbering methods [e.g., the renumbering method by Scholl and Klein (1999)] not tested here.

Table 2 illustrates the detailed results by BBR methods with different branching methods. It is observed that the number of 10,000 station-loads produces the best performance and it consumes less average computation time than that of 50,000 station-loads. BM2-10,000, BM3-10,000 and BM4-10,000 produce better performance than BM1-10,000 regarding the RPD-Avg value, indicating that renumbering the tasks somewhat enhances the performance of BBR method. It is also observed that BM3-10,000 is the best performer in terms of the RPD-Avg indicator, whereas BM4-10,000 is the best performer in terms of the #OPT indicator. As RPD-Avg provides the information on solving all the instances and #OPT provides no information for the instances that are not solved optimally, this study selects the factor with the minimum RPD-Avg and RPD-Var as the best performer. Hence, BM3-10,000 is selected as the best value and utilized when re-implementing BBR in Sects. 3.7 and 4.

### 3.2 Search directions

SALBP-I can be solved either in the forward direction (tasks are allocated to workstation 1, workstation 2, etc.), or in the backward direction (tasks are allocated to the last workstation, the second last workstation, etc.). As some instances might be easier to solve in the reverse direction, Hoffmann (1992) spends half of the time searching in the forward direction and then switches to the backward direction. However, Scholl and Klein (1997) developed a bidirectional branching rule to determine whether to perform a forward step or backward step to generate the new partial solution $Y(A', U', S_1, S_2, \ldots, S_m, S_{m+1})$ based on a selected subproblem $X(A, U, S_1, S_2, \ldots, S_m)$.

Different from bidirectional branching rule, the published BBR method utilizes one simple formula to determine which direction to search (Sewell and Jacobson 2012). Recall that, the direction selection in BBR method determines whether one problem is solved in forward or backward direction at the root. In other words, this direction selection procedure is executed only once for the whole problem when utilizing the direction selection method; a mono-directional branching rule is carried out for each selected partial solution when utilizing the bidirectional branching rule. This section tests four search directions summarized in Table 3, where the forward (backward) direction refers to solving the instances in forward (backward) direction.

Table 4 presents the computational results, and it is clear that SD4 produces the best performance with the minimum RPD-Avg value and minimum average computation time. SD1 is the second-best performer, SD3 is the third-best performer, and SD2 is the worst performer. For SD3, the search space is much larger than SD1, SD2 and SD4 (only forward or backward direction is utilized) and hence SD3 is outperformed by SD4 utilizing direction

**Table 1** Description of branching methods

| Purpose | Abbreviation | Description |
|---|---|---|
| Calibrate the number of maximum station-loads | BM1-1000 | Do not renumber the tasks and the number of maximum station-loads is 1000 |
| | BM1-5000 | Do not renumber the tasks and the number of maximum station-loads is 5000 |
| | BM1-10,000 | Do not renumber the tasks and the number of maximum station-loads is 10,000 (this is utilized in the original BBR method) |
| | BM1-50,000 | Do not renumber the tasks and the number of maximum station-loads is 50,000 |
| Calibrate the method of renumbering the tasks | BM1-10,000 | Do not renumber the tasks and the number of maximum station-loads is 10,000 (this is utilized in the original BBR method) |
| | BM2-10,000 | Renumber the tasks using task time |
| | BM3-10,000 | Renumber the tasks using positional weight |
| | BM4-10,000 | Renumber the tasks using task time and positional weight |

**Table 2** The results by BBR methods utilizing different branching methods

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| BM1-1000 | 1135 | 0.3571 | 7.6923 | 1.0049 | 137.22 |
| BM1-5000 | 1136 | 0.3510 | 6.1538 | 0.9848 | 136.81 |
| BM1-10,000 | 1136 | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| BM1-50,000 | 1136 | 0.3508 | 6.1538 | 0.9845 | 137.47 |
| BM1-10,000 | 1136 | 0.3504 | 6.1538 | 0.9835 | 136.86 |
| BM2-10,000 | 1136 | 0.3481 | 5.9048 | 0.9769 | 137.19 |
| BM3-10,000 | 1136 | **0.3432** | 6.1538 | 0.9619 | 136.58 |
| BM4-10,000 | 1137 | 0.3477 | 6.1538 | 0.9754 | 136.78 |

Best value(s) in bold

**Table 3** Description of utilized search directions

| Purpose | Abbreviation | Description |
|---|---|---|
| Calibrate the search direction | SD1 | Forward direction |
| | SD2 | Backward direction |
| | SD3 | Bidirectional branching method by Scholl and Klein (1997) |
| | SD4 | Direction selection method by Sewell and Jacobson (2012) (this is utilized in the original BBR method) |

**Table 4** The results by BBR methods utilizing different search directions

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| SD1 | 1131 | 0.3602 | 6.1538 | 0.9931 | 155.08 |
| SD2 | 1116 | 0.6519 | 7.9681 | 1.7106 | 185.76 |
| SD3 | 1132 | 0.3862 | 6.0952 | 1.0661 | 155.75 |
| SD4 | **1136** | **0.3504** | 6.1538 | 0.9835 | **136.86** |

Best value(s) in bold

selection method by Sewell and Jacobson (2012). Notice that the search directions SD1 and SD2 are theoretically equivalent, and any difference in performance between SD1 and SD2 is more attributable to the instances than the search directions themselves. This comparative study shows that SD4 is the best performer, indicating that it is worthwhile to determine the proper search direction before

solving an instance. Hence, SD4 is selected and utilized when re-implementing BBR in Sects. 3.7 and 4.

### 3.3 Upper bounds

Effective upper bounds help to decrease the number of subproblems, and Hoffman heuristic (the original Hoffman heuristic is referred to as OHH) is a well-known and effective methodology. OHH builds a solution from station to station; it generates a number of station-loads for each station and selects the one with minimum idle time. There are several other improved adaptations of OHH, and among them, the modified Hoffmann heuristic (MHH) by Sewell and Jacobson (2012) produces quite effective performance with the cost of increased running time.

Suppose that a selected partial solution is $\wp = (A, U, S_1, S_2, \ldots, S_m)$, a new partial solution is $(A', U', S_1, S_2, \ldots, S_m, S_m + 1)$ and MHH selects a station-load with the maximum value of $\sum_{i \in S_{m+1}} (t_i + \alpha \cdot w_i + \beta \cdot |F_i| - \gamma)$ for station $m + 1$, where $S_{m+1}$ is the set of tasks allocated to station $m + 1$, $F_i(F_i^*)$ is the set of immediate (all) successors of task $i$, $w_i$ is the positional weight of task $i$ ($w_i = t_i + \sum_{j \in F_i^*} t_j$), $|F_i|$ is the number of tasks in set $F_i$, and $\alpha$, $\beta$ and $\gamma$ are three parameters. The values of $\alpha$, $\beta$ and $\gamma$ are set as follows:$\alpha, \beta \in \{0, 0.005, 0.010, 0.015, 0.020\}$ and $\gamma \in \{0, 0.01, 0.02, 0.03\}$. All the possible combinations of these factors are tested, and the best one among the achieved station number is considered to be the UB by MHH. This section also tests one simple constructive heuristic as an example: the well-known ranked positional weight method (Scholl and Becker 2006). This kind of heuristic might lead to solutions of less quality, but it is very quick to obtain complete solutions.

Table 5 lists the tested methods to achieve upper bounds, where the suffix number denotes the maximum number of generated station-loads, NUB means that no method is employed to achieve upper bounds and RPW denotes the ranked positional weight method applied to achieve upper bounds.

Table 6 illustrates the quality of the solutions by these methods, and Table 7 presents the results by BBR methods using upper bounds by these methods, where the presented time is the whole time by BBR methods and the consumed time to achieve UB is also included. In Table 6, it is observed that, among the methods to calibrate the number of station-loads, the MHH1-50,000 is the best performer and the utilization of a larger number of station-loads does not necessarily improve the achieved RPD-Avg value. Regarding calibrating the methods of obtaining upper bound, MHH3-10,000 produces the peak performance. The RPW and OHH-1000 achieve the worst RPD-Avg values, but they require the minimum running times.

In Table 7, it is observed that the BBR methods with different upper bounds show similar performance with the running time increasing. For calibrating the number of station-loads, MHH1-10,000 produces the best performance. MHH1-100,000 produces the second-worst performance as much time is consumed to achieve the upper bounds, which suggests that the utilization of much time to obtain the upper bounds is not a good option. For calibrating the method of obtaining upper bound, it is observed that utilization of MHH methods outperforms NUB and RPW, and the MHH1-10,000 again obtains the best performance. Among the three renumbering methods, MHH3-10,000 obtains the best performance. To select the best combination of the number of station-loads and the renumbering methods, the interaction between the number of station-loads and the renumbering methods is further studied in Sect. 3.7.

### 3.4 Lower bounds

Lower bounds are utilized to prune a partial solution that cannot achieve a smaller number of stations. The available lower bounds include the well-known seven lower bounds (LB1, LB2, LB3, LB4, LB5, LB6 and LB7) (Scholl and Klein 1997), the LB8 (Fleszar and Hindi 2003), the BPLB by solving bin packing problem optimally and several other improved variants (Pape 2015; Pereira 2015). Detailed descriptions of these lower bounds refer to the cited papers. This section mainly tests the combinations of the LB1, LB2, LB3, LB6, BPLB and four other lower bounds as illustrated in Table 8. LB1, LB2, LB3, LB6, LB4, LB5, LB7 and LB8 are applied in sequence before utilizing dominance rules, and BPLB is utilized after utilizing all the dominance rules as BPLB costs much more time than other lower bounding methods. Due to its relative ineffectiveness and tremendous time in solving the very large-size instances, the BPLB is not utilized when solving the Otto-1000 with 1000 tasks as suggested by Morrison et al. (2014).

Table 9 presents the detailed results obtained with the utilization of different branching methods. It is observed that the fast LBM4 produces the best result regarding #OPT and RPD-Avg, and it outperforms LBM1 and LBM3 clearly in terms of #OPT and RPD-Avg. This finding suggests that the utilization of LB6 and BPLB with the cost of larger time is worthwhile. Nevertheless, the utilization of LB4, LB5, LB7 or LB8 leads to poor performance as much time is consumed to calculate the lower bounds. Hence, LBM4 is selected and utilized when re-implementing BBR in Sects. 3.7 and 4.

**Table 5** Description of the methods to achieve upper bounds

| Purpose | Abbreviation | Description |
|---|---|---|
| Calibrate the number of station-loads | MHH1-1000 | MHH with 1000 station-loads |
| | MHH1-5000 | MHH with 5000 station-loads |
| | MHH1-10,000 | MHH with 10,000 station-loads (this is utilized in the original BBR method) |
| | MHH1-50,000 | MHH with 50,000 station-loads |
| | MHH1-100,000 | MHH with 100,000 station-loads |
| Calibrate the method of obtaining upper bound | NUB | No method is applied to achieve UB |
| | RPW | Ranked positional weight method is applied to achieve UB |
| | OHH-10,000 | Original MHH with 10,000 station-loads |
| | MHH1-10,000 | MHH with 10,000 station-loads (this is utilized in the original BBR method) |
| | MHH2-10,000 | Renumber the tasks using task time |
| | MHH3-10,000 | Renumber the tasks using positional weight |
| | MHH4-10,000 | Renumber the tasks using task time and positional weight |

**Table 6** The quality of the achieved upper bounds

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| MHH1-1000 | 891 | 1.6403 | 20.0000 | 2.8001 | 0.42 |
| MHH1-5000 | 893 | 1.6423 | 20.0000 | 2.8100 | 1.31 |
| MHH1-10,000 | 893 | 1.6422 | 20.0000 | 2.8083 | 1.79 |
| MHH1-50,000 | **894** | **1.6386** | 20.0000 | 2.8074 | 2.35 |
| MHH1-100,000 | 893 | 1.6395 | 20.0000 | 2.8068 | 2.72 |
| RPW | 325 | 4.1932 | 50.0000 | 4.6077 | 0.00 |
| OHH-10,000 | 764 | 2.2901 | 20.0000 | 3.3886 | 0.04 |
| MHH1-10,000 | 893 | 1.6422 | 20.0000 | 2.8083 | 1.79 |
| MHH2-10,000 | 891 | 1.6480 | 20.0000 | 2.8143 | 1.35 |
| MHH3-10,000 | **899** | **1.6143** | 20.0000 | 2.7938 | 1.51 |
| MHH4-10,000 | 893 | 1.6407 | 20.0000 | 2.8122 | 2.06 |

Best value(s) in bold

**Table 7** The results by BBR methods utilizing different upper bounds

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| MHH1-1000 | 1135 | 0.3528 | 6.1538 | 0.9882 | 136.93 |
| MHH1-5000 | 1136 | 0.3507 | 6.1538 | 0.9840 | 137.58 |
| MHH1-10,000 | 1136 | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| MHH1-50,000 | 1136 | 0.3505 | 6.1538 | 0.9839 | 137.54 |
| MHH1-100,000 | 1135 | 0.3519 | 6.1538 | 0.9846 | 137.88 |
| NUB | 1135 | 0.3530 | 5.9615 | 0.9861 | 150.82 |
| RPW | 1133 | 0.3522 | 5.9813 | 0.9813 | 149.72 |
| OHH-10,000 | 1136 | 0.3519 | 5.7540 | 0.9900 | 140.55 |
| MHH1-10,000 | 1136 | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| MHH2-10,000 | 1134 | 0.3551 | 6.1538 | 0.9901 | 136.90 |
| MHH3-10,000 | 1136 | 0.3508 | 6.1538 | 0.9844 | 137.71 |
| MHH4-10,000 | 1136 | 0.3510 | 6.1538 | 0.9842 | 137.72 |

Best value(s) in bold

**Table 8** Description of utilized lower bounds

| Purpose | Abbreviation | Description |
|---|---|---|
| Calibrate the LB1, LB2, LB3, LB6 and BPLB | LBM1 | Only LB1, LB2 and LB3 are utilized |
| | LBM2 | LB1, LB2, LB3 and LB6 are utilized |
| | LBM3 | LB1, LB2, LB3 and BPLB are utilized (this is utilized in the original BBR method) |
| | LBM4 | LB1, LB2, LB3, LB6 and BPLB are utilized |
| Calibrate other lower bounds | LBM4 | LB1, LB2, LB3, LB6 and BPLB are utilized |
| | LBM5 | LB1, LB2, LB3, LB6, BPLB and LB4 are utilized |
| | LBM6 | LB1, LB2, LB3, LB6, BPLB and LB5 are utilized |
| | LBM7 | LB1, LB2, LB3, LB6, BPLB and LB7 are utilized |
| | LBM8 | LB1, LB2, LB3, LB6, BPLB and LB8 are utilized |

**Table 9** The results by BBR methods utilizing different lower bounds

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| LBM1 | 1120 | 0.3820 | 5.1331 | 0.9915 | 162.16 |
| LBM2 | 1135 | 0.3519 | 6.1538 | 0.9845 | 137.16 |
| LBM3 | 1131 | 0.3637 | 5.1331 | 0.9762 | 144.30 |
| LBM4 | **1136** | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| LBM4 | **1136** | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| LBM5 | 1112 | 0.6474 | 9.9404 | 1.7209 | 164.60 |
| LBM6 | 1112 | 0.6215 | 9.9404 | 1.6608 | 162.79 |
| LBM7 | 1136 | 0.3529 | 5.6391 | 0.9892 | 137.53 |
| LBM8 | 1110 | 0.9326 | 10.4762 | 2.3856 | 167.00 |

Best value(s) in bold

### 3.5 Dominance rules

Dominance rules are employed to prune the dominated partial solutions to reduce the search tree. Table 10 illustrates the tested dominance rules in Sewell and Jacobson (2012) to test the effect of these dominance rules on the performance of BBR. Detailed descriptions of these dominance rules refer to Scholl and Klein (1997) and Sewell and Jacobson (2012).

Table 11 illustrates the results by BBR methods utilizing different dominance rules. It is clear that the BBR methods with memory-based dominance rule (DRM3 and DRM4) outperform the BBR methods without memory-based dominance rule (DRM1 and DRM2), indicating that memory-based dominance rule enhances the performance of BBR method by a significant margin. It is also observed that the utilization of more dominance rules enhances the performance of BBR method. Hence, DRM4 is selected and utilized when re-implementing BBR in Sects. 3.7 and 4.

**Table 10** Description of utilized dominance rules

| Purpose | Abbreviation | Description |
|---|---|---|
| Calibrate the dominance rules | DRM1 | Only maximal load rule and extended Jackson rule are utilized |
| | DRM2 | Maximal load rule, extended Jackson rule and no-successors rule are utilized |
| | DRM3 | Maximal load rule and memory-based dominance rule are utilized |
| | DRM4 | Maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule are utilized (this is utilized in the original BBR method) |

**Table 11** The results by BBR methods utilizing different dominance rules

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| DRM1 | 1075 | 0.6980 | 10.0760 | 1.7284 | 206.47 |
| DRM2 | 1075 | 0.6601 | 9.9609 | 1.6614 | 206.35 |
| DRM3 | 1135 | 0.3900 | 6.7308 | 1.0710 | 141.90 |
| DRM4 | **1136** | **0.3504** | 6.1538 | 0.9835 | 136.86 |

Best value(s) in bold

### 3.6 Search strategies

Search strategies determine the order in which subproblems are explored, and this order has an important impact on the computation time of BBR method. There are several well-known search strategies: depth-first search (DFS) (Scholl and Klein 1997), best-first search (BFS) (Yolmeh and Salehi 2017), breadth-first search (BrFS) and cyclic best-first search (CBFS) (Sewell and Jacobson 2012).

DFS begins with selecting one subproblem at depth 1, then selects one subproblem at depth 2 and subsequently selects one subproblem successively at deeper depth until the deepest depth is reached. DFS returns to the top of the tree after exhausting all subproblems within that sub-tree. Namely, DFS first exhausts the search at the deepest depths before coming back to the previous ones. BFS always searches the best node in terms of an evaluation function or bound. BFS is heavily based on the selection criterion, and improper selection criterion might lead to poor performance or large running time to achieve a complete solution. BrFS is a heavy and slow search strategy; it generates all subproblems at depth 1, depth 2 and finally at the deepest depth until the complete optimal solution is achieved. One clear drawback is that this method is very slow and the search procedure might terminate due to the time limit before a complete solution is achieved in solving large-size instances. However, BrFS might be utilized to prove the optimality of some hard instances when a tight UB is achieved by other search strategies (Sewell and Jacobson 2012).

CBFS is a hybrid of DFS and BFS, and it begins with selecting one subproblem at depth 1, then selects one subproblem at depth 2 and subsequently selects one subproblem successively at deeper depth until the deepest depth is reached. When the deepest depth is reached, again one subproblem at depth 1 is selected and this procedure is repeated until the optimal solution is achieved or a termination criterion is satisfied. Namely, CBFS searches the best node at a given depth and cyclically changes the current depth. Li et al. (2018) developed a modified cyclic best-first search, referred to as MCBFS, using a new selection criterion, in which subproblem at depth $l$ is not chosen if there is plenty of promising subproblems (10,000 in this paper) at depth $l+1$. This modified edition avoids too many subproblems at one depth and increases the speed to achieve high-quality and complete solutions.

Apart from the aforementioned search strategies, there is another search strategy applied in dynamic programming method (Bautista and Pereira 2009). In this research, this search strategy is denoted as beam search strategy (BSS) as this strategy is quite similar to beam search heuristic. In BSS, there are two parameters: the number of subproblems selected at each depth and the maximum number of station-loads for one selected subproblem. In other words, the BSS differentiates from DFS and CBFS in that BSS selects several best subproblems at each depth but DFS and CBFS only select one best subproblem at each depth. All the DFS, BFS, CBFS, MCBFS and BSS are tested in this section as presented in Table 12. BrFS is not tested here as BrFS cannot obtain a complete solution in solving very large-size instances within an acceptable amount of time.

For DFS, BFS, CBFS and BSS, the selection criterion is a necessary and important part and there are several different published selection criteria. Suppose that $b(\wp)$ is utilized to evaluate one subproblem $\wp$ ($\wp = (A, U, S_1, S_2, \ldots, S_m)$). The one, among the subproblems, with the minimum value of $b(\wp)$ is regarded as the best subproblem and selected. Table 12 presents the utilized strategies along with several different selection criteria, where CT is the given cycle time, $t_i$ is the operation time of task $i$, $I$ is the total idle time in the former $m$ stations, $\lambda$ is an input number (set to 0.02), $\mathrm{LB}(U)$ is the lower bound of unassigned task set $U$ and $|U|$ is the number of unassigned tasks. For MCBFS methods, the subproblem at depth $l$ is not selected if there are 10,000 subproblems at depth $l+1$. Regarding the BSS-$a$-$b$, the first number $a$ is the maximum number of states, maintained from one stage to the following stage, and the second number $b$ is the maximum number of states developed from a state. The subproblem at depth $l$ is not chosen if there are $10,000$ subproblems at depth $l+1$.

Table 13 presents the results utilizing different search strategies. Regarding the station-load selection criterion, it is clear that this is quite a big difference among different station-load selection criteria. Among the six MCBFS methods, MCBFS-S5 and MCBFS-S6 obtain the best performance, whereas MCBFS1 and MCBFS3 show clearly poor performance. Notice that the selection criterion S5 in

**Table 12** Description of the utilized search strategies

| Purpose | Abbreviation | Description |
|---|---|---|
| Calibrate the station-load selection criterion | MCBFS-S1 | Modified cyclic best-first search with $b(\wp) = \mathrm{CT} - \sum_{i \in S_m} t_i$ |
| | MCBFS-S2 | Modified cyclic best-first search with $b(\wp) = I$ |
| | MCBFS-S3 | Modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U)$ |
| | MCBFS-S4 | Modified cyclic best-first search with $b(\wp) = I/m - \lambda\lvert U\rvert$ (this load selection criterion is utilized in the original BBR method) |
| | MCBFS-S5 | Modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U) - 10 \cdot m + I/m - \lambda\lvert U\rvert$ |
| | MCBFS-S6 | Modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| Calibrate the two parameters of beam search strategy | BSS-S6-5-1000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-10-1000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-5-5000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-10-5000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-5-10,000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-10-10,000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| Calibrate the search strategy | DFS-S6 | Depth-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BFS-S6 | Best search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | CBFS-S6 | Cyclic best-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ (this search strategy is utilized in the original BBR method) |
| | MCBFS-S6 | Modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-10-1000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |
| | BSS-S6-5-5000 | Beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda\lvert U\rvert$ |

MCBFS-S5 and the selection criterion S6 in MCBFS-S6 are equivalent when utilizing DFS, CBFS and MCBFS. However, these two different selection criteria might produce different performances when utilizing BFS strategy. This finding suggests that the utilization of the effective station-load selection criterion has great impact on the performance of BBR algorithm. As for calibrating the two parameters of beam search strategy, the BSS-S6-10-5000 is the best performer in terms of the solution quality or CPU-Avg. Specifically, BSS-S6-10-5000 outperforms BSS-S6-5-1000, BSS-S6-10-1000, BSS-S6-5-5000 and BSS-S6-5-10,000 in terms of solution quality and BSS-S6-10-10,000 in terms of CPU-Avg.

Regarding the search strategies, it is observed that BSS-S6-10-5000 is the best performer in terms of RPD-Avg. MCBFS-S6 is the second-best performer, and it shows slightly better performance than CBFS-S6 in terms of RPD-Avg and CPU-Avg. BFS6 and DFS-S6 are the worst two performers in terms of #OPT and RPD-Avg. In this study, both MCBFS6 and BSS-S6-10-5000 are selected and tested when re-implementing BBR in Sects. 3.7 and 4.

### 3.7 The interaction between parameters

This section aims at testing the possible interactions between the components and tries to find an effective combination of the parameter values that achieves a better performance. Nevertheless, there are many combinations of the parameters and it is impractical to test all combinations. Hence, this section mainly tests two sets of combinations as follows. For each parameter, only two or three effective factors are selected and tested.

The first combinations are conducted to determine the best method to achieve the upper bound, and Table 14 presents the detailed computational results utilizing the six combinations of the number of the station-loads and the utilization of the renumbering methods. It is observed that the MHH1-10,000 is the best performer and MHH3-1000 is the second-best performer. Nevertheless, the best method might be different under different termination criteria. The initial experiments show that MHH3-1000 produces the better performance than MHH1-10,000 when the computation time is short (180 s and 500 s) as less time is utilized for the branch-and-bound procedure when utilizing MHH1-10,000. Hence, MHH3-1000 is selected when re-implementing BBR in the latter part of Sects. 3.7 and 4 as the algorithms are tested under three termination criteria

**Table 13** The results by BBR methods utilizing different search strategies

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| MCBF-S1 | 1118 | 0.8594 | 10.7692 | 2.2382 | 152.13 |
| MCBF-S2 | 1136 | 0.5727 | 8.5938 | 1.5112 | 137.79 |
| MCBF-S3 | 1105 | 0.5981 | 8.0078 | 1.5822 | 158.22 |
| MCBF-S4 | 1136 | 0.5564 | 8.2031 | 1.4809 | 137.92 |
| MCBF-S5 | 1136 | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| MCBF-S6 | 1136 | **0.3504** | 6.1538 | 0.9835 | 136.86 |
| BSS-S6-5-1000 | 1134 | 0.3437 | 7.6923 | 0.9614 | 138.51 |
| BSS-S6-10-1000 | 1135 | 0.3310 | 7.6923 | 0.9292 | 138.27 |
| BSS-S6-5-5000 | 1135 | 0.3371 | 5.4206 | 0.9387 | 137.70 |
| BSS-S6-10-5000 | 1136 | **0.3255** | 5.2336 | 0.9073 | 138.41 |
| BSS-S6-5-10,000 | 1135 | 0.3376 | 5.4206 | 0.9400 | 138.16 |
| BSS-S6-10-10,000 | 1136 | 0.3255 | 5.2336 | 0.9073 | 138.67 |
| DFS-S6 | 1114 | 0.6575 | 9.9609 | 1.7728 | 154.42 |
| BFS-S6 | 1094 | 0.7304 | 9.9404 | 1.8536 | 171.97 |
| CBFS-S6 | 1136 | 0.3507 | 6.1538 | 0.9840 | 136.87 |
| MCBFS-S6 | 1136 | 0.3504 | 6.1538 | 0.9835 | 136.86 |
| BSS-S6-10-5000 | 1136 | **0.3255** | 5.2336 | 0.9073 | 138.41 |

Best value(s) in bold

**Table 14** The six combinations and results by BBR methods utilizing these combinations

| Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|
| MHH1-1000 | 1135 | 0.3476 | 6.1538 | 0.9697 | 136.66 |
| MHH3-1000 | 1136 | 0.3441 | 6.1538 | 0.9636 | 136.42 |
| MHH1-5000 | 1136 | 0.3470 | 6.1538 | 0.9706 | 137.53 |
| MHH3-5000 | 1136 | 0.3470 | 6.1538 | 0.9706 | 137.38 |
| MHH1-10,000 | 1136 | **0.3432** | 6.1538 | 0.9619 | 136.58 |
| MHH3-10,000 | 1136 | 0.3461 | 5.9813 | 0.9740 | 136.54 |

Best value(s) in bold

(180 s, 500 s and 900 s). Notice that the gap between the methods to obtain upper bounds is very small (within 0.01%) and hence it is acceptable to select any of these tested MHH methods.

The second combinations are tested to study the interactions between three important parameters: lower bounds, station-load selection criteria and the search strategies. Only the two effective factors are tested here, leading to a total number of eight combinations. Table 15 provides the combinations of these three parameters and computational results utilizing the parameter combinations. From this table, it is observed that the combination of LBM4 and BSS-S6-10-5000 achieves the best performance in terms of RPD-Avg. The combination of LBM4 and MCBFS-S6 shows the best performance in terms of #OPT and a similar performance in terms of RPD-Avg.

From the above parameter evaluation, it might be concluded that the superiority of the BBR method is mainly attributed to two aspects: proper dominance rules (mainly the utilization of the memory-based dominance rule) and a proper search strategy (the search strategy with a proper station-load selection criterion). The above findings also reveal the reasons leading to the high performance of the published BBR methods (Sewell and Jacobson 2012; Morrison et al. 2014) (mainly due to the memory-based dominance rule and a good search strategy). It also clarifies the reasons why re-implemented SALOME and BDP by Pape (2015) produce a clearly better performance (namely good search strategy).

## 4 Computational study

This section improves and re-implements three well-known exact methods (SALOME, BDP and BBR) using the effective parameters as experimented in Sect. 3. To our

**Table 15** The combinations of three parameters and the corresponding results by BBR methods

| Lower bound | Search strategy with station-load selection criterion | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|---|
| LBM4 | MCBFS-S4 | 1136 | 0.5517 | 8.3984 | 1.4648 | 136.98 |
| LBM4 | MCBFS-S6 | 1136 | 0.3441 | 6.1538 | 0.9636 | 136.42 |
| LBM4 | BSS-S4-10-5000 | 1135 | 0.5382 | 6.1753 | 1.3928 | 138.96 |
| LBM4 | BSS-S6-10-5000 | 1135 | **0.3221** | 5.2336 | 0.9011 | 139.11 |
| LBM7 | MCBFS-S4 | 1135 | 0.5561 | 7.4349 | 1.4692 | 137.76 |
| LBM7 | MCBFS-S6 | 1136 | 0.3444 | 6.1538 | 0.9651 | 136.94 |
| LBM7 | BSS-S4-10-5000 | 1135 | 0.5381 | 6.7308 | 1.3962 | 139.47 |
| LBM7 | BSS-S6-10-5000 | 1135 | 0.3231 | 5.4206 | 0.9060 | 139.53 |

Best value(s) in bold

best knowledge, the best versions of SALOME and BDP are reported in Pape (2015) which produce the best results; however, the codes are not available. Hence, this study presents two comparative studies to evaluate the re-implemented methods as follows. (1) The results by the re-implemented algorithms under three termination criteria (180 s, 500 s and 900 s) are compared with the published results in the literature. (2) The re-implemented algorithms are compared with the original version (if the code is available) and several versions utilizing the original parameter settings. All the re-implemented algorithms are coded in C ++ programming language, and experiments are conducted on a set of virtual machines (each has one virtual processor and 8 GB of RAM) in a tower type of server. The server is equipped with two Intel Xeon E5-2680 v2 processors (40 processor cores) at 2.8 GHz and 64 GB of RAM memory.

The details of these re-implemented methodologies are clarified as follows. Regarding the re-implemented SALOME, referred to ISALOME, the applied parameter values are: branching method with a maximum number of 10,000 station-loads (BM1-10,000) as branching method, bidirectional branching method (SD3) as search direction, MHH with 1000 station-loads (MHH-1000) as upper bound, LB1, LB2, LB3, LB6 and BPLB (LBM4) as lower bounds, maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule (DRM4) as dominance rules and modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda|U|$ (MCBFS-S6) as search strategy. Regarding the re-implemented BDP, referred to as IBDP, the applied parameter values are: branching method with a maximum number of 10,000 station-loads (BM1-10,000) as branching method, direction selection method (SD4) as search direction, MHH with 1000 station-loads (MHH-1000) as upper bound, LB1, LB2, LB3, LB6 and BPLB (LBM4) as lower bounds, only memory-based dominance rule as dominance rule and beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda|U|$ (BSS-S6-5-5000) as search strategy.

Regarding the re-implemented BBR, this study develops two versions, referred to as IBBR1 and IBBR2. Both IBBR1 and IBBR2 utilize the branching method with a maximum number of 10,000 station-loads and renumbering the tasks using positional weight (BM3-10,000) as branching method, direction selection method (SD4) as search direction, MHH with 1000 station-loads (MHH-1000) as upper bound, LB1, LB2, LB3, LB6 and BPLB (LBM4) as lower bounds, maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule (DRM4) as dominance rules. Nevertheless, IBBR1 utilizes the modified cyclic best-first search with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda|U|$ (MCBFS-S6) as search strategy, whereas IBBR2 utilizes the beam search strategy with $b(\wp) = \mathrm{LB}(U) + I/m - \lambda|U|$ (BSS-S6-10-5000) as search strategy.

## 4.1 Comparison with published results

To the authors' best knowledge, the recently re-implemented SALOME by Pape (2015), referred to as SALOME-Pape, is currently the best version of the SALOME algorithm. The re-implemented BDP by Pape (2015), referred to as BDP-Pape, is the current best version among the studies on BDP. The BBR by Morrison et al. (2014), referred to as BBR-Morrison, is the current best version of BBR method which also produces achieving competing results. Hence, this section compares the results published by these three methods with the results by re-implemented methods in this paper. Notice that the SALOME-Pape and BDP-Pape were run on a 2.8 GHz processor with 4 GB of RAM, and the BBR-Morrison was run on a single core of an Intel Core i7-930 2.8 GHz processor with 12 GB of RAM.

Table 16 presents the detailed results published and that by re-implemented methods for Scholl's 269 instance (referred to as Scholl-269), Otto-100 with 100 tasks and Otto-1000 with 1000 tasks. Here, RPD reports the average of the RPD values for all the tested instances and the tested methods terminate when the optimal solution is found and verified or the maximum computation time limit is reached. From this table, it is clear that the re-implemented ISALOME is enhanced greatly by using these high-performing parameters and it achieves all the optimal solutions for Scholl-269 instances for the first time. Additionally, it outperforms the SALOME-Pape (Pape 2015) for both Otto-100 and Otto-1000 by a significant margin with the allowed maximum time of 900 s. Regarding the re-implemented IBDP, it is also improved greatly by achieving all the optimal solutions for Scholl-269 instances and outperforming BDP-Pape (Pape 2015) for both Otto-100 and Otto-1000 with shorter running time. As for the two IBBR methods, they outperform the BBR-Morrison (Morrison et al. 2014) for both Otto-100 and Otto-1000 with shorter running times. Surprisingly, IBBR1 improves the RPD value from 1.21 to 0.84 for Otto-1000; IBBR1 improves the RPD value from 1.21 to 0.78 for Otto-1000 with less running times. Among all the exact methods in Table 16, it is observed that BBR1 and BBR2 are the best two performers, where the BBR2 with beam search strategy produces clearly better performance for Otto-1000.

## 4.2 Comparison among re-implemented methods

To further evaluate the improvements proposed by the re-implemented algorithms, this section compares the re-implemented algorithms with the original version (if the code is available) and several versions utilizing the original parameter settings.

To evaluate the ISALOME, two versions of SALOME are tested here: (1) SALOME-O1 where the depth-first search in Scholl and Klein (1997) is utilized; (2) SALOME-O2 where the station-load selection criterion in Scholl and Klein (1997) ($b(\wp) = \mathrm{LB}(U)$) is utilized. The proposed IBDP is compared with the two versions of the BDP method: (1) BDP-O1 where all the instances are solved in the forward direction as proposed in Bautista and Pereira (2009); (2) BDP-O2 where the station-load selection criterion in Pape (2015) ($b(\wp) = I$) is utilized. The proposed IBBR methods are compared with two versions of the BBR method: (1) BBR-O1 where all the parameters are set as that in Morrison et al. (2014); (2) BBR-O2 where the original station-load selection criterion in Morrison et al. (2014) ($b(\wp) = I/m - \lambda|U|$) is utilized. Notice that the BBR-O1 corresponds to the original BBR method in Morrison et al. (2014), which might be regarded as the current best exact algorithm. BBR-O1 utilizes the branching method with a maximum number of 10,000 station-loads (BM1-10,000) as branching method, direction

**Table 16** Comparison between published results and the results by re-implemented methods

| Instances | Scholl-269 | | | Otto-100 | | | Otto-1000 | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | #OPT | RPD | Time limit (s) | #OPT | RPD | Time limit (s) | #OPT | RPD | Time limit (s) |
| SALOME-Pape | 259 | – | 180 | 441 | 0.41 | 180 | 323 | 1.09 | 900 |
| ISALOME | 268 | 0.01 | 180 | 502 | 0.09 | 180 | 346 | 1.20 | 180 |
| | 269 | 0.00 | 500 | 509 | 0.06 | 500 | 348 | 1.02 | 500 |
| | **269** | **0.00** | 900 | **512** | **0.05** | 900 | **349** | **0.93** | 900 |
| BDP-Pape | 268 | – | 180 | 494 | 0.13 | 180 | 350 | 1.38 | 900 |
| | – | – | – | 500 | 0.09 | 3600 | 350 | 1.20 | 3600 |
| IBDP | 269 | 0.00 | 180 | 504 | 0.10 | 180 | 344 | 1.09 | 180 |
| | 269 | 0.00 | 500 | 514 | 0.04 | 500 | 350 | 0.96 | 500 |
| | **269** | **0.00** | 900 | **514** | **0.04** | 900 | **350** | **0.90** | 900 |
| BBR-Morrison | 269 | 0.00 | 3600 | 515 | 0.04 | 3600 | 350 | 1.21 | 3600 |
| IBBR1 | 269 | 0.00 | 180 | 510 | 0.06 | 180 | 350 | 1.09 | 180 |
| | 269 | 0.00 | 500 | 517 | 0.03 | 500 | 350 | 0.93 | 500 |
| | 269 | 0.00 | 900 | **517** | **0.03** | 900 | 350 | 0.84 | 900 |
| IBBR2 | 269 | 0.00 | 180 | 506 | 0.08 | 180 | 348 | 1.17 | 180 |
| | 269 | 0.00 | 500 | 516 | 0.03 | 500 | 350 | 0.86 | 500 |
| | 269 | 0.00 | 900 | 516 | 0.03 | 900 | **350** | **0.78** | 900 |

Best value(s) in bold

selection method (SD4) as search direction, MHH with 10,000 station-loads (MHH-10,000) as upper bound, LB1, LB2, LB3 and BPLB (LBM3) as lower bounds, maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule (DRM4) as dominance rules and cyclic best-first search with $b(\wp) = I/m - \lambda|U|$ (CBFS-S4) as search strategy.

Table 17 presents the detailed results by all the re-implemented methods. From this table, it is observed that the proposed ISALOME outperforms SALOME-O1 and SALOME-O2 under all the three termination criteria, demonstrating the proper parameters enhance the performance of SALOME by a significant margin. Similarly, the proposed IBDP outperforms BDP-O1 and BDP-O2 and the proposed IBBR1 and IBBR2 outperform BBR-O1 and BBR-O2 under all the three termination criteria. In summary, all the re-implemented exact methods benefit from the proper parameter setting, and these re-implemented exact methods show clearly better performance than their original editions. Moreover, re-implemented BBR methods again achieve the best performance, and they might be regarded as the new state-of-the-art exact methods for SALBP-I.

## 5 Conclusion and future research

This study presents a comparative study of exact methods in solving Type I simple assembly line balancing problem (SALBP-I) to minimize the number of workstations within the given cycle time. The various structural parameters (including branching method, search direction, method to achieve upper bounds, utilized lower bounds, utilized dominance rules and search strategy) are carefully

**Table 17** Comparison between the SALOME methods, BDP methods and BBR methods

| Time limit | Evaluation criteria | #OPT | RPD-Avg | RPD-Max | RPD-Var | CPU-Avg |
|---|---|---|---|---|---|---|
| 180 s | SALOME-O1 | 1067 | 0.7651 | 10.3846 | 1.8775 | 39.92 |
| | SALOME-O2 | 1087 | 0.7474 | 10.2857 | 1.9334 | 39.15 |
| | ISALOME | 1116 | 0.5150 | 8.1905 | 1.3619 | 35.24 |
| | BDP-O1 | 1105 | 0.4963 | 7.6923 | 1.2463 | 38.80 |
| | BDP-O2 | 1119 | 0.7441 | 7.8000 | 1.8560 | 34.36 |
| | IBDP | 1117 | 0.4715 | 7.6923 | 1.2281 | 34.33 |
| | BBR-O1 | 1125 | 0.5983 | 7.9696 | 1.5471 | 32.14 |
| | BBR-O2 | 1128 | 0.6773 | 8.3984 | 1.7463 | 30.54 |
| | IBBR1 | 1129 | **0.4569** | 7.4144 | 1.2425 | 30.36 |
| | IBBR2 | 1123 | 0.4978 | 10.1392 | 1.3162 | 31.86 |
| 500 s | SALOME-O1 | 1093 | 0.7067 | 10.3846 | 1.8331 | 103.43 |
| | SALOME-O2 | 1102 | 0.6737 | 9.1797 | 1.7938 | 99.85 |
| | ISALOME | 1126 | 0.4322 | 7.0342 | 1.1759 | 90.99 |
| | BDP-O1 | 1125 | 0.4141 | 5.9813 | 1.0936 | 95.97 |
| | BDP-O2 | 1130 | 0.6778 | 7.8000 | 1.7249 | 85.49 |
| | IBDP | 1133 | 0.4012 | 5.9813 | 1.0862 | 85.67 |
| | BBR-O1 | 1131 | 0.5381 | 7.9696 | 1.4144 | 82.68 |
| | BBR-O2 | 1136 | 0.5870 | 8.3984 | 1.5503 | 78.80 |
| | IBBR1 | 1136 | 0.3800 | 6.3462 | 1.0614 | 78.53 |
| | IBBR2 | 1135 | **0.3543** | 5.5888 | 0.9810 | 80.62 |
| 900 s | SALOME-O1 | 1099 | 0.6910 | 10.3846 | 1.8159 | 177.37 |
| | SALOME-O2 | 1103 | 0.6382 | 8.8462 | 1.6960 | 172.64 |
| | ISALOME | 1130 | 0.3882 | 6.4762 | 1.0709 | 156.71 |
| | BDP-O1 | 1128 | 0.3872 | 5.9813 | 1.0392 | 163.76 |
| | BDP-O2 | 1132 | 0.6465 | 7.2835 | 1.6567 | 146.04 |
| | IBDP | 1133 | 0.3775 | 5.9813 | 1.0306 | 146.50 |
| | BBR-O1 | 1132 | 0.4965 | 7.6172 | 1.3033 | 144.09 |
| | BBR-O2 | 1136 | 0.5517 | 8.3984 | 1.4648 | 136.98 |
| | IBBR1 | 1136 | 0.3441 | 6.1538 | 0.9636 | 136.42 |
| | IBBR2 | 1135 | **0.3221** | 5.2336 | 0.9011 | 139.11 |

Best value(s) in bold

investigated. Computational study shows that different parameters lead to quite different performances and a proper parameter setting significantly improves the performance of an exact method. Based on the structural parameter evaluation, three well-known exact methods are improved and re-implemented [well-known SALOME, bounded dynamic programming (BDP) heuristic and branch, bound and remember (BBR) algorithm]. The comparative study indicates that the re-implemented algorithms show clearly better performance than the original editions. Among these methods, the improved BBR is the best performer by outperforming the published results clearly and might be regarded as the new state-of-the-art exact algorithm.

During the investigation of the structural parameters, several conclusions are achieved as follows.

1. Regarding branching method, the branching method with 10,000 station-loads produces the best performance. The task renumbering methods show better performance, where renumbering the tasks using positional weight is the best performer.
2. Regarding the search direction, the direction selection method by Sewell and Jacobson (2012) produces the best results. The bidirectional branching method by Scholl and Klein (1997) is outperformed by the direction selection method probably due to the larger search space needed by the bidirectional branching method.
3. Regarding the methods to achieve upper bounds, they show a similar performance when being embedded into the BBR method although they obtain quite different upper bounds. Also, utilizing much time to achieve tighter upper bounds might result in increased overall running time (time for obtaining an upper bound and executing BBR procedure).
4. Regarding lower bound, the utilization of bin packing lower bound and LB6 produces better results. Nevertheless, using LB4, LB5 and LB8 leads to a poor performance as much time is consumed to calculate the lower bounds.
5. Regarding the dominance rule, the memory-based dominance rule is quite effective and the utilization of the memory-based dominance rule improves BBR's performance. The utilization of more dominance rules also enhances the performance of BBR method.
6. Regarding the search strategy, the station-load selection criterion is quite important and the criterion of $b(\wp) = \mathrm{LB}(U) + I/m - \lambda|U|$ is the best performer, where $I$ is the total idle time in the former $m$ stations, $\lambda$ is an input number (set to 0.02), $\mathrm{LB}(U)$ is the lower bound of unassigned task set $U$, and $|U|$ is the number of unassigned tasks. Cyclic best-first search, modified cyclic best-first search and beam search strategy show clear superiority over the depth-first search strategy and the best-first search strategy.

In future, the findings of this research might be applied to enhance the BBR methods in other variants of the assembly line balancing problem, including U-shaped assembly lines, two-sided assembly lines (Li et al. 2017), multi-manned lines (Kucukkoc et al. 2019) and many others. Future research might take into account other tighter time consuming lower bounds summarized by Scholl and Klein (1997) and Pereira (2015) and investigate the sequence of utilizing the lower bounds and dominance rules. The memory-based rule might be enhanced by combining with other dominance rules to compare two sub-solutions stored and eliminate the dominated one. It is also suggested to study the relationship between the problem complexity and the necessary number of station-loads at each depth and subsequently develop more proper settings of the number of station-loads (rather than 10,000 for all instances).

**Acknowledgements** This study is supported by National Natural Science Foundation of China under Grant 61803287 and China Postdoctoral Science Foundation under Grant 2018M642928.

## Compliance with ethical standards

**Conflict of interest** Zixiang Li, Ibrahim Kucukkoc and Qiuhua Tang declare that they have no conflict of interest.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

## References

Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solution approaches. Int J Prod Econ 142(2):259–277

Bautista J, Pereira J (2007) Ant algorithms for a time and space constrained assembly line balancing problem. Eur J Oper Res 177(3):2016–2032

Bautista J, Pereira J (2009) A dynamic programming based heuristic for the assembly line balancing problem. Eur J Oper Res 194(3):787–794

Blum C (2008) Beam-ACO for simple assembly line balancing. INFORMS J Comput 20(4):618–627

Blum C (2010) Iterative beam search for simple assembly line balancing with a fixed number of work stations. arXiv preprint arXiv:1012.3273

Blum C, Miralles C (2011) On solving the assembly line worker assignment and balancing problem via beam search. Comput Oper Res 38(1):328–339

Borba L, Ritt M (2014) A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. Comput Oper Res 45:87–96

Borba L, Ritt M, Miralles C (2018) Exact and heuristic methods for solving the robotic assembly line balancing problem. Eur J Oper Res 270(1):146–156

Boysen N, Fliedner M, Scholl A (2007) A classification of assembly line balancing problems. Eur J Oper Res 183(2):674–693

Çil ZA, Mete S, Özceylan E, Ağpak K (2017) A beam search approach for solving type II robotic parallel assembly line balancing problem. Appl Soft Comput 61:129–138

Ege Y, Azizoglu M, Ozdemirel NE (2009) Assembly line balancing with station paralleling. Comput Ind Eng 57(4):1218–1225

Esmaeilbeigi R, Naderi B, Charkhgard P (2015) The type E simple assembly line balancing problem: a mixed integer linear programming formulation. Comput Oper Res 64:168–177

Fleszar K, Hindi KS (2003) An enumerative heuristic and reduction methods for the assembly line balancing problem. Eur J Oper Res 145(3):606–620

Hoffmann TR (1963) Assembly line balancing with a precedence matrix. Manag Sci 9(4):551–562

Hoffmann TR (1992) Eureka: a hybrid system for assembly line balancing. Manag Sci 38(1):39–47

Johnson RV (1988) Optimally balancing large assembly lines with "Fable". Manag Sci 34(2):240–253

Kellegöz T, Toklu B (2012) An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations. Comput Oper Res 39(12):3344–3360

Klein R, Scholl A (1996) Maximizing the production rate in simple assembly line balancing—a branch and bound procedure. Eur J Oper Res 91(2):367–385

Kucukkoc I, Zhang DZ (2015) Type-E parallel two-sided assembly line balancing problem: mathematical model and ant colony optimisation based approach with optimised parameters. Comput Ind Eng 84:56–69. https://doi.org/10.1016/j.cie.2014.1012.1037

Kucukkoc I, Li Z, Li Y (2019) Type-E disassembly line balancing problem with multi-manned workstations. Optim Eng. https://doi.org/10.1007/s11081-019-09465-y

Lapierre SD, Ruiz A, Soriano P (2006) Balancing assembly lines with tabu search. Eur J Oper Res 168(3):826–837

Li Z, Kucukkoc I, Nilakantan JM (2017) Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. Comput Oper Res 84:146–161

Li Z, Kucukkoc I, Zhang Z (2018) Branch, bound and remember algorithm for U-shaped assembly line balancing problem. Comput Ind Eng 124:24–35

Liu SB, Ng KM, Ong HL (2008) Branch-and-bound algorithms for simple assembly line balancing problem. Int J Adv Manuf Technol 36(1):169–177

Miralles C, García-Sabater JP, Andrés C, Cardós M (2008) Branch and bound procedures for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. Discrete Appl Math 156(3):352–367

Morrison DR, Sewell EC, Jacobson SH (2014) An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset. Eur J Oper Res 236(2):403–409

Nourie FJ, Venta ER (1991) Finding optimal line balances with OptPack. Oper Res Lett 10(3):165–171

Ogan D, Azizoglu M (2015) A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. J Manuf Syst 36:46–54

Otto A, Otto C, Scholl A (2013) Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. Eur J Oper Res 228(1):33–45

Pape T (2015) Heuristics and lower bounds for the simple assembly line balancing problem type 1: overview, computational tests and improvements. Eur J Oper Res 240(1):32–42

Pereira J (2015) Empirical evaluation of lower bounding methods for the simple assembly line balancing problem. Int J Prod Res 53(11):3327–3340

Pereira J (2018) The robust (minmax regret) assembly line worker assignment and balancing problem. Comput Oper Res 93:27–40

Pereira J, Álvarez-Miranda E (2018) An exact approach for the robust assembly line balancing problem. Omega 78:85–98

Sabuncuoglu I, Erel E, Tanyer M (2000) Assembly line balancing using genetic algorithms. J Intell Manuf 11(3):295–310

Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168(3):666–693

Scholl A, Klein R (1997) SALOME: a bidirectional branch-and-bound procedure for assembly line balancing. INFORMS J Comput 9(4):319–334

Scholl A, Klein R (1999) Balancing assembly lines effectively—a computational comparison. Eur J Oper Res 114(1):50–58

Scholl A, Voß S (1997) Simple assembly line balancing—heuristic approaches. J Heuristics 2(3):217–244

Sewell EC, Jacobson SH (2012) A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS J Comput 24(3):433–442

Sternatz J (2014) Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. Eur J Oper Res 235(3):740–754

Vilà M, Pereira J (2013) An enumeration procedure for the assembly line balancing problem based on branching by non-decreasing idle time. Eur J Oper Res 229(1):106–113

Vilà M, Pereira J (2014) A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Comput Oper Res 44:105–114

Wei N-C, Chao IM (2011) A solution procedure for type E simple assembly line balancing problem. Comput Ind Eng 61(3):824–830

Wu E-F, Jin Y, Bao J-S, Hu X-F (2008) A branch-and-bound algorithm for two-sided assembly line balancing. Int J Adv Manuf Technol 39(9):1009–1015

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. Eur J Oper Res 206(3):703–707

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