# Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem

Zixiang Li , Ibrahim Kucukkoc , J. Mukund Nilakantan

| | |
|---|---|
| PII: | S0305-0548(17)30061-8 |
| DOI: | 10.1016/j.cor.2017.03.002 |
| Reference: | CAOR 4207 |

| | |
|---|---|
| To appear in: | *Computers and Operations Research* |

| | |
|---|---|
| Received date: | 14 December 2016 |
| Revised date: | 8 March 2017 |
| Accepted date: | 9 March 2017 |

## Highlights:

- Heuristics and meta-heuristics proposed for TALBP-II are comprehensively reviewed.
- A set of encoding schemes and decoding procedures is summarized.
- New objective functions and an iterative search mechanism are developed.
- Eighteen meta-heuristics are evaluated on a set of benchmark problems.
- New best and optimum solutions of TALBP-II test problems are also achieved.

# Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem

Zixiang Li [a], Ibrahim Kucukkoc [b1], J. Mukund Nilakantan [c]

[a] *Industrial Engineering Department, Wuhan University of Science and Technology, China*.

*Email: zixiangliwust@gmail.com*

[b] *Industrial Engineering Department, Balikesir University, Cagis Campus, Balikesir 10145, Turkey.*

*Email: ikucukkoc@balikesir.edu.tr*

[c] *Department of Mechanical and Manufacturing Engineering, Aalborg University, Denmark.*
*Email: mnj@m-tech.aau.dk*

**Abstract:** This paper presents a comprehensive review and evaluation of heuristics and meta-heuristics for the two-sided assembly line balancing problem. Though a few reviews have been presented, some latest methods are not included and there is no comparison of the meta-heuristics in terms of their performances. Furthermore, since various kinds of encoding schemes, decoding procedures and objective functions have been applied, the results cannot be generalized and the published comparison might be unfair. This paper contributes to knowledge by comparing the published methods, ranging from well-known simulated annealing to recent published iterated local search, and evaluating the six encoding schemes, 30 decoding procedures and five objective functions on the performances of the meta-heuristics meanwhile. The experimental design approach is applied to obtain valid and convincing results by testing algorithms under four termination criteria. Computational results demonstrate that the proper selection of encoding scheme, decoding procedure and objective function improves the performance of the algorithms by a significant margin. Another unique contribution of this paper is that 15 new best solutions are obtained for the large-sized type-II two-sided assembly line balancing problem during the re-implementation and evaluation of the meta-heuristics tested.

[1] Corresponding author

**Keywords:** Assembly line balancing; two-sided assembly lines; survey; heuristics; meta-heuristics

## 1. Introduction

As a flow oriented production system, assembly lines have great ramifications in various industries, including the automotive and consumer electronics, among others. In an assembly line, a set of tasks is divided to and processed on a set of workstations. Each workstation operates the allocated tasks within a pre-determined and fixed time, referred to as cycle time. The assembly line balancing problem (ALBP) is to determine the allocation of tasks to workstations considering one or more optimization criteria [1].

On the basis of the layout of the assembly line, the assembly line balancing problems can be classified one-sided assembly line balancing problem and two-sided assembly line balancing problem (TALBP). The two-sided assembly line has great applications in assembling large-sized products such as cars, buses and trucks [2]. Different from one-sided assembly lines, the TALBPs can be characterized by a set of tasks that must be divided to and processed on a set of mated-stations, each containing two facing and opposite workstations. Two cooperative workers on each mated-station operate the tasks in parallel at both left and right sides. Due to the utilization of both sides, the tasks are portioned into three types: L-type tasks, R-type tasks and E-type tasks. L-type/R-type tasks must be allocated to the left/right side whereas E-type tasks are allocated to either left or right side. As far as the optimization criterion is concerned, the TALBP can be divided into three categories: TALBP-I with the workstation number minimization criterion, TALBP-II with the cycle time minimization criterion and TALBP-E with the line efficiency maximization criterion. For TALBP-I, the minimization of the number of mated-stations is also considered as an ultimate or additional goal in some studies.

Regarding the TALBP-I, there are *Nt*! possible task permutations if utilizing the task permutation oriented encoding, where *Nt* is the number of tasks. Nevertheless, there are a total number of $2^{Ne} \cdot Nt!$ possible solutions since the E-type tasks can be allocated to either side, where *Ne* is the number of E-type tasks [3]. Obviously, the search space for TALBP is larger than that of one-sided ALBP and, consequently, TALBP is much more complex than one-sided ALBP. Since the simple ALBP with workstation number minimization is already NP-hard [1], the more complex TALBP-I and TALBP-II also belong to the NP-hard category.

There are four certain constraints in TALBP: precedence constraint, cycle time constraint, assignment constraint and direction constraint. The first three (namely, precedence, cycle time and assignment constraints) are general for all assembly line balancing problems, whereas direction constraint is the particular constraint in TALBP. Notice that the TALBP is different from the ALBP with parallel multi-manned workstations [4] due to the existence of L-type and R-type tasks. For ALBP with parallel multi-manned workstations, tasks can be allocated to either side while L-type/R-type tasks in TALBP must be assigned to the left/right side. Due to the utilization of both sides and the precedence constraint, some idle times in the middle of workstations emerge, donated as sequence-dependent idle time. To illustrate the above statements, a small numerical example adapted from Purnomo, Wee [5] is provided. The input data (precedence relationships, task times and operation directions) of the TALBP consisting of 9 tasks are depicted in Figure 1. In this figure, nodes denote tasks, and the labels over them refer to operation times and preferred directions. The directed arrows correspond to the precedence relationships between tasks, i.e. the pointed task is an immediate successor of the starting task, where each task may have more than one immediate successor. During the task allocation process, the precedence constraint and operation direction constraint must be satisfied as well as the capacity constraint. Figure 2 presents the detailed balancing configuration of tasks after balancing. As seen, all the L-type/R-type tasks are allocated to the left/right side and the starting time of each task is bigger than or equal to the ending times of its predecessors. For instance, tasks 2 and 3 are immediate predecessors of task 6. Therefore, task 6 can only be initialized after the completion of tasks 2 and 3, resulting in one-unit sequence-dependent idle time (see mated-station 2). The sequence-dependent idle time can be eliminated or reduced by optimizing the task sequence on workstations [3]. All in all, TALBP needs to determine the task assignment to mated-stations, the allocated sides of E-type tasks and the task sequence on workstations.

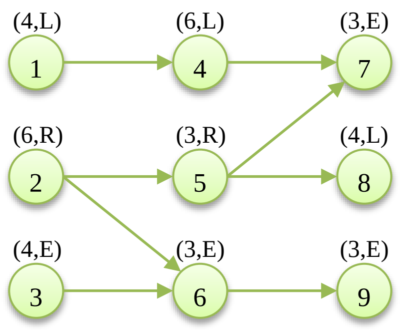

**Figure 1** Input data for the 9-task problem

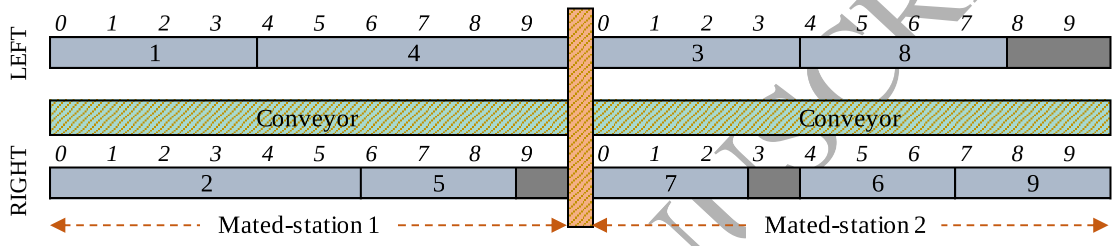

**Figure 2** The balancing configuration of tasks for the 9-task problem

The applied methods to TALBP can be divided into three categories: exact methods, heuristic methods and meta-heuristic methods. This paper reviews all these methods, but mainly evaluates the meta-heuristics for simplicity. The main reason is that the re-implementation of exact methods or heuristic methods might not be necessary since re-implemented exact methods or heuristic methods always obtain the same results as the published ones. Regarding the meta-heuristic methods, they comprise the majority of the published methods, and they are the state-of-the-art methods, to our best knowledge, for both TALBP-I and TALBP-II [3, 6]. In the literature, several papers have presented algorithm comparisons whereas these evaluations have one main shortcoming. This is the possible unfairness caused by comparing the obtained results with the published ones. These comparisons ignore the different performances of the algorithms caused by the different encoding schemes, decoding procedures and objective functions. For instance, the study of Li, Tang [3] shows that the proper decoding scheme and objective enhance the performance of an algorithm by a significant margin, which might throw some published comparisons on algorithms into doubt. Apart from the above factor, the improper parameter setting and termination criterion might also cause possible unfairness. The authors hold that the parameters of all tested algorithms need to be calibrated carefully, and testing algorithms from short to very large computational times is a good method to avoid biased results and prejudiced comparison.

The primary purpose of this paper is to present an up-to-date review and evaluation of existing heuristics and meta-heuristics on TALBP-II. This paper evaluates a total number of 18 recent meta-heuristics, which cover most of the published ones. Considering the existence of many encoding schemes, decoding procedures and objective functions, the second purpose of this research is testing the impacts of these factors on the algorithms' performances, and subsequently providing guidelines for proper selection of encoding schemes, decoding procedures and objective functions. Specifically, six encoding schemes, 30 decoding procedures and five objective functions are reviewed and compared. These reviewed methods cover most of the published effective ones, and some of them are first developed and compared. For simplicity, these re-implemented algorithms mainly solve the TALBP-II since TALBP-II is less studied and TALBP-I has been studied well in Li, Tang [3] where all optimum solutions for large-sized problems are achieved. Notice that TALBP-II is more complex than TALBP-I, and TALBP-II has great applications for reconfiguration of the installed two-sided assembly lines [6, 7]. All the algorithms are carefully calibrated using an experimental design approach and tested under four termination criteria of four elapsed CPU times on the TALBP benchmark problems, ranging from the P9 with nine tasks to P205 with 205 tasks.

The rest of this paper is structured as follows. Section 2 deals with applied objective functions, and Section 3 reviews the encoding schemes and decoding procedures. Later on, Section 4 introduces the heuristics as well as meta-heuristics applied to TALBP. A comprehensive evaluation of various meta-heuristics is presented in Section 5. Finally, the conclusions are drawn in the last section together with the future research directions.

## 2. Objective function

Objective functions serve as the guide for the algorithm evolution, and they affect the performance of the algorithms to a great extent. The criteria of minimizing the number of workstations and minimizing the cycle time are the most applied objective functions, but they become ineffective when solving large-sized problems (consisting high-numbers of tasks). In fact, many achieved solutions have the same number of workstations or cycle time, and the general objectives cannot determine a better one.

As a consequence, some new objective functions turn up to differentiate these solutions and guide the search direction while preserving the ability to optimize the general objectives. These new objective functions are always blended with problem-specific knowledge. As the TALBP-I and TALBP-II are the two most studied problems in the literature, the objective functions applied for those will be presented in the following subsections, together with a newly introduced station-oriented function. However, it should also be noted here that Kucukkoc and Zhang [8] maximized the efficiency of parallel two-sided lines (type-E) minimizing the product of two conflicting objectives, i.e. cycle time and the number of workstations. The objective function used in their study can simply be represented with $Min\ Z = CT \times Ns$, where $Ns$ simplifies the second term -which corresponds to the total number of opened workstations- in the original formulation.

## 2.1 Objective functions for TALBP-I

The most applied objective function for TALBP-I is minimizing the weighted summation of the number of mated-stations and the number of workstations [9] as in Equation (1). In this expression, $Nm$ and $Ns$ are the number of mated-stations and the number of workstations, and $w_1$ and $w_2$ are weighting parameters.

$$\text{Min FitI1} = w_1 \times Nm + w_2 \times Ns \tag{1}$$

Another objective function is to balance the workloads in between workstations and minimize the workstation number [10] simultaneously as in Equation (2). In this expression, index $j$ refers to a mated-station belonging to the mated-station set $J$, and index $k$ denotes a side which is equal to 1 for the left side and 2 for the right side. $CT$ is the given cycle time and $t_i$ is the operation time of task $i$. The term $x_{ijk}$ indicates the allocation of task $i$, where $x_{ijk}$ is equal to 1 when task $i$ is allocated to workstation $(j,k)$ or equal to 0 when task $i$ is allocated to another workstation. Another relative objective function refers to Özcan and Toklu [11] where the workstation number and workload balance are also simultaneously optimized.

$$\text{Min FitI2} = \sqrt{\sum_{j\in J}\sum_{k=1,2}\left(CT - \sum_i t_i \times x_{ijk}\right)^2 / Ns} + \sum_{j\in J}\sum_{k=1,2}\left(CT - \sum_i t_i \times x_{ijk}\right)/Ns \tag{2}$$

Since there are many solutions having the same workstation number, Özcan, Gökçen [12] developed a new objective function, given in Equation (3). This expression makes some workstations endure more workload and others take less workload. Maximizing this objective function helps optimize the workstation number, but it can provide more information to differentiate solutions.

$$\text{Max FitI3} = \sum_{j\in J}\sum_{k=1,2}\left(CT - \sum_i t_i \times x_{ijk}\right)^2 \tag{3}$$

The fourth objective function, given in Equation (4), is a new station-oriented objective function [3] , where $ST_{jk}$ is the sum of task times on workstation $(j,k)$. This expression preserves the solutions with more workload on the former mated-stations. The former part in this expression is the general objective, and the second part inherits the feature of the objective in expression (3) to differentiate solutions. The basic idea behind this objective is that if more workload is allocated to the former mated-station, much fewer mated-stations to be needed to endure the remaining workload.

$$\text{Min FitI4} = w_1 \times Nm + w_2 \times Ns + \sum_{j\in J}\sum_{k=1,2}(Nm + 1 - j) \times \left(CT - ST_{jk}\right)/(CT \times Nm) \tag{4}$$

Considering the sequence-dependent idle times, this research develops a new station-oriented objective function, given in Equation (5), where $SF_{jk}$ is the completion time of the last task on workstation $(j,k)$. Notice that $SF_{jk}$ is larger than or equal to $ST_{jk}$ due to the existence of sequence-dependent idle times. The term $SF_{jk} - ST_{jk}$ aims at reducing sequence-dependent idle times.

$$\text{Min FitI5} = w_1 \times Nm + w_2 \times Ns + \sum_{j\in J}\sum_{k=1,2}(Nm + 1 - j) \times \left(\left(CT - ST_{jk}\right) + \left(SF_{jk} - ST_{jk}\right)\right)/(CT \times Nm) \tag{5}$$

The latter four objectives are developed to further differentiate the solutions and guide the search direction. Still, they also share the purpose of optimizing the workstation number. In summary, these five objective functions propose different evolutionary ways to achieve the same goal of optimizing the workstation number.

**2.2 Objective functions for TALBP-II**

The general objective function for TALBP-II is minimizing the cycle time [13] expressed with Equation (6).

$$\text{Min FitII1} = \text{CT} \tag{6}$$

In a recent study, Tang, Li [6] developed a new function, given in Equation (7), by constraining the operation time on stations within $[Cm-(CT-Cm)/2\,, Cm+(CT-Cm)/2]$, where *Cm* is the ideal average workload on workstations ($\mathrm{Cm}=\sum_{i\in I}t_i/(2\times Nm)$).

$$\text{Min FitII2} = \mathrm{CT}+\sum_{j\in J}\sum_{k=1,2}(Nm-j)\times\left(max\left(0, SF_{jk}-\frac{Cm+CT}{2}\right)+max\left(0, Cm-\frac{CT-Cm}{2}-SF_{jk}\right)\right)/(2\times CT\times Nm) \tag{7}$$

Subsequently, Li, Tang [14] developed a new one to minimize the cycle time and smoothen workload distribution, which is described in Equation (8). In this expression, $w_1,\ w_2,\ w_3$ and $w_4$ are four coefficients. This expression constitutes four parts where the first two parts aim at reducing idle times and the last two parts optimize the workload balance. One main feature of this objective is that it takes the sequence-dependent idle time reduction into account. Li, Tang [14] set the values of these coefficients to 10, 5, 1 and 1 respectively which makes the reduction of the cycle time have the highest priority.

$$\text{Min FitII3} = w_1\times\sum_{j\in J}\sum_{k=1,2}\left(CT-ST_{jk}\right)/2Nm+w_2\times\sum_{j\in J}\sum_{k=1,2}\left(SF_{jk}-ST_{jk}\right)/2Nm+w_3\times\sqrt{\sum_{j\in J}\sum_{k=1,2}\left(CT-ST_{jk}\right)^2/2Nm}+w_4\times\sqrt{\sum_{j\in J}\sum_{k=1,2}\left(SF_{jk}-ST_{jk}\right)^2/2Nm} \tag{8}$$

The two station-oriented evaluation functions in Section 2.1 are also modified to guide the search direction for TALBP-II using Equations (9) and (10). Be aware of that the second parts in Equations (9) and (10) are set to be much less 1.0, and thus the second part takes effect only when the solutions have the same cycle time values.

$$\text{Min FitII4} = \mathrm{CT}+\sum_{j\in J}\sum_{k=1,2}(Nm+1-j)\times\left(CT-ST_{jk}\right)/(CT\times Nm) \tag{9}$$

$$\text{Min FitII5} = \mathrm{CT}+\sum_{j\in J}\sum_{k=1,2}(Nm+1-j)\times\left(\left(CT-ST_{jk}\right)+\left(SF_{jk}-ST_{jk}\right)\right)/(CT\times Nm) \tag{10}$$

The latter four objectives aim at guiding the search to the right direction, but they also share the purpose of minimizing the cycle time. In summary, all of the five algorithms have the same purpose of optimizing the cycle times but through different ways.

## 3. Encoding and decoding procedures

Encoding and decoding procedures are typically designed to characterize the nature of the TALBP. They are basic components of optimization algorithms and introduced in sequence. This section summarizes six encoding schemes and 30 decoding procedures.

### 3.1 Encoding schemes

Generally speaking, there are three types of encoding schemes: sequence-oriented encoding, workstation-oriented encoding and partition-oriented encoding. For the TALBP, there are two options for applying each of the above three encoding schemes: (i) utilizing a side vector to determine the directions of E-type tasks [15] or (ii) utilizing the heuristic methods to determine the directions of E-type tasks in the decoding procedures [6, 11], resulting in a total number of six encoding schemes. These encoding schemes are summarized as follows.

- Sequence-oriented encoding (SOE): The code is the task sequence which determines the priority or the assignment sequence of tasks. All tasks are sequentially allocated to mated-stations based on the task sequence [6, 11].
- Sequence-oriented and direction-oriented encoding (SDOE): A new direction vector is introduced in sequence-oriented encoding to describe the allocated directions of tasks [15].
- Workstation-oriented encoding (WOE): The code is mated-station string which corresponds to the allocated mated-stations of tasks. If task $i$ is allocated to mated-station $j$, the $i$th component in the string is $j$ [5, 13].
- Workstation-oriented and direction-oriented encoding (WDOE): A new direction vector is introduced in workstation-oriented encoding to describe the allocated directions of tasks.
- Partition-oriented encoding (POE): Separators are introduced in sequence-oriented encoding for partitioning tasks into mated-stations. There are two vectors: task sequence vector and separator vector [16].
- Partition-oriented and direction-oriented encoding (PDOE): A new direction vector is introduced in partition-oriented encoding to describe the allocated directions of tasks.

For TALBP-I, the researches on sequence-oriented encoding constitute the majority of the published papers whereas the only applied method among the remained five ones is workstation-oriented encoding by Kim, Kim [17]. This reason lies in that the number of mated-station remains to be optimized for TALBP-I. Kim, Kim [17] embedded the workstation-oriented encoding into a genetic algorithm to solve small-sized problems. The large-sized problems are not solved and the genetic algorithm with workstation-oriented encoding is declared to be outperformed by a tabu search algorithm [11].

Regarding TALBP-II, there are four encoding schemes applied: sequence-oriented encoding [6], sequence-oriented and direction-oriented encoding [15], and workstation-oriented encoding [5, 13] and partition-oriented encoding [16]. There is a variant of the partition-oriented encoding in Li, Tang [18], where workstation vector and task sequence vector are employed to determine the task assignment. Notice that each pair of task sequence and separator vector in partition-oriented encoding corresponds to a pair of workstation vector and task sequence vector, and thus the encoding in Li, Tang [18] is emerged into partition-oriented encoding as a special case. These encoding schemes exhibit diverse performances, and Tang, Li [6] indicate that the sequence-oriented encoding outperforms the workstation-oriented encoding in Kim, Song [13]. In addition, Lei and Guo [15] demonstrate that variable neighborhood search algorithm with sequence-oriented and direction-oriented encoding outperforms the genetic algorithms with workstation-oriented encoding [13]. Apart from the comparison on sequence-oriented encoding and workstation-oriented encoding in Tang, Li [6], no research presents detailed comparison on decoding schemes.

### 3.2 Decoding procedures

Once encoding scheme is determined, the decoding procedure is applied to obtain a feasible solution. Since decoding methods for TALBP-I have been summarized in Li, Tang [3], this paper mainly presents the decoding procedures for TALBP-II. For clarity, the general idea of decoding procedure is simplified as follows.

**Step 1**: Open a new mated-station and go to Step 3.

**Step 2**: If all tasks have been allocated, terminate. Otherwise, open a new mated-station.

**Step 3:** Execute the following steps.

Step 3.1: Determine whether assignable tasks exist. If no tasks are assignable, go to Step 2. Otherwise, go to Step 3.2.

Step 3.2: Select an assignable task and determine the allocated side of the current mated-station as the current workstation.

Step 3.3: Allocate this task to the selected workstation.

Step 3.4: Update the remained capacities for both sides and go to Step 3.1.

Notice that the methods to determine the assignable tasks depend on the applied decoding schemes. If any of the WOE, WDOE, POE or PDOE is applied, a task is assignable when the corresponding mated-station in the decoding is equal to the current mated-station, ignoring the cycle time constraint in the decoding scheme. For SOE or SDOE, a task is considered to be assignable when all of its predecessors have been allocated and it can be finished within the cycle time for the former $Nm - 1$ mated-station. When the last mated-station is involved, a task is considered to be assignable when all of its predecessors have been allocated and the cycle time constraint is ignored to obtain a feasible solution [3].

After the assignable task set is determined, the next step is selecting a task and a side of the current mated-station. Nevertheless, there are many different methods. The possible task assignment procedures are listed as follows, of which eight are summarized by Li, Tang [3]:

- Task-to-workstation procedure-1 (TS1): An assignable task in the former position of the task sequence is selected. If it is an L-type or an R-type task, it is allocated to the left side or right side. If the selected task can be allocated to either side, a random side is selected.
- Task-to-workstation procedure-2 (TS2): This procedure differs from TS1 when the selected task can be allocated to either side. For TS2, the left (right) side is selected by default when the selected task can be allocated to either side.
- Task-to-workstation procedure-3 (TS3): This procedure differs from TS1 when the selected task can be allocated to either side. For TS3, the side with a larger remained capacity is selected as the current workstation or a side is randomly selected when both sides contain the same remained capacity.

- Task-to-workstation procedure-4 (TS4): This procedure differs from TS1 when the selected task can be allocated to either side. For TS4, the side with a larger remained capacity is selected as the current workstation or the left (right) side is selected by default when both sides contain the same remained capacity.
- Workstation-to-task procedure-1 (ST1): Both sides of the current mated-station are checked whether to be available to allocate tasks. If only one side is able to allocate tasks, this side is definitely selected. If both sides are able to allocate tasks, the side with the larger remained capacity is selected or a side is randomly selected when both sides have the same remained capacity. An assignable task in the former position of the task sequence is selected and allocated to the above select side.
- Workstation-to-task procedure-2 (ST2): This procedure differs from ST1 when both of the two sides are available to allocate tasks and they have the same remained capacities. For ST2, the left (right) side is selected by default when both sides have the same remained capacity.
- Workstation-to-task procedure-3 (ST3): This procedure differs from ST1 in the method of selecting an assignable task. For ST3, all of the assignable tasks are checked at first whether tasks which can be operated at the earliest possible time of the selected side exist among them. If this kind of tasks exist, the assignable tasks cannot be operated at the earliest possible time are removed from the assignable tasks set. Finally, the task in the former position of task sequence is selected and allocated to the selected side.
- Workstation-to-task procedure-4 (ST4): This procedure differs from ST1 in two aspects. For ST4, the left (right) side is selected by default when both sides are available to allocate tasks and they have the same remained capacity. Later on, all of the assignable tasks are checked at first whether tasks which can be operated at the earliest possible time of the selected side exist among them. If this kind of tasks exist, the assignable tasks cannot be operated at the earliest possible time are removed from the assignable task set. Finally, the task in the former position of task sequence is selected and allocated to the selected side.
- Workstation-to-task procedure-5 (ST5): This procedure differs from ST4 in the task selection process. Apart from the features of ST4 in the task selection process, ST5 gives priorities to tasks whose possible finishing times are larger than the ideal average workload on each station or *Cm*. If there are assignable tasks whose possible finishing times are larger than or equal to *Cm*, those

tasks whose finishing times are smaller than *Cm* is deleted from the assignable tasks set. Subsequently, the task in the former position of task sequence is selected and allocated to the selected side.

- Two vector procedure (TVP): This procedure differs from TS1 in employing direction vector to determine the sides of E-type tasks.

Among the 10 task assignment procedures, the former eight ones are taken from Li, Tang [3] and the ninth one is taken from Tang, Li [6], and the tenth is taken from Lei and Guo [15]. The former nine are applied when direction-oriented encoding is not applied, whereas the last one is applied only when direction-oriented encoding is applied. Table 1 summarizes the encoding schemes and the corresponding task assignment procedures. Specifically, there are 9, 1, 9, 1, 9, 1 task assignment procedures corresponding to SOE, SDOE, WOE, WDOE, POE and PDOE, respectively. Since the method to obtain feasible solutions in an encoding scheme is different from the others, there is a total of 30 decoding methods. It is worthwhile to point out that a task sequence is also necessary for WOE and WDOE in the decoding procedure, and it is determined based on the ranked positional weight that is the sum of the processing times for a task and all of its successors [13].

**Table 1** Summary of encoding schemes and task assignment procedures

| Encoding scheme | Task assignment procedure |
|---|---|
| SOE [6] | TS1, TS2, TS3, TS4, ST1, ST2, ST3, ST4, ST5 |
| SDOE [15] | TVP |
| WOE [13] | TS1, TS2, TS3, TS4, ST1, ST2, ST3, ST4, ST5 |
| WDOE | TVP |
| POE [16] | TS1, TS2, TS3, TS4, ST1, ST2, ST3, ST4, ST5 |
| PDOE | TVP |

The detailed description of encoding and decoding schemes can be found in the references cited in this section. All the detailed descriptions along with the codes in C++ programming language are available upon request. Depending on the comprehensive computational experiments to be presented in Section 5, these methods have a great impact on the performances of the algorithms.

## 4. Solution methods for TALBP

The methods applied to TALBP can be divided into three groups: exact methods, heuristic methods and meta-heuristic methods. This section first reviews the exact methods and heuristic methods, followed by the review on meta-heuristics published in the literature.

### 4.1 Exact and heuristic methods

Regarding exact methods, there are only three studies reported to solve TALBP-I [19-21]. Hu, Wu [20] developed a station-oriented enumerative algorithm and they addressed small-sized problems. Later on, Wu, Jin [19] and Xiaofeng, Erfei [21] proposed branch-and-bound algorithms to address large-sized problems.

Though many heuristic methods have been applied to simple ALBP [1], the applications of heuristic methods to TALBP are relatively small. Bartholdi [22] first developed a first-fit heuristic to minimize the workstation number, and this was the first method applied to TALBP. Later on, Lee, Kim [2] proposed a group assignment procedure, where a group of tasks is assigned at a time. This method was demonstrated to outperform the first-fit heuristic in Bartholdi [22]. Özcan and Toklu [23] utilized a heuristic approach, referred to as 2-COMSOAL/S, based on the computer method of sequencing operations for assembly lines. This heuristic method was able to find the optimum solutions for small-sized problems. Another relevant research was applying the first-fit rule in Bartholdi [22] to address TALBP-II, but the results by this heuristic methods were outperformed by a genetic algorithm in Kim, Song [13]. Heuristic methods have very fast speed to obtain a feasible solution, but the performances of them are not satisfying when solving large-sized problems. Notice that this section only reviews the heuristics applied to TALBP, please refer to Scholl and Becker [1] for more heuristic methods applied to simple ALBP.

### 4.2 Meta-heuristic methods

Meta-heuristic methods comprise the majority of the researches on TALBP, and they are of good choice to obtain satisfying results within acceptable computational time. These methods start with an initial solution or an initial population, and they evolve until a termination criterion is met. This section reviews the application of meta-heuristics to TALBP.

Kim, Kim [17] developed a genetic algorithm to address TALBP-I with positional constraints, which was the first application of a meta-heuristic to TALBP. The workstation-oriented encoding scheme and genetic operators were first presented. Nevertheless, this method was only tested by small-sized problems. Taha, El-Kharbotly [24] developed a new genetic algorithm with sequence-oriented encoding to address TALBP-I utilizing a hybrid crossover and a modified scramble mutation operator. This new genetic algorithm outperforms several other meta-heuristic methods in the experimental study. Kim, Song [13] developed a genetic algorithm with workstation-oriented encoding to address TALBP-II. They also introduced localized evolution and steady-state reproduction to enhance the performance of the genetic algorithm. This algorithm was proven to outperform the first-fit heuristic in Bartholdi [22] and the original genetic algorithm. Subsequently, Purnomo, Wee [5] extended this genetic algorithm to address TALBP-II with assignment restrictions. Kucukkoc and Zhang [25] utilized the genetic algorithm for parallel two-sided assembly line balancing and verified its performance through computational tests. Rabbani, Moghaddam [26] considered the mixed-model line configuration in a multiple U-shaped line concept and proposed a mixed-integer program for modelling the balancing problem of such a line. A genetic algorithm based approach was also proposed for solving large-sized instances with the aim of minimizing both cycle time and the number of stations.

Another application of the genetic algorithm was solving stochastic two-sided U-type assembly line balancing [27], where a heuristic priority rule-based procedure was embedded into this method.

Baykasoglu and Dereli [28] proposed an ant colony optimization algorithm to address TALBP-I with zoning constraints, and this paper introduced a pheromone quantity update mechanism. This algorithm outperformed genetic algorithm in Kim, Kim [17] and group assignment procedure in Lee, Kim [2]. Simaria and Vilarinho [29] improved the ant colony algorithm by employing two ants on both sides to simultaneously build a solution for TALBP-I and mixed-model TALBP-I. This improved ant colony algorithm outperformed the group assignment procedure in Lee, Kim [2]. Subsequently, Kucukkoc and Zhang [30] developed ant colony algorithm to address mixed-model parallel two-sided assembly line balancing and sequencing problems. Kucukkoc and Zhang [31] further tested the proposed ant colony algorithm by comparing three heuristic methods, and they blended it with the genetic algorithms to solve this problem [32]. In a latter work, Kucukkoc and Zhang [33] utilized this method to address Type-E parallel two-sided assembly line balancing problem and response surface methodology was applied to parameter calibration. This algorithm outperformed three heuristic methods in the computational study. The algorithm was further extended to solve mixed-model parallel two-sided assembly line balancing problem [34] and outperformed six heuristics used commonly in the line balancing domain.

Özcan and Toklu [11] constructed a tabu search algorithm for the TALBP-I with sequence-oriented encoding. Tabu search algorithm utilized a tabu list to avoid repeated local searches, and this method obtained better results than an ant colony algorithm in Baykasoglu and Dereli [28], which was the best performer until then. Özcan, Gökçen [12] later extended tabu search algorithm to address parallel TALBP-I with the sequence-oriented encoding. The results obtained by tabu search algorithm were reported but no comparative study was carried out.

Özcan and Toklu [35] developed a simulated annealing algorithm to tackle mixed-model TALBP-I using the sequence-oriented encoding. Simulated annealing algorithm was able to find the optimum solutions for small-sized problems, but the reported results were outperformed by hybrid honey bee mating optimization algorithm in Yuan, Zhang [36]. Simulated annealing could be regarded as a simple method, and it was extended to address stochastic TALBP-I by Özcan [37] and cost-oriented TALBP-I by Roshani, Fattahi [38]. Khorasanian, Hejazi [39] improved simulated annealing algorithm to address TALBP-I considering the relationship between tasks using a modified form of the insert method to generate a neighbor solution. Khorasanian, Hejazi [39] indicated that results by the improved simulated annealing algorithm outperformed the published ones by several other algorithms. Aghajani, Ghodsi [16] utilized simulated annealing algorithm to address robotic mixed-model TALBP-II and introduced new encoding to determine both the task assignment and robot allocation. Simulated annealing algorithm was shown to be able to find the optimum solution for small-sized problems. Recently, Li, Tang [18] developed a multi-objective edition to tackle robotic TALBP-II, and this algorithm outperformed the fast non-dominated sorting genetic algorithm. Jawahar, Ponnambalam [40] proposed a simulated annealing algorithm and a heuristic algorithm, called enumerative heuristic algorithm, for minimizing number of stations as well as unbalanced workload in a pareto-front notion.

Özbakır and Tapkan [41] proposed a bee algorithm, which belongs to swarm intelligence based meta-heuristics, to solve the fuzzy multi-objective TALBP-I. It was later extended to zone constrained

TALBP-I by Özbakır and Tapkan [10]. Özbakır and Tapkan [10] verified the performance of the proposed algorithm by comparing their results against those existing in the literature. Tapkan, Ozbakir [42] proposed bee algorithm and artificial bee colony algorithm to further address constrained TALBP-I, and their study showed that bee algorithm outperformed artificial bee colony algorithm. The bee algorithm was later used to solve constrained fuzzy multi-objective TALBP by Tapkan, Özbakır [43], where both fuzzy multi-objective and multiple constraints were studied. Recently, Tapkan, Özbakır [44] utilized bee algorithm and artificial bee colony algorithm to tackle parallel TALBP-I with walking times and the results by these two algorithms outperformed the reported ones by tabu search algorithm in Özcan, Gökçen [12]. Another relevant study was carried out by Tang, Li [6] to address TALBP-II using improved artificial bee colony algorithm. The improved algorithm was demonstrated to outperform nine other algorithms, including tabu search in Özcan and Toklu [11] and simulated annealing algorithm in Khorasanian, Hejazi [39]. This algorithm also updated 22 best solutions for TALBP-II benefiting from a sequence-oriented encoding and a new decoding procedure.

Chutima and Chimklai [45] introduced modified particle swarm optimization to tackle multi-objective mixed-model TALBP-I and this new meta-heuristic outperformed four other multi-objective optimization algorithms. Later, Delice, Kızılkaya Aydoğan [46] improved this particle swarm optimization by utilizing a new task selection mechanism for mixed-model TALBP-I, and the improved edition was proven to surpass the simulated annealing algorithm in Özcan and Toklu [35]. Subsequently, Delice, Aydoğan [47] extended the particle swarm optimization to address two-sided U-type assembly lines. All the above modified particle swarm optimization algorithms utilized task probability matrix to obtain feasible solutions and updated this probability matrix based on the global best or local best solutions. This mechanism was similar to that proposed by ant colony algorithm in Baykasoglu and Dereli [28]. Chiang, Urban [48] utilized a discrete particle swarm optimization algorithm to address stochastic TALBP-I, but no comparative study was carried out to compare this method with other algorithms. Another application of particle swarm optimization was applying a co-evolutionary particle swarm optimization algorithm to address robotic two-sided assembly line by Li, Janardhanan [49]. The mechanism for this co-evolutionary particle swarm optimization algorithm was different from the above modified particle swarm optimization algorithms. The research utilized crossover operator to move the particle to the global best and local best individuals, and the co-evolutionary edition contained two sub-swarms to tackle task assignment and robot allocation, respectively. The co-evolutionary method outperformed another five meta-heuristics proven in the research.

Tuncel and Aydin [50] presented a teaching-learning-based optimization algorithm to tackle TALBP-I with multiple constraints, but this algorithm was not compared with other published ones. Tang, Li [51] improved this algorithm by hybridizing variable neighborhood search to enhance local search capacity. The improved algorithm was shown to outperform the late acceptance hill-climbing algorithm in Yuan, Zhang [9]. Subsequently, Li, Zhang [52] improved the teaching-learning-based optimization algorithms by introducing a self-learning phase achieved by the late acceptance hill-climbing algorithm in Yuan, Zhang [9], where a multi-objective TALBP-I with multiple constraints was considered. The multi-objective edition was demonstrated to surpass the fast non-dominated sorting genetic algorithm. The hybrid teaching-learning based optimization algorithm was then applied to solving stochastic two-sided assembly line balancing problem by Tang, Li [53] embedding a new priority-based decoding approach.

Li, Tang [14] implemented an iterated greedy search algorithm to address TALBP-II with assignment restrictions, where heuristic initialization and strong local search method were developed. Li, Tang [14] stated that iterated greedy search was an effective and simple algorithm with few parameters. The computational study demonstrated that the iterated greedy search algorithm outperformed eight recent algorithms, including tabu search algorithm in Özcan and Toklu [11], teaching-learning-based optimization algorithm in Tang, Li [51] and modified particle swarm optimization in Chutima and Chimklai [45]. In a later research, Li, Tang [3] extended this method to TALBP-I by employing a new local search method, which outperformed nine algorithms and achieved the optimum solutions for all the large-sized problems. The TALBP-II was handled allowing parallel performance of tasks by Sepahi and Naini [54]. The problem was formulated as a linear programing model and a robust heuristic procedure was proposed.

Apart from the above algorithms, Yuan, Zhang [9] proposed a late acceptance hill-climbing algorithm for TALBP-I with multiple constraints, and this algorithm was able to find the optimum solutions for small-sized problems. Wang, Guan [55] implemented a hybrid imperialist competitive algorithm to solve TALBP-I with multiple constraints, and this algorithm outperformed the late acceptance hill-climbing algorithm in Yuan, Zhang [9]. Li, Zhang [56] adopted the multi-objective imperialist competitive algorithm to tackle the mixed-model TALBP-I. Yang, Zhang [57] introduced multi-neighborhood-based path relinking for TALBP-I, and the results obtained by this path relinking were better than the published ones of several compared meta-heuristics. Lei and Guo [15] conducted a variable neighborhood search for TALBP-II and the main feature of this algorithm was the application of sequence-oriented and direction-oriented encoding. The results of this variable neighborhood search updated many reported results for TALBP-II achieved by genetic algorithm in Kim, Song [13]. Purnomo and Wee [7] implemented a multi-objective harmony search algorithm to address bi-objective TALBP-II and this algorithm outperformed the fast non-dominated sorting genetic algorithm proven in the computational study. A modified discrete cuckoo search algorithm, enhanced with a new individual generation procedure, was proposed by Li, Dey [58] for balancing two-sided robotic assembly lines. As the proposed algorithm deals with two sub-problems, it employs two sub-swarms, each addressing a sub-problem. The performance of the proposed approach was verified by with the results of computational tests and their statistical analysis. Recently, Abdullah Make, Ab. Rashid [59] presented a survey on two-sided assembly line balancing problems. The objective functions and constraints considered in various studies have been investigated including the optimization methods employed, but no computational evaluation of those methods has been conducted.

## 5. Comparative evaluation of meta-heuristics

This section mainly presents the comparative study of the meta-heuristics on solving TALBP-II. Nevertheless, the evaluation of meta-heuristics is far from simple due to so many encoding schemes, decoding procedures and objective functions. In fact, there are 150 different configurations for each algorithm if employing all 30 decoding procedures and 5 objective functions, leading to extremely large scale experiments. However, the researches mainly care about the effective ones, and not all the configurations are needed to be tested. For that reason, this section first evaluates the objective functions, encoding schemes and decoding procedures by embedding them into a simulated annealing algorithm as an example. One of the effective combination is selected and embedded into the other algorithms. Later on, 18 algorithms selected from the meta-heuristics reviewed in Section 4.2 are evaluated with statistical analysis. For some algorithms, several variants of them are also included to further analyze the features of the algorithms.

Selecting the tested benchmark problems and the termination criterion are also important. This paper utilizes widely applied benchmarks for TALBP, which is summarized by Tang, Li [6] for TALBP-II. These benchmark problems are divided into two portions: small-sized problems (variants of P9, P12, P16 and P24) and large-sized problems (variants of P65, P148 and P205). There is a total of 39 cases studied, please refer to Tang, Li [6] regarding the detailed mated-station numbers for TALBP-II. The remaining problem is determining a proper termination criterion, and this paper employs the termination criterion utilized by Tang, Li [6] and Li, Tang [14] which is a pre-determined elapsed CPU time limit of $Nt \times Nt \times \tau$ milliseconds, where $\tau$ is an input parameter. To examine the performance of the algorithms from small to very large computational times, all the algorithms are tested under four termination criteria, namely $\tau$ is set to be equal to 5, 10, 15 and 20 respectively. All the methods are coded in C++ language on Microsoft Visual Studio 2012 platform and all the experiments are carried out on a set of computers equipped with Intel(R) Core2(TM) CPU 2.33 GHZ, 3GB of RAM. All the pseudocodes and the programming in C++ language of all the tested algorithms are available upon request.

Note that the decoding procedures need an initial cycle time when utilizing encoding scheme SOE and SDOE. This paper set the initial cycle time for decoding as $\beta \times \mathrm{Cm}$ following Tang, Li [6]. The initial cycle time for decoding is updated with $CT_{Best} - 1$, where $CT_{Best}$ is the best cycle time among all individuals during the evolution process. This expression makes sure that the best cycle time reduces gradually. However, when a new best cycle time is achieved, the solutions achieved using the current encoding codes usually obtain much larger cycle times observed in preliminary experiments. The main reason lies behind is that a mated-station is allocated with as more workload as possible and the current task assignment suits the current best cycle time. This situation leads to the poor performance of the algorithm using greedy acceptance, where the original cycle times achieved using the original best cycle time are compared with the new cycle times achieved using the current best cycle time. To overcome this possible drawback, this paper introduces a new method, referred to as iterative mechanism, where the current function values of the individuals are replaced with the new function values achieved using $CT_{Best} - 1$. This method guarantees that the incumbent individuals and the new individuals are compared using the same initial cycle time $CT_{Best} - 1$. The performance of the iterative mechanism will be tested in the following subsection.

## 5.1 Evaluation of encoding, decoding and objective function

This section presents the evaluation of the summarized objective functions, encoding schemes, decoding procedures and introduced iterative mechanism. There are 150 different configurations for each algorithm when employing different encoding and decoding schemes and objective functions, and this number increases to 200 if it is considered whether the iterative mechanism is applied. Specifically, iterative mechanism only takes effect on encoding schemes SOE and SDOE, and thus the increased number is calculated with $(9 + 1) \times 5 = 50$, where 9 and 1 are the numbers of decoding methods for encoding scheme SOE and SDOE, respectively and 5 is the number of objective functions. For simplicity, this section mainly presents the comparative study on these factors utilizing simulated annealing algorithm. Each configuration solves ten cases of the largest problem, P205, and where each case is solved for 10 times. The termination criterion is a pre-determined elapsed CPU time of $Nt \times Nt \times 5$ milliseconds. Since the cycle times for different cases are different, this research utilizes the Relative Percentage Deviation (RPD), calculated using Equation (11), to transfer the achieved cycle times.

$$\mathrm{RPD} = 100 \times (CT_{sol} - CT_{LB})/CT_{LB} \tag{11}$$

where $CT_{sol}$ is the cycle time achieved by a configuration for one case and $CT_{LB}$ is the lower bound of the cycle time calculated by Tang, Li [6] using Equation (12) for the same case. In Equation (12), $T_{Tsum}$, $T_{Lsum}$ and $T_{Rsum}$ are the sum of the operation times of all tasks, L-type tasks and R-type tasks. The term $t_{max}$ denotes the maximum of the operations times of all tasks.

$$CT_{LB} = \lceil max(T_{Tsum}/(2 \times Nm), T_{Lsum}/Nm, T_{Rsum}/Nm, t_{max}) \rceil \tag{12}$$

After achieving all of the RPD values, the average RPD value of all the solved cases in one run is utilized for comparison. Thus, 10 average RPD values are obtained for each configuration since each case is solved for 10 times, repeatedly (please see results provided as supplementary material). Since so many configurations are considered, a large body of experimental results are achieved. Due to page limit, this section mainly presents four sets of comparative studies, each studying a factor while fixing other factors to very effective values, listed as follows:

- Comparison of objective functions: Five objective functions are evaluated while utilizing SOE, ST5 and iterative mechanism.
- Comparison of utilizing the iterative mechanism: The results achieved by iterative mechanism are compared with and without iterative mechanism while utilizing SOE, ST5 and $FitII5$.
- Comparison of task assignment procedures: A total of 10 task assignment procedures is evaluated while utilizing SOE or SDOE, $FitII5$ and iterative mechanism.
- Comparison of encoding schemes: Six encoding schemes are evaluated while utilizing ST5 or TVP, $FitII5$ and iterative mechanism.

Notice that SOE corresponds to nine task assignment procedures and SDOE corresponds to only one task assignment procedure. SOE is utilized for the first nine task assignment procedures and SDOE is utilized for the tenth task assignment procedure in the comparison on task assignment procedures. Similarly, SOE, WOE and POE utilize ST5; and SDOE, WDOE and PDOE utilize TVP in comparison on encoding schemes.

To check whether there is a statistically significant difference between the levels of a factor, this paper utilizes parametric analysis of variance (ANOVA) method and the non-parametric Friedman rank-based analysis following Tang, Li [6] and Li, Tang [3]. The ANOVA method is employed when fulfilling independence of the residuals, homogeneity of the variance, and normality of the residuals. Otherwise, both the ANOVA method and Friedman test are proposed to detect the differences in the levels, where Friedman test is proposed to ascertain the results by ANOVA test. More description of the ANOVA test and Friedman test refers to the aforementioned cited papers. For both methods, the p-value is an important parameter to determine whether there is a significant difference among the levels. There is a significant difference when the p-value is smaller than a pre-determined number $\alpha$ ($\alpha$ is set to 0.05). The means plots for the four factors along with 95% confidence intervals are illustrated in Figure 3.

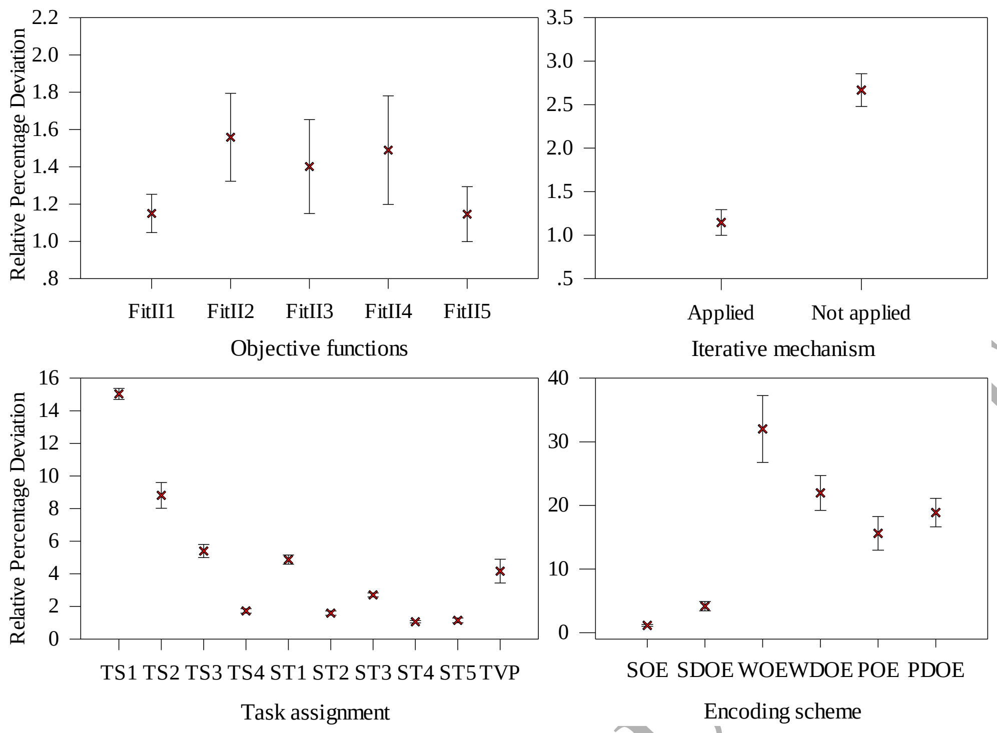

**Figure 3** Means plot and 95% Tukey HSD confidence intervals for the four tested factors

ANOVA test or Friedman test shows that there is a statistically significant difference among the levels for any tested factor. The statistical results demonstrate that the encoding scheme, decoding procedure and objective functions have statistically significant impact on the performance of an algorithm. This proves the necessity of studying encoding scheme, decoding procedure and objective functions. Regarding the objective functions, FitII1 and FitII5 are the two best performers. FitII1 outperforms FitII2, FitII3 and FitII4 with the assistance of the iterative mechanism. This finding seems to be conflicting with that in Tang, Li [6] and Li, Tang [14], but as a matter of fact this is not the case. Different from that the published ones, this research utilizes the iterative mechanism and thus the applied objectives in this paper show different performances from the published ones. The utilization of the iterative mechanism helps obtaining better results than that of not utilizing it. As for the task assignment procedures, the results coincide with that in Li, Tang [3], where TS4, ST2, ST4 and ST5 are the best performers. TVP, which is first compared in this paper, outperforms the TS1, TS2, TS3 and ST1 whereas it is outperformed by TS4, ST2, ST3, ST4 and ST5. This finding suggests that the utilization of an effective heuristic to decide on selected side of E-type tasks is more efficient than the utilization of the direction vector. However, the utilization of direction vector outperforms random selection of E-type tasks. Regarding the encoding schemes, the difference is the largest among the four factors. The smallest average RPD value of SOE is about 1.0 whereas the largest average RPD value of WOE is over 30.0. The reason lying behind this consequence is that WOE and WDOE utilize heuristics to determine the task sequence, which cannot reduce the sequence-dependent idle times effectively. All in all, the computational results suggest that the proposed iterative mechanism is very effective, and the encoding scheme of SOE and the task assignment procedure of ST5 are the good performers. These results also demonstrate that it is mandatory to select an effective objective function and task assignment procedure in the implementation of the algorithms.

Since so many combinations of the factor levels exist, one effective combination of SOE, ST5, FitII5 and the iterative mechanism is embedded into the tested algorithms for simplicity in the subsequent sections. Note that the comparison via utilizing other algorithms is omitted due to page limit, but the above combination is quite effective for all the tested algorithms.

**5.2 Compared algorithms**

This section presents the compared meta-heuristics which are selected from that stated in Section 4.2. The applied methods are summarized in Table 2. The selected algorithms include the well-known algorithms and the recent and effective ones. During the re-implementation process, some modifications are necessary and they are executed under two principles: the effectiveness and simplicity. Some reported algorithms are ineffective in our preliminary experiments, and thus they are improved, such as utilizing a new neighborhood structure. Others might be intricate where many improvements are introduced and many parameters need to be calibrated, and thus these methods are simplified while preserving high efficiency. Specifically, the two-point crossover operator [6] is utilized for crossover after testing one- point crossover operator and two-point crossover operator. The insert operator and swap operator are both applied and randomly selected in the neighborhood structure, which shows superior performance than only utilizing one neighborhood operator when computational time increases. Apart from these, this table also contains the important information in re-implementing the algorithms, which have great impact on the performances of the tested algorithms. Notice that the operators are designed based on the recent paper by Li, Tang [3], where a comprehensive evaluation of meta-heuristics is presented.

Apart from the operators, the other parameters also play an important role in the performances of the tested algorithms. The parameter values for some re-implemented algorithms are taken from Tang, Li [6] and Li, Tang [3] where the full factorial design is utilized for parameter calibration. The parameter values for other algorithms are determined utilizing the full factorial design following Tang, Li [6] and Li, Tang [3]. Please see Table A1 for the list of parameters used. For a specific algorithm, all the combinations of the parameter values are tested by solving ten cases of the largest-size problem P205. Each configuration solves these cases for ten times and the termination criterion is an elapsed CPU time limited to $Nt \times Nt \times 5$ milliseconds.

**Table 2** Summary of the tested algorithms

| Algorithm | Abbreviation | Information |
|---|---|---|
| Simulated annealing algorithm [39] | SA | - |
| Tabu search algorithm [11] | TS | - |
| Late acceptance hill-climbing algorithm [9] | LAHC | - |
| Iterated greedy algorithm [14] | IG1 | - |
| Iterated greedy algorithm [3] | IG2 | - |
| Iterated greedy algorithm [3] | IG3 | IG3 differs from the published one in that insert operator and swap operator are both applied and randomly selected in the local search procedure. |
| Genetic algorithm [13] | GA | The elitism strategy is applied, which clones the best individual to the offspring. |
| Teaching-learning-based optimization algorithm [51] | TLBO | - |
| Bee optimization algorithm [42] | BA | - |
| Artificial bee colony algorithm [42] | ABC | A scout is applied to replace the worst individual or one of the same individual with a randomly generated solution in current swarm when no improvement on the best solution is achieved. |
| Discrete artificial bee colony algorithm [6] | DABC | Crossover operator is applied in employed bee phase, and swap operator and insert operator are utilized in the onlooker phase. Tournament selection is applied in onlooker phase for selecting an individual. A scout was applied to replace the worst individual or one of the same individual with a randomly generated solution in the current swarm when no improvement on the best solution is achieved. |

| | | |
|---|---|---|
| Particle swarm optimization [60] | PSO1 | Random-keys method is utilized for encoding, and the elitism strategy is applied, which clones the best individual to the offspring. |
| Particle swarm optimization [49] | PSO2 | Crossover operator, insert operator and swap operator are utilized for population evolution. |
| Ant colony optimization algorithm [28] | ACO | - |
| Two-ant colony optimization algorithm [29] | 2ACO | - |
| Modified particle swarm optimization algorithm [45] | PSONG1 | - |
| Modified particle swarm optimization algorithm [45] | PSONG2 | Selection probability of task $j$ is set to be $p(j)=\frac{[PM_{ij}]^{\alpha}[\eta_j]^{\beta}}{\sum_{j\in A_i}[PM_{ij}]^{\alpha}[\eta_j]^{\beta}}$, where $PM_{ij}$ is the selection probabilities of the task $j$ immediately after the task $i$, $\eta_j$ is the ranked positional weight heuristic and $A_i$ is the set of assignable tasks after allocating task *i*. |
| Modified particle swarm optimization algorithm[45] | PSONG3 | PSONG inherits the features of 2ACO and PSONG, and two particles work simultaneously in left and right sides. |

The RPD is also utilized to transfer the achieved cycle times. The average RPD value of the ten solved cases after one run is utilized for comparison, and eventually, 10 average RPD values for each combination of parameter values are obtained. After achieving all the average RPD values, ANOVA method is applied to analyze the results and select the best combination of the parameters following Li, Tang [14] and Li, Janardhanan [49]. The readers might refer to the papers cited above for the detailed calibration process.

**5.3 Comparative evaluation of the algorithms**

This section analyzes the performances of different meta-heuristics. The RPD is again applied to transfer the achieved cycle times when solving different cases. Since the optimum cycle times for small-sized problems are already known, the RPD for small-sized problems is modified and calculated using Equation (12), where $CT_{OPT}$ is the optimum cycle time for a case which has been reported by Tang, Li [6].

$$\text{RPD} = 100 \times (CT_{sol} - CT_{OPT})/CT_{OPT} \qquad (12)$$

A total number of 39 cases summarized in Tang, Li [6] are solved, and each case is solved for 10 independent times by each algorithm. The average RPD values of tested algorithms are exhibited in

Table 3, where only the results of 13 best performers are presented (note that the detailed results are provided as supplementary material). Each cell in the table contains the average RPD value of several cases after ten times of independent run. For instance, each cell for P205 is the average value of $11 \times 10$ data, where 11 is the number of tested cases and 10 is the number of independent runs.

Under the first termination criterion ($\tau = 5$), the IG3 is the most effective with an overall RPD value of 0.25, and SA is the second best performer with an overall RPD value of 0.34. The other algorithms are ranked in an increasing order of the overall RPD values as follows: ABC, IG2, LAHC, TS, DABC, GA, BA, PSO1, PSONG2, IG1 and ACO. Regarding other three termination criteria ($\tau = 10$, $\tau = 15$ and $\tau = 20$), IG3 is also the best performer with overall RPD values of 0.21, 0.19 and 0.18, respectively. ABC is the second best performer with overall RPD values of 0.26. 0.23 and 0.21, respectively. IG2 is the third best performer with overall RPD values of 0.28. 0.25 and 0.22, respectively. For other algorithms, the ranking orders of the overall RPD values under three other termination criteria have some variation. Still, it is observed that they are similar to that under the first termination criterion. The computational results also suggest that the local search methods, IG2, IG3, SA, LAHC and TS outperform the population-based algorithms, including GA, BA, PSO1, ACO and PSONG2. It seems that the experimental results conflict with that reported in Tang, Li [6], which is attributed to the application of the iterative mechanism and the new objective function. In the iterative mechanism, the current function values of the individuals are updated when a new best cycle time is achieved. The iterative mechanism makes sure that the incumbent individual and the new individuals are compared using the same initial cycle time for decoding.

**Table 3** Average RPD values of compared algorithms

| Problem | SA | TS | LAHC | IG1 | IG2 | IG3 | GA | BA | ABC | DABC | PSO1 | ACO | PSONG2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tau = 5$ | | | | | | | | | | | | | |
| P9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P16 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P24 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P65 | 0.35 | 0.38 | 0.44 | 0.58 | 0.34 | **0.26** | 0.57 | 0.68 | 0.34 | 0.49 | 0.84 | 0.74 | 0.72 |
| P148 | 0.02 | 0.17 | 0.15 | 0.55 | 0.14 | **0.10** | 0.38 | 0.16 | 0.13 | 0.32 | 0.19 | 0.17 | 0.17 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P205 | 1.02 | 1.03 | 1.01 | 1.47 | 1.00 | **0.69** | 1.06 | 1.41 | 0.95 | 0.92 | 1.40 | 1.74 | 1.63 |
| Avg | 0.34 | 0.38 | 0.38 | 0.62 | 0.36 | **0.25** | 0.46 | 0.52 | 0.34 | 0.40 | 0.55 | 0.62 | 0.59 |
| $\tau = 10$ | | | | | | | | | | | | | |
| P9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P16 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P24 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P65 | 0.32 | 0.37 | 0.43 | 0.46 | 0.30 | **0.22** | 0.45 | 0.53 | 0.28 | 0.42 | 0.84 | 0.69 | 0.65 |
| P148 | 0.01 | 0.14 | 0.12 | 0.44 | 0.09 | **0.06** | 0.33 | 0.13 | 0.08 | 0.24 | 0.19 | 0.14 | 0.15 |
| P205 | 0.91 | 0.93 | 0.83 | 1.16 | 0.78 | **0.59** | 0.90 | 1.11 | 0.72 | 0.70 | 1.37 | 1.64 | 1.52 |
| Avg | 0.30 | 0.34 | 0.32 | 0.49 | 0.28 | **0.21** | 0.39 | 0.41 | 0.26 | 0.31 | 0.54 | 0.58 | 0.55 |
| $\tau = 15$ | | | | | | | | | | | | | |
| P9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P16 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P24 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P65 | 0.32 | 0.36 | 0.41 | 0.41 | 0.27 | **0.20** | 0.38 | 0.45 | 0.27 | 0.38 | 0.84 | 0.67 | 0.60 |
| P148 | 0.01 | 0.12 | 0.10 | 0.37 | 0.07 | **0.05** | 0.29 | 0.11 | 0.06 | 0.20 | 0.18 | 0.12 | 0.13 |
| P205 | 0.87 | 0.91 | 0.76 | 1.04 | 0.69 | **0.53** | 0.82 | 0.95 | 0.63 | 0.65 | 1.37 | 1.59 | 1.47 |
| Avg | 0.29 | 0.33 | 0.29 | 0.43 | 0.25 | **0.19** | 0.35 | 0.35 | 0.23 | 0.28 | 0.54 | 0.56 | 0.52 |
| $\tau = 20$ | | | | | | | | | | | | | |
| P9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P16 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P24 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| P65 | 0.32 | 0.35 | 0.40 | 0.37 | 0.23 | **0.19** | 0.33 | 0.39 | 0.26 | 0.37 | 0.84 | 0.66 | 0.59 |
| P148 | 0.01 | 0.12 | 0.10 | 0.31 | 0.07 | **0.05** | 0.27 | 0.09 | 0.05 | 0.18 | 0.18 | 0.11 | 0.12 |
| P205 | 0.85 | 0.89 | 0.72 | 0.94 | 0.63 | **0.51** | 0.76 | 0.85 | 0.58 | 0.61 | 1.37 | 1.55 | 1.44 |
| Avg | 0.28 | 0.32 | 0.28 | 0.38 | 0.22 | **0.18** | 0.32 | 0.31 | 0.21 | 0.26 | 0.54 | 0.55 | 0.51 |

* Best average RPD for each problem is given in bold.

Though the observed difference among the algorithms in Table 3 is quite clear, it is still suggested to check whether this difference is statistically significant using a statistical technique. Again, multifactor

ANOVA is employed where algorithm type and computational time are selected as two factors. Since the performance of algorithms on different problems is quite large, the average RPD value of 39 cases in one run is employed in the ANOVA test. Since each case is solved for ten independent times, there are 10 average RPD values for each combination of the two factors. Nevertheless, an initial ANOVA test shows that the normality of the residuals is violated which is attributed to the big difference in the performances of eighteen algorithms. The non-parametric Friedman test might be a good choice, but it cannot analyze the interactions between the algorithm type and computational time. For the above reasons, this research utilizes both the ANOVA test and Friedman test, where Friedman test is employed to check and strengthen the findings of the ANOVA test.

ANOVA analysis suggests that there are statistically significant differences between the algorithms, computational time and interaction of two tested factors. Detailed ANOVA results are omitted here (provided as supplementary material instead), and the means plot of the interaction of two tested factors is depicted in Figure 4. To have a better picture of the results, only the best seven algorithms are depicted in Figure 4. This figure provides a direct and clear observation into the performances of the algorithms.

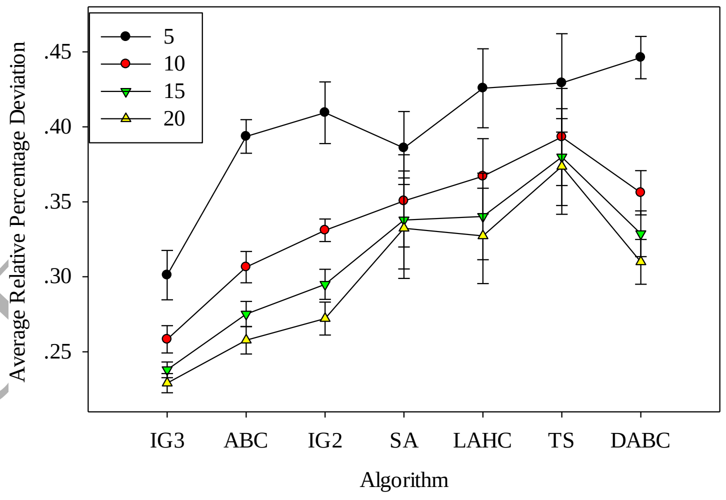

**Figure 4** Means plot and 95% Tukey HSD confidence intervals for the interactions tested algorithms and maximum elapsed CPU time

It is clear that IG3 is the best performer from the beginning to the end, and the results slightly improve with increasing CPU time after CPU time reaches $Nt \times Nt \times 15$ milliseconds. This situation is attributed to that the results by IG3 are very close to the lower bounds and improvements become much more difficult. Regarding the ABC method, the achieved results improve by a significant margin with CPU time increasing, and it becomes the second best performer after CPU time reaches $Nt \times Nt \times 10$ milliseconds. As for SA, LAHC and TS, the improvements are quite clear before reaching $Nt \times Nt \times 10$ milliseconds whereas little improvements are achieved after that. Regarding the ABC, IG2 and DABC method, the improvement in the obtained cycle times with the increasing CPU time is seen clearly.

Let us focus on the variants of PSONG and ACO (ACO, 2ACO, PSONG1, PSONG2 and PSONG3). These five algorithms are different from the others in that they select a task based on probabilities rather than encoding scheme. The means plot of the interaction of the five algorithms and computational times is depicted in Figure 5. It is observed that PSONG2 is the best performer, while 2ACO, ACO and PSONG3 are the second, third and fourth best performers, respectively. PSONG1 is the worst performer. The computational results yield following findings:

- Embedding heuristic information into the PSONG2 improves the performance of original PSONG.
- PSONG2 shows superiority over the ACO, and it might be strong competitor to ACO which also selects a task based on probabilities.

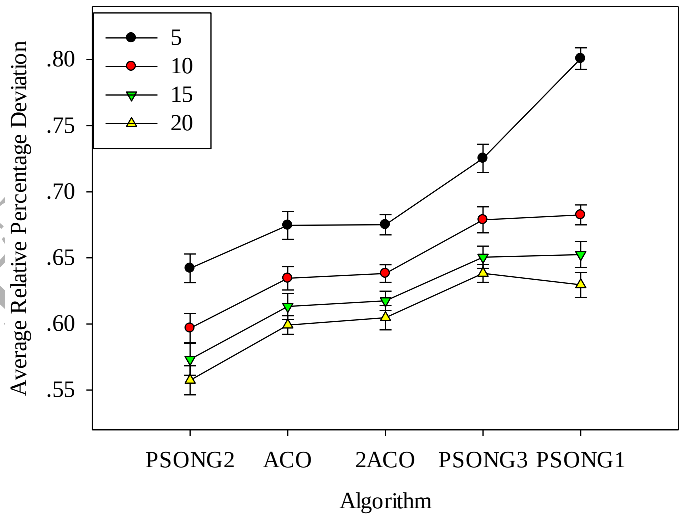

**Figure 5** Means plot and 95% Tukey HSD confidence intervals for the interactions five algorithms and elapsed CPU time

The test results of Friedman statistical test are not given here (provided as supplementary material instead). It is sufficient to say that the results of the Friedman test coincide with that of ANOVA and the results reported in Table 3. Notice that the above computational results are achieved utilizing the combination of SOE, ST5, FitII5 and iterative mechanism. The comparative results might be different when utilizing other combinations.

Since optimum cycle times for some large-sized problems are still in absence, Table 4 reports the new best solutions found during the experimental tests of this research. The current best cycle times and the best cycle times achieved by the nine most effective algorithms are exhibited for clarity. Notice that the current best cycle times are the minimum values of the reported ones in Tang, Li [6], Li, Tang [14] and Lei and Guo [15]. Table 4 also reports the lower bounds ($CT_{LB}$) to measure the gap between the newly achieved best solutions and the possible optimum cycle times. This table shows that many best solutions are updated by the re-implemented algorithms. Specifically, the best solutions for four cases of P65 ($Nm = 5,\ Nm = 6,\ Nm = 7$ and $Nm = 8$), three cases of P148 ($Nm = 7,\ Nm = 11$ and $Nm = 12$) and eight cases of P205 ($Nm = 4,\ Nm = 6,\ Nm = 7,\ Nm = 8,\ Nm = 9,\ Nm = 10,$ $Nm = 11$ and $Nm = 12$) are updated. Meanwhile, the optimum solutions for six cases are first achieved: three cases for P65 and three cases for P148. Note that the achieved solution is considered to be optimum when its cycle time is equal to the corresponding lower bound. This is because it is not possible to obtain a better solution than the lower bound. Regarding the number of achieved best solutions, it is observed that IG3 is the best performer which updates 12 cases, TS is the second best performer which updates eight cases, and ABC is the third best performer which updates seven cases. GA updates for only one case. If the achieved results are compared to the current best ones, IG3 outperforms the published ones for 12 cases, and TS and ABC outperform the published ones for 12 cases and 11 cases, respectively. These computational results demonstrate that proper selection of objective function, encoding scheme and decoding procedure enhances the performance of the algorithms, and, as a consequence, algorithms are capable of obtaining satisfying results.

**Table 4** Best cycle times found for the large-sized problems

| Problem | Nm | $CT_{LB}$ | Current best | SA | TS | LAHC | IG2 | IG3 | GA | BA | ABC | DABC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P65 | 4 | 638 | 638 | 638 | 638 | 638 | 638 | 638 | 638 | 638 | 638 | 638 |
| | 5 | 510 | 511 | 511 | 511 | 511 | **510** | **510** | 511 | 511 | 511 | 511 |
| | 6 | 425 | 426 | 426 | 426 | 426 | 426 | 426 | 426 | 426 | **425** | 426 |
| | 7 | 365 | 367 | **365** | **365** | 366 | 366 | **365** | 366 | **365** | **365** | 366 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 | 319 | 321 | **320** | **320** | 321 | **320** | **320** | **320** | 321 | **320** | 321 |
| P148 | 4 | 641 | 641 | 641 | 641 | 641 | 641 | 641 | 641 | 641 | 641 | 641 |
| | 5 | 513 | 513 | 513 | 513 | 513 | 513 | 513 | 513 | 513 | 513 | 513 |
| | 6 | 427 | 427 | 427 | 427 | 427 | 427 | 427 | 427 | 427 | 427 | 427 |
| | 7 | 366 | 367 | **366** | **366** | **366** | **366** | **366** | 367 | **366** | **366** | **366** |
| | 8 | 321 | 321 | 321 | 321 | 321 | 321 | 321 | 321 | 321 | 321 | 321 |
| | 9 | 285 | 285 | 285 | 285 | 285 | 285 | 285 | 285 | 285 | 285 | 285 |
| | 10 | 257 | 257 | 257 | 257 | 257 | 257 | 257 | 257 | 257 | 257 | 257 |
| | 11 | 233 | 234 | **233** | 234 | 234 | 234 | 234 | 234 | 234 | **233** | 234 |
| | 12 | 214 | 215 | **214** | **214** | **214** | **214** | **214** | 215 | **214** | **214** | **214** |
| P205 | 4 | 2919 | 2927 | 2927 | **2926** | **2926** | 2927 | 2927 | 2927 | 2930 | 2928 | **2926** |
| | 5 | 2335 | 2342 | 2343 | 2344 | 2345 | 2345 | 2343 | 2345 | 2347 | 2342 | 2342 |
| | 6 | 1946 | 1954 | **1952** | 1953 | **1952** | 1953 | **1952** | 1953 | 1956 | **1952** | **1952** |
| | 7 | 1668 | 1676 | 1677 | **1675** | 1676 | **1675** | **1675** | 1677 | 1679 | 1676 | 1677 |
| | 8 | 1460 | 1469 | 1468 | 1468 | 1468 | 1469 | **1467** | 1468 | 1475 | 1469 | **1467** |
| | 9 | 1297 | 1309 | 1306 | 1305 | 1306 | 1306 | **1304** | 1308 | 1309 | 1306 | 1305 |
| | 10 | 1168 | 1180 | 1178 | 1178 | 1176 | 1180 | **1175** | 1185 | 1184 | 1176 | 1182 |
| | 11 | 1062 | 1074 | 1071 | **1070** | **1070** | 1071 | **1070** | 1073 | 1074 | 1072 | 1071 |
| | 12 | 973 | 984 | 984 | **982** | **982** | 983 | **982** | 984 | 986 | 983 | 983 |
| | 13 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 |
| | 14 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 | 944 |

* New best solutions are given in bold.

## 6. Conclusions and future research directions

This research provides an extensive review and evaluation of heuristics and meta-heuristics applied in TALBP. Meanwhile, six encoding schemes and 30 decoding procedures are summarized, where part of them are first developed. This research also points out the search space for TALBP-I, which is $2^{Ne} \cdot Nt!$ possible solutions rather than $Nt!$ possible solutions considered for the simple ALBP, where $Nt$ is the number of tasks and $Ne$ is the number of E-type tasks. A total of eighteen algorithms is carefully tested and evaluated, which includes the majority of the applied methods for TALBP.

Several conclusions can be drawn based on the comprehensive experiments on seven sets of benchmark problems and statistical analysis of their results. Firstly, the encoding and decoding procedures have a great effect on the performances of the algorithms, and the combination of sequence-oriented encoding and task assignment procedure ST5 is the good performer. Secondly, the objective function considered is another important factor which influences the algorithm performance by a significant margin. Thirdly, the iterative mechanism is quite effective, which makes sure that the incumbent and the new individuals are compared using the same initial cycle time. It also helps preserve the small improvements on individuals. Lastly, the iterated greedy search algorithms, bee algorithms, simulated annealing algorithm, tabu search algorithm, and late acceptance hill-climbing algorithm are efficient for TALBP-II utilizing the employed combination of the encoding scheme, decoding procedure and objective function. Furthermore, these findings also demonstrate the possible comparison unfairness published where algorithms utilize different encoding schemes, decoding procedures or objective functions. Another unique contribution of this paper is that new best solutions for 15 large-sized TALBP-II instances (of which six are optimum) are obtained for the first time in the literature.

The future research might apply these findings to other two-sided assembly line balancing problems. For instance, the mixed-model TALBP, TALBP with multiple constraints and parallel TALBP might be further studied with the application of these findings. Specifically, the summarized encoding schemes and decoding procedures contain the majority of the published ones, and they are applicable to other two-sided assembly line balancing problems directly or with a simple modification. The summarized objective function shows that effective function leads to the high performance of the tested algorithms, and proposing these summarized objective function for other two-sided assembly line balancing problems might further enhance the performance of the algorithms. The procedures tested in this paper can be employed by the developers of new heuristics/meta-heuristics and the new best solutions reported in this research can be a reference for the comparison of algorithms to be developed in future researches.

Accelerating the search speed developing new encoding/decoding procedures and objective functions might be another challenging topic. In TALBP, the more allocated workload to former mated-station always leads to better solutions. Hence, blending local search into the decoding procedure to ensure more workload to former mated-station might achieve better solutions and speed up the search speed.

The new objective might take the operation times of tasks and the precedence relationship into account to allocate the tasks with larger operation times and more successors to former mated-station.

Since multi-objective optimization algorithms are not evaluated in this study, it is suggested to evaluate the published multi-objective optimization algorithms in the future. In the evaluation of the multi-objective algorithm, these findings achieved by solving one-objective TALBP might also benefit multi-objective TALBP, such as the encoding schemes and decoding procedures. In addition, most published multi-objective algorithms consider two or three objectives, whereas there might be more than three objectives in real applications. Hence, future research might also study the multi-objective algorithm with many objectives.

Another research avenue is studying the TALBP with realistic considerations observed in real applications. For example, incompatible task groups constraints have been used by Zhang, Kucukkoc [61] to handle incompatible tasks which cannot be performed at the same time. In common practice, negative zoning constraints are used to represent tasks which cannot be assigned to the same workstation. However, in industry, there may be some tasks which can be assigned to the same workstation (and so performed subsequently), but not in the same mated-station (cannot be performed concurrently one on each side of the line). Let us assume a two-sided line utilized for the assembly of a safety cabin, which needs some tasks to be performed by operators inside the product. If the space inside the product is limited, two operators may not get inside it, which prevents the performance of two inside tasks at the same time. Also, it is widely observed in real-world applications that operators travel between the workstations even in a straight line [62]. This issue has naturally been studied and the walking times of operators have been considered for those working in either adjacent lines of a parallel line system or front/back branches of a U-shaped line. However, traveling workers have not been taken into account for a two-sided assembly line, to the best of the authors’ knowledge. Due to the labor-intensive structure of the final assembly systems, ergonomics issues also need to be investigated more in two-sided lines.

Sustainability and energy efficiency are two trending topics in recent studies on industrial engineering and operations research problems. However, the research on these are strictly limited in the line balancing domain (see [18, 63]). As the flexibility crucially determines the survivability of enterprises in today's highly consumer-centric global market, it needs to be further investigated in two-sided lines. Consideration of express lines [64] (each of which employs different throughput rate or even specialized operators) and utilization of reconfigurable assembly lines (see [65, 66]) adaptable to radical changes in the assembly strategy can also be further investigated in a flexible assembly concept. The promising development of cloud manufacturing also desires its involvement in the line balancing problems [67].

## Appendices

**Table A1** Parameter values for the re-implemented methods after calibration

| Algorithm | Parameters | Value |
|---|---|---|
| SA | Initial temperature | 1.0 |
| | Ratio of temperature decreasing | 0.95 |
| | Iteration rate | 100 |
| | Neighborhood structure type | Swap operation and insert operation |
| TS | Tabu length | 100 |
| | Neighborhood structure type | Swap operation and insert operation |
| LAHC | List length | 10 |
| | Neighborhood structure type | Swap operation and insert operation |
| IG1 | Temperature | 0.001 |
| | Destruction size (d) | 4 |
| IG2 | Temperature | 0.001 |
| | Destruction size (d) | 4 |
| | a | 40 |
| | b | 3 |
| IG3 | Temperature | 0.001 |
| | Destruction size (d) | 4 |
| | a | 40 |
| | b | 3 |
| GA | Population size | 120 |
| | Selection type | Binary tournament selection |

| | | |
|---|---|---|
| | Crossover type | Two-point crossover |
| | Crossover probability | 0.4 |
| | Mutation type | Swap operation and insert operation |
| | Mutation probability | 1-Crossover probability |
| TLBO | Population size | 160 |
| BA | Population size | 80 |
| | Number of employed bees | 40 |
| | Number of best employed bees | 2 |
| | Number of onlookers for each best employed bee | 10 |
| | Number of onlookers for each employed bee except for the best employed bee | 1 |
| ABC | Population size | 40 |
| | Neighborhood operator | Swap operation and insert operation |
| DABC | Population size | 120 |
| | Neighborhood operator for employed bees | Two-point crossover |
| | Neighborhood operator for onlookers | Swap operation and insert operation |
| | Scout phase | Sending a scout when there is no improvement within an iteration to replace the worst individual in the population |
| PSO1 | Number of swarms | 8 |
| | Number of particles in a swarm | 60 |
| | Global learning factor (c1) | 1.0 |
| | Local learning factor (c2) | 1.0 |
| | Initial weight (w) | 1.0 |
| PSO2 | Number of swarms | 8 |
| | Number of particles in a swarm | 60 |
| | c | 0.5 |
| ACO | Population size | 120 |
| | Selection probability of task j | $p(j)=\frac{[\tau_{ij}]^{\alpha}[\eta_j]^{\beta}}{\sum_{j\in A_i}[\tau_{ij}]^{\alpha}[\eta_j]^{\beta}}$ , $\eta_j$ is the ranked positional weight heuristic, $\alpha$ and $\beta$ are parameters |
| | Parameters: $\alpha$ and $\beta$ | (0.2, 1.0) |
| | Initial pheromone trails | 1.0 |
| | Evaporation coefficient $\rho$ | 0.1 |
| 2ACO | Population size | 120 |

| | | |
|---|---|---|
| | Selection probability of task j | $p(j) = \frac{[\tau_{ij}]^{\alpha}[\eta_j]^{\beta}}{\sum_{j \in A_i}[\tau_{ij}]^{\alpha}[\eta_j]^{\beta}}$ , $\eta_j$ is the ranked positional weight heuristic, $\alpha$ and $\beta$ are parameters |
| | Parameters: $\alpha$ and $\beta$ | (0.2, 1.0) |
| | Initial pheromone trails | 1.0 |
| | Evaporation coefficient $\rho$ | 0.1 |
| PSONG1/ PSONG2/ PSONG3 | Number of swarms | 4 |
| | Number of particles in a swarm | 40 |
| | Parameters: $\alpha$ and $\beta$ | (0.2, 1.0) |
| | Global learning factor (c1) | 0.2 |
| | Local learning factor (c2) | 0.1 |

## References

[1] Scholl A, Becker C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research. 2006;168(3):666-93.

[2] Lee TO, Kim Y, Kim YK. Two-sided assembly line balancing to maximize work relatedness and slackness. Computers & Industrial Engineering. 2001;40(3):273-92.

[3] Li Z, Tang Q, Zhang L. Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study. Computers & Operations Research. 2017;79:78-93.

[4] Kellegöz T, Toklu B. An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations. Computers & Operations Research. 2012;39(12):3344-60.

[5] Purnomo HD, Wee H-M, Rau H. Two-sided assembly lines balancing with assignment restrictions. Mathematical and Computer Modelling. 2013;57(1–2):189-99.

[6] Tang Q, Li Z, Zhang L. 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. 2016;97:146-56.

[7] Purnomo HD, Wee H-M. Maximizing production rate and workload balancing in a two-sided assembly line using Harmony Search. Computers & Industrial Engineering. 2014;76:222-30.

[8] Kucukkoc I, Zhang DZ. Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. Computers and Industrial Engineering. 2015;84:56–69, doi: http://dx.doi.org/10.1016/j.cie.2014.12.037.

[9] Yuan B, Zhang C, Shao X. A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. Journal of Intelligent Manufacturing. 2015;26(1):159-68.

[10] Özbakır L, Tapkan P. Bee colony intelligence in zone constrained two-sided assembly line balancing problem. Expert Systems with Applications. 2011;38(9):11947-57.

[11] Özcan U, Toklu B. A tabu search algorithm for two-sided assembly line balancing. The International Journal of Advanced Manufacturing Technology. 2008;43(7):822-9.

[12] Özcan U, Gökçen H, Toklu B. Balancing parallel two-sided assembly lines. International Journal of Production Research. 2010;48(16):4767-84.

[13] Kim YK, Song WS, Kim JH. A mathematical model and a genetic algorithm for two-sided assembly line balancing. Computers & Operations Research. 2009;36(3):853-65.

[14] Li Z, Tang Q, Zhang L. Minimizing the Cycle Time in Two-Sided Assembly Lines with Assignment Restrictions: Improvements and a Simple Algorithm. Mathematical Problems in Engineering. 2016;2016(Article ID 4536426):1-15.

[15] Lei D, Guo X. Variable neighborhood search for the second type of two-sided assembly line balancing problem. Computers & Operations Research. 2016;72:183-8.

[16] Aghajani M, Ghodsi R, Javadi B. Balancing of robotic mixed-model two-sided assembly line with robot setup times. The International Journal of Advanced Manufacturing Technology. 2014;74(5-8):1005-16.

[17] Kim YK, Kim Y, Kim YJ. Two-sided assembly line balancing: A genetic algorithm approach. Production Planning & Control. 2000;11(1):44-53.

[18] Li Z, Tang Q, Zhang L. Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm. Journal of Cleaner Production. 2016;135:508-22.

[19] Wu E-F, Jin Y, Bao J-S, Hu X-F. A branch-and-bound algorithm for two-sided assembly line balancing. The International Journal of Advanced Manufacturing Technology. 2008;39(9):1009-15.

[20] Hu X, Wu E, Jin Y. A station-oriented enumerative algorithm for two-sided assembly line balancing. European Journal of Operational Research. 2008;186(1):435-40.

[21] Xiaofeng H, Erfei W, Jinsong B, Ye J. A branch-and-bound algorithm to minimize the line length of a two-sided assembly line. European Journal of Operational Research. 2010;206(3):703-7.

[22] Bartholdi JJ. Balancing two-sided assembly lines: a case study. International Journal of Production Research. 1993;31(10):2447-61.

[23] Özcan U, Toklu B. Balancing two-sided assembly lines with sequence-dependent setup times. International Journal of Production Research. 2010;48(18):5363-83.

[24] Taha RB, El-Kharbotly AK, Sadek YM, Afia NH. A Genetic Algorithm for solving two-sided assembly line balancing problems. Ain Shams Engineering Journal. 2011;2(3–4):227-40.

[25] Kucukkoc I, Zhang DZ. A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem. Production Planning & Control. 2015;26(11):874-94.

[26] Rabbani M, Moghaddam M, Manavizadeh N. Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout. International Journal of Advanced Manufacturing Technology. 2012;59(9-12):1191-210.

[27] Delice Y, Kızılkaya Aydoğan E, Özcan U. Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach. International Journal of Production Research. 2016;54(11):3429-51.

[28] Baykasoglu A, Dereli T. Two-sided assembly line balancing using an ant-colony-based heuristic. The International Journal of Advanced Manufacturing Technology. 2008;36(5):582-8.

[29] Simaria AS, Vilarinho PM. 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. Computers & Industrial Engineering. 2009;56(2):489-506.

[30] Kucukkoc I, Zhang DZ. Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Research. 2014;52(12):3665-87.

[31] Kucukkoc I, Zhang DZ. Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. International Journal of Production Economics. 2014;158:314-33.

[32] Kucukkoc I, Zhang DZ. Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines. The International Journal of Advanced Manufacturing Technology. 2016;82(1):265-85.

[33] Kucukkoc I, Zhang DZ. Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters. Computers & Industrial Engineering. 2015;84:56-69.

[34] Kucukkoc I, Zhang DZ. Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach. Computers & Industrial Engineering. 2016;97:58-72.

[35] Özcan U, Toklu B. Balancing of mixed-model two-sided assembly lines. Computers & Industrial Engineering. 2009;57(1):217-27.

[36] Yuan B, Zhang C, Shao X, Jiang Z. An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. Computers & Operations Research. 2015;53:32-41.

[37] Özcan U. Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. European Journal of Operational Research. 2010;205(1):81-97.

[38] Roshani A, Fattahi P, Roshani A, Salehi M, Roshani A. Cost-oriented two-sided assembly line balancing problem: A simulated annealing approach. International Journal of Computer Integrated Manufacturing. 2012;25(8):689-715.

[39] Khorasanian D, Hejazi SR, Moslehi G. Two-sided assembly line balancing considering the relationships between tasks. Computers & Industrial Engineering. 2013;66(4):1096-105.

[40] Jawahar N, Ponnambalam SG, Sivakumar K, Thangadurai V. Heuristics for Multiobjective Optimization of Two-Sided Assembly Line Systems. The Scientific World Journal. 2014;2014:1-16.

[41] Özbakır L, Tapkan P. Balancing fuzzy multi-objective two-sided assembly lines via Bees Algorithm. Journal of Intelligent & Fuzzy Systems. 2010;21(5):317-29.

[42] Tapkan P, Ozbakir L, Baykasoglu A. Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms. Applied Soft Computing. 2012;12(11):3343-55.

[43] Tapkan P, Özbakır L, Baykasoğlu A. Bees Algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem. Optimization Letters. 2012;6(6):1039-49.

[44] Tapkan P, Özbakır L, Baykasoğlu A. Bee algorithms for parallel two-sided assembly line balancing problem with walking times. Applied Soft Computing. 2016;39:275-91.

[45] Chutima P, Chimklai P. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. Computers & Industrial Engineering. 2012;62(1):39-55.

[46] Delice Y, Kızılkaya Aydoğan E, Özcan U, İlkay MS. A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. Journal of Intelligent Manufacturing. 2014:1-14.

[47] Delice Y, Aydoğan EK, Özcan U, İlkay MS. Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm. 4OR. 2016:1-30.

[48] Chiang W-C, Urban TL, Luo C. Balancing stochastic two-sided assembly lines. International Journal of Production Research. 2015:1-19.

[49] Li Z, Janardhanan MN, Tang Q, Nielsen P. Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. Advances in Mechanical Engineering. 2016;8(9):14.

[50] Tuncel G, Aydin D. Two-sided assembly line balancing using teaching–learning based optimization algorithm. Computers & Industrial Engineering. 2014;74:291-9.

[51] Tang QH, Li ZX, Zhang LP, Floudas CA, Cao XJ. Effective hybrid teaching-learning-based optimization algorithm for balancing two-sided assembly lines with multiple constraints. Chinese Journal of Mechanical Engineering. 2015;28(5):1067-79.

[52] Li D, Zhang C, Shao X, Lin W. A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. Journal of Intelligent Manufacturing. 2016;27(4):725-39.

[53] Tang Q, Li Z, Zhang L, Zhang C. Balancing stochastic two-sided assembly line with multiple constraints using hybrid teaching-learning-based optimization algorithm. Computers & Operations Research. 2017;82:102-13.

[54] Sepahi A, Naini SGJ. Two-sided assembly line balancing problem with parallel performance capacity. Applied Mathematical Modelling. 2016;40(13-14):6280-92.

[55] Wang B, Guan Z, Li D, Zhang C, Chen L. Two-sided assembly line balancing with operator number and task constraints: a hybrid imperialist competitive algorithm. The International Journal of Advanced Manufacturing Technology. 2014;74(5):791-805.

[56] Li D, Zhang C, Tian G, Shao X, Li Z. Multiobjective Program and Hybrid Imperialist Competitive Algorithm for the Mixed-Model Two-Sided Assembly Lines Subject to Multiple Constraints. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2016;PP(99):1-11.

[57] Yang Z, Zhang G, Zhu H. Multi-neighborhood based path relinking for two-sided assembly line balancing problem. Journal of Combinatorial Optimization. 2016;32(2):396-415.

[58] Li Z, Dey N, Ashour AS, Tang Q. Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem. Neural Computing and Applications. 2017.

[59] Abdullah Make MR, Ab. Rashid MFF, Razali MM. A review of two-sided assembly line balancing problem. The International Journal of Advanced Manufacturing Technology. 2016.

[60] Hamta N, Fatemi Ghomi SMT, Jolai F, Akbarpour Shirazi M. A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. International Journal of Production Economics. 2013;141(1):99-111.

[61] Zhang DZ, Kucukkoc I, Karaoglan AD. Rebalancing of mixed-model two-sided assembly lines with incompatible task groups: An industrial case study. 46th International Conference on Computers & Industrial Engineering (CIE46), 29-31 October 2016, Tianjin, China2016.

[62] Sikora CGS, Lopes TC, Magatão L. Traveling worker assembly line (re)balancing problem: Model, reduction techniques, and real case studies. European Journal of Operational Research. 2016.

[63] Urban TL, Chiang W-C. Designing energy-efficient serial production lines: The unpaced synchronous line-balancing problem. European Journal of Operational Research. 2016;248(3):789-801.

[64] Rabbani M, Ziaeifar A, Manavizadeh N. Mixed-model assembly line balancing in assemble-to-order environment with considering express parallel line: problem definition and solution procedure. International Journal of Computer Integrated Manufacturing. 2013;27(7):690-706.

[65] Yuan M, Xu H. Reconfigurable assembly line balancing with the hybrid genetic algorithm. 2010:1398-401.

[66] Colledani M, Gyulai D, Monostori L, Urgo M, Unglert J, Van Houten F. Design and management of reconfigurable assembly lines in the automotive industry. CIRP Annals - Manufacturing Technology. 2016;65(1):441-6.

[67] Yuan M, Deng K, Chaovalitwongse WA, Cheng S. Multi-objective optimal scheduling of reconfigurable assembly line for cloud manufacturing. Optimization Methods and Software. 2016:1-13.