Soft Computing (2022) 26:2241–2258
https://doi.org/10.1007/s00500-021-06472-z

OPTIMIZATION

# Hybrid meta-heuristic algorithms for U-shaped assembly line balancing problem with equipment and worker allocations

Morteza Khorram[1] · Mahmood Eghtesadifard[1] · Sadegh Niroomand[2]

Accepted: 20 October 2021 / Published online: 10 November 2021

**Abstract**
In this paper, a new U-shaped assembly line balancing problem is studied. For the first time, the criteria such as equipment cost, number of stations and activity performing quality level are considered to be optimized simultaneously by activity to station and worker to station decisions. For this aim, a multi-objective nonlinear formulation is proposed and its linearized version is also presented. Since, according to the literature, the U-shaped assembly line balancing problem with equipment requirements is an NP-hard problem, the problem of this study is NP-hard too. Because of this complexity, the classical algorithms like simulated annealing, variable neighborhood search, and classical genetic algorithm with a novel encoding/decoding scheme are used as solution approaches. As an extension, two hybrid versions of the proposed classical algorithms are proposed according to the characteristics of the problem. In order to evaluate the proposed meta-heuristics, because the problem is new, some test problems are generated randomly. Computational study of the paper, including sensitivity analysis of the proposed meta-heuristics and final experiments on the test problems, proves the superiority of the hybrid versions of the classical algorithms.

**Keywords** U-shaped assembly line balancing · Meta-heuristic algorithm · Worker assignment · NP-hard problem · Equipment assignment

## 1 Introduction

Assembly lines are one of the most widely used traditional manufacturing systems for mass and high volume production, which help to increase the efficiency and speed of production systems as well as reduce the production costs and timely respond to fluctuations in demands of competitive markets. These lines were first introduced by Henry Ford at Ford automobile factories, and since that time various researches have been done to improve the performance of the assembly lines (Yuan et al. 2015; Saif et al. 2017; Make et al. 2017; Zhou and Tan 2018; Pattanaik and Jena 2019). An assembly line consists of a set of several workstations located along with a transportation system with a flow of materials between them. Each of the stations performs a set of activities. Materials and product parts flow continuously on the line by the transportation system, and after the operations of a station are performed, they are transferred to the next station until they reach the end of the line. Each station performs a set of activities from the required activities for completing the product. Regarding the production cycle time, a number of activities are assigned to each workstation. According to the assembly line balancing problem, the assignment should be done in a way that one or more objectives is optimized subject to the cycle time of the line and precedence relationships of the activities. The objectives could be minimizing the number of workstations, minimizing cycle time, minimizing total costs, etc. (see Scholl and Klein (1999), Ozcan and Toklu (2009) for the number of workstation minimization, Ozcan and Toklu (2009) Foroughi et al. (2016) for cycle time minimization, and Foroughi and Gökçen (2018), Yavari et al. (2019), Li et al. (2019) for cost minimization).

Due to the type and characteristics of the products, the layout of the line, as well as the technical and operational

✉ Sadegh Niroomand
niroomand@fabad-ihe.ac.ir

[1] Department of Industrial Engineering, Shiraz University of Technology, Shiraz, Iran

[2] Department of Industrial Engineering, Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran

requirements, different classifications are presented by the researchers for assembly line balancing problems. The most well-known classifications provided by Baybars (1986) have divided assembly line balancing problems into two categories. The first group is called the simple assembly line balancing problem, and the second one is called the general assembly line balancing problems. The simple assembly balancing problem considers some simple assumptions so that the balance problem can easily be solved. Assembly line balancing problems that remove one or more of the assumptions of simple assembly balancing problems are in the category of general assembly line balancing problem. Also, in a different classification, assembly lines can be divided based on the layout of stations. The assembly lines can be divided into four types according to the layout (Hadi-Vencheh and Mohamadghasemi, 2013) of the stations such as straight assembly lines, U-shaped assembly lines, parallel assembly lines, and two-sided assembly lines (Saif et al. 2014). In straight assembly lines, all stations are placed on a direct path and the pieces or materials pass through the stations from the beginning to the end of the line. An important condition is that an activity can be assigned to a station if all of its predecessors are assigned to that station or earlier stations. U-shaped assembly lines generalize straight assembly lines in which the stations are located in such a way that entry and exit of the materials are carried out from one position and operations of two or more stations can be performed by one operator (Saif et al. 2014; Delice et al. 2017; Aigbedo and Monden, 1997). An important condition is that an activity can be assigned to a station if either all of its predecessors or all of its successors are assigned to that station or earlier stations (see Fig. 1). In these lines, the time of worker's unemployment is reduced. Moreover, the number of stations required for U-shaped lines is never greater than the number of stations required for straight assembly lines (Baykasoglu 2006; Ajenblit and Wainwright 1998).

In recent years, many studies have been done on assembly line balancing problems, and various algorithms have been used to solve them. In the assembly line balancing problems, depending on the type of line and the type of problem, different objectives can be considered. For example, studies of Chica et al. (2011) and Hamta et al. (2013) have been done to reduce the cycle time as a variable. The studies of Ponnambalam et al. (2000) and Nourmohammadi and Zandieh (2011) have met the goals of minimizing the number of workstations and the smoothness index in assembly line balancing problems. Also, various meta-heuristic algorithms have been used to solve these problems. The genetic algorithm was used by Miltenburg (2002), Rabbani et al. (2012), Alavidoost et al. (2017), and Zhang and Gen (2011) to solve a variety of the assembly line balancing problems. The U-shaped assembly line was first studied by Miltenburg and Wijngaard (1994). Subsequently, many studies have been done to develop the formulation and solution approaches to this problem. Kim, et al. (2000), for the first time, considered the U-shaped assembly line balancing problem with the sequencing problem simultaneously. Kara et al. (2009) proposed a fuzzy goal programming and fuzzy binary programming model for this problem. Hamta et al. (2014) provided an integer nonlinear mathematical model for this problem. On the other hand, some researches were carried out by Agpak

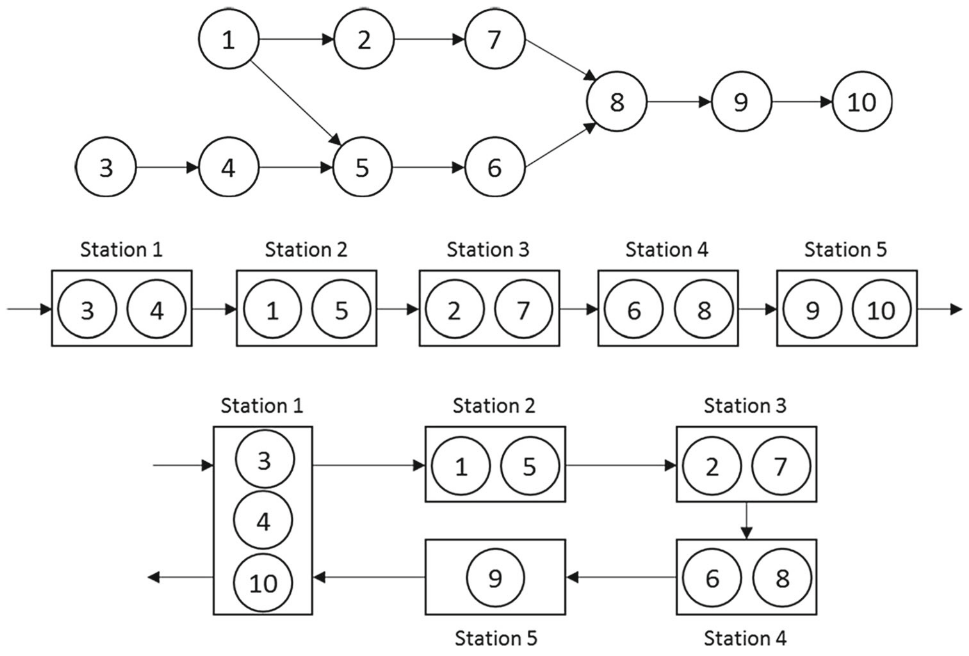

**Fig. 1** A precedence graph, its simple balancing, and its U-shaped balancing (Source: Khorram 2018)

et al. (2012) and Ogan and Azizoglu (2015) to improve and develop the mathematical model of the U-shaped assembly line balancing problem. Alavidoost et al., (2017) solved straight and U-shaped assembly line balancing problems using an improved genetic algorithm with consideration of fuzzy values for activity performing times. Also, Ogan and Azizoglu (2015) proposed a mathematical model for the U-shaped assembly line balancing problem by considering activities required equipment. Then they used a typical branch and bound algorithm to solve this problem. Li et al. (2017) presented a new mixed-integer linear programming model with the goal of minimizing the number of workstations for balancing the U-shaped assembly lines. In this model, one constraint is used to represent the precedence relationship constraint instead of two constraints in the studies of the literature. Li et al. (2018) proposed a branch, bound and remember algorithm for the U-shaped assembly line balancing problem. Li et al. (2021) proposed an enhanced beam search heuristic for U-shaped assembly line balancing problems.

There are a few researches employing the simulated annealing algorithm to solve the assembly line balancing problems. One is the study of Suresh and Sahu (1994) which used the simulated annealing algorithm to solve the assembly line balancing problem. Also, McMullen and Frazier (1998) used this algorithm to solve the multi-objective assembly line balancing problem with parallel workstations, random task times, and multiple products. In addition, Roshani et al. (2012) used the simulated annealing algorithm to solve a two-sided assembly line balancing problem. Baykasoglu (2006) used this algorithm to solve U-shaped assembly line balancing problem with the objectives of maximizing line performance and line smoothness index. Erel et al. (2001) used this algorithm to solve a typical U-shaped assembly line balancing problem. In addition, Manavizadeh et al. (2013) investigated the U-shaped assembly line balancing problem with the aim of minimizing the number of workstations through balancing workload and maximizing the weighted efficiency with the simulated annealing algorithm.

In this study, an approach consisting of mathematical formulation and meta-heuristic algorithms is developed to solve a new U-shaped assembly balancing problem. Compared to the literature, a new nonlinear multi-objective formulation (Hadi-Vencheh 2011; Kovács et al. 2017) for the U-shaped assembly line balancing problem is presented and linearized. This problem for the first time considers the criteria such as equipment purchasing cost, number of stations, and activity performing quality simultaneously and optimizes them by activity and worker allocations. Since the U-shaped assembly line balancing problem with equipment requirements is an NP-hard problem (Ogan and Azizoglu, 2015), the use of meta-heuristic algorithms for the problem of this study is inevitable. As a solution approach, the classical simulated annealing and classical variable neighborhood search algorithms with a novel encoding/decoding scheme are used and hybridized according to the characteristics of the problem. In order to evaluate the proposed meta-heuristics, because the problem is new, some test problems are generated randomly. Computational study of the paper including sensitivity analysis of the proposed meta-heuristics and final comparison of them proves the effectiveness of the hybrid versions of the classical simulated annealing for the problem of this study.

The remainder of this paper has four sections. Section 2 presents the problem description and its formulation. Section 3 develops the meta-heuristic solution approaches. Section 4 includes the computational study of the problem and the solution approaches. The paper ends with the concluding remarks of Sect. 5.

## 2 Problem description and formulation

The problem of this study is a typical U-shaped assembly line balancing problem. This problem has some properties and assumptions which are explained as follow.

- A set of activities with their given precedence relationships are to be assigned to a set of given stations in order to produce a single product.
- A set of workers is available where each of them is able to perform all of the activities at a known quality level.
- Cycle time of the line is known.
- There is a set of equipment where one or some of them is used to perform each activity.
- For each established station, the set of required equipment and a worker should be allocated.
- For each station, after allocating its activities and workers, the lowest quality among its activities determines the quality level of that station. Accordingly, the dis-quality level of each station is one minus its quality level.
- The following objectives are considered simultaneously,
  - Minimization of the equipment purchasing cost.
  - Minimization of the number of established stations.
  - Minimization of the average of the dis-quality level of the line.

For this problem, the following notations are considered,

**Sets** $PR_i$ The set of predecessors for activity

$SC_i$ The set of successors for activity $i$.

$TL_i$ The set of equipment used for activity $i$.

**Indexes** $i, p$ Indexes used for activities $i, p \in \{1, 2, ..., I\}$.

$k, r$ Indexes used for stations $k, r \in \{1, 2, ..., K\}$.

$u$ Index used for workers $u \in \{1, 2, ..., U\}$.

$l$ Index used for equipment $l \in \{1, 2, ..., L\}$.

**Parameters** $t_i$ The time to perform activity $i$.

$ec_l$ The purchasing cost of equipment $l$.

$q_{iu}$ The quality level of worker $u$ for performing activity $i$.

$ct$ Cycle time.

**Variables** $U_i$ A binary variable. It takes value of 1 if activity $i$ is assigned to the front leg of the U-shaped line, it takes value of zero if activity $i$ is assigned to the behind leg (the front leg is the first leg passing by the product).

$X_{ik}$ A binary variable. It takes value of 1 if activity $i$ is assigned to station $k$, otherwise, its value is zero.

$S_k$ A binary variable. It takes value of 1 if station $k$ includes at least one activity, otherwise, its value is zero.

$Z_{lk}$ A binary variable. It takes value of 1 if equipment $l$ is assigned to station $k$, otherwise, its value is zero.

$W_{uk}$ A binary variable. It takes value of 1 if worker $u$ is assigned to station $k$, otherwise, its value is zero.

$Q_k$ The quality level of station $k$.

According to the above-mentioned notations and assumptions, the following mathematical formulation is proposed to optimize the proposed U-shaped assembly line balancing problem.

Objective function 1

$$OF_1 = \min \sum_{l=1}^{L} \sum_{k=1}^{K} ec_l Z_{lk} \tag{1}$$

Objective function 2

$$OF_2 = \min \sum_{k=1}^{K} S_k \tag{2}$$

Objective function 3

$$OF_3 = \min \frac{\sum_{k=1}^{K} 1 - Q_k}{\sum_{k=1}^{K} S_k} \tag{3}$$

subject to

$$\sum_{k=1}^{K} kX_{pk} - \sum_{k=1}^{K} kX_{ik} \le K\left(1 + U_p - 2U_i\right),\ \forall (p,i) | p \in PR_i \tag{4}$$

$$\sum_{k=1}^{K} kX_{sk} - \sum_{k=1}^{K} kX_{ik} \le KU_i,\ \forall (i,s) | s \in SC_i \tag{5}$$

$$\sum_{k=1}^{K} X_{ik} = 1, \forall i \tag{6}$$

$$S_k \ge X_{ik},\ \forall i, k \tag{7}$$

$$S_k \le \sum_{i=1}^{I} X_{ik}, \forall k \tag{8}$$

$$\sum_{i=1}^{I} t_i X_{ik} \le ct,\ \forall k \tag{9}$$

$$\sum_{u=1}^{K} W_{uk} = S_k, \forall k \tag{10}$$

$$\sum_{k=1}^{K} W_{uk} \le 1,\ \forall u \tag{11}$$

$$(1 - q_{iu}) W_{uk} X_{ik} \le 1 - Q_k,\ \forall i, k, u \tag{12}$$

$$X_{ik} \le \frac{\sum_{l \in Tl_i} Z_{lk}}{|Tl_i|}, \forall i, k \tag{13}$$

$$X_{ik}, Z_{lk}, S_k, W_{uk} \in \{0, 1\}, \forall i, k, l, u \tag{14}$$

$$Q_k \ge 0, \forall k \tag{15}$$

Objective function (1) minimizes total equipment purchasing cost for all stations. Objective function (2) minimizes the number of established stations. Objective function (3) minimizes average dis-quality level of the established stations. Constraints (4)-(5) respect the precedence relationships of the activities. These constraints also respect to the concept of U-shaped assembly line, which says an activity can be assigned to a station if either all of its predecessors or all of its successors have been assigned to earlier stations. Constraint (6) assigns each activity to only one station. Constraints (7)-(8) consider a station as an established station if it contains at least one activity. Constraint (9) respects to the cycle time for each station. Constraints (10)-(11) assign workers to established stations. Constraint (12) determines the dis-quality level of each station. Constraint (13) determines the required equipment for the activities of each station, where, $|Tl_i|$ is the cardinality of set $Tl_i$. Finally, constraints (14)-(15) are sign constraints.

The proposed formulation (1)-(15) is a nonlinear formulation because of objective function (3) and constraint (12). In the rest of this section, this formulation is linearized. For this aim, some steps are followed. These steps are explained in the rest of this section.

*Step 1*

Linearize constraint (12) by the following conversion, where, $Y_{iuk}$ is a binary variable.

$$(1-q_{iu})W_{uk}X_{ik} \le 1-Q_k \Leftrightarrow \begin{cases} (1-q_{iu})Y_{iuk} \le 1-Q_k \\ W_{uk}+X_{ik} \ge 2Y_{iuk}\ ,\ \forall i,k,u \\ W_{uk}+X_{ik} \le 1+Y_{iuk} \end{cases} \tag{16}$$

*Step 2*

Use a typical Charnes and Cooper (1962) conversion method for objective function (3) as follow, where $T$ is a non-negative variable.

$$OF_3 = \min \frac{\sum_{k=1}^{K} 1-Q_k}{\sum_{k=1}^{K} S_k} \Leftrightarrow \begin{cases} OF_3 = \min \sum_{k=1}^{K} T - T(Q_k) \\ \textit{subject to} \\ \sum_{k=1}^{K} S_k = \frac{1}{T} \Rightarrow \sum_{k=1}^{K} T(S_k) = 1 \end{cases} \tag{17}$$

*Step 3* In the conversion (17), the nonlinear term $\sum_{k=1}^{K} T(S_k) = 1$ is linearized as follow, where $V_k = T(S_k)$ is a non-negative variable.

$$\sum_{k=1}^{K} T(S_k) = 1 \Leftrightarrow \begin{cases} \sum_{k=1}^{K} V_k = 1, \forall k \\ V_k \le S_k \\ V_k \le T \\ V_k \ge T-(1-S_k) \end{cases} \tag{18}$$

*Step 4* In the conversion (17), the nonlinear term $T(Q_k)$ is linearized as follow, where $R_k = T(Q_k)$ is a non-negative variable. In this conversion, the inequality $(1-q_{iu})Y_{iuk} \le 1-Q_k$ obtained in Step 1 also is influenced.

$$OF_3 = \min \sum_{k=1}^{K} T - T(Q_k) \Leftrightarrow OF_3 = \min \sum_{k=1}^{K} T - R_k \tag{19}$$

$$(1-q_{iu})Y_{iuk} \le 1-Q_k \Leftrightarrow (1-q_{iu})Y_{iuk}T \le T - R_k,\ \forall i,k,u \tag{20}$$

*Step 5* The nonlinear term $Y_{iuk}T$ of inequality (20) is replaced by a non-negative variable $B_{iuk} = Y_{iuk}T$, and the inequality is linearized as follow,

$$(1-q_{iu})Y_{iuk}T \le T-R_k \Leftrightarrow \begin{cases} (1-q_{iu})B_{iuk} \le T-R_k, \forall i,k,u \\ B_{iuk} \le Y_{iuk} \\ B_{iuk} \le T \\ B_{iuk} \ge T-(1-Y_{iuk}) \end{cases} \tag{21}$$

According to the obtained linearized constraints and objective functions of the above steps, the nonlinear formulation (1)-(15) is equivalent to the below linear formulation.

Objective function 1

$$OF_1 = \min \sum_{l=1}^{L} \sum_{k=1}^{K} ec_l Z_{lk} \tag{22}$$

Objective function 2

$$OF_2 = \min \sum_{k=1}^{K} S_k \tag{23}$$

Objective function 3

$$OF_3 = \min \sum_{k=1}^{K} T - R_k \tag{24}$$

subject to

$$\sum_{k=1}^{K} kX_{pk} - \sum_{k=1}^{K} kX_{ik} \le K\left(1+U_p-2U_i\right), \forall(p,i)|p \in PR_i \tag{25}$$

$$\sum_{k=1}^{K} kX_{sk} - \sum_{k=1}^{K} kX_{ik} \le KU_i \forall(i,s)|s \in SC_i \tag{26}$$

$$\sum_{k=1}^{K} X_{ik} = 1, \forall i \tag{27}$$

$$S_k \ge X_{ik},\ \forall i,k \tag{28}$$

$$S_k \le \sum_{i=1}^{I} X_{ik},\ \forall k \tag{29}$$

$$\sum_{i=1}^{I} t_i X_{ik} \le ct,\ \forall k \tag{30}$$

$$\sum_{u=1}^{K} W_{uk} = S_k,\ \forall k \tag{31}$$

$$\sum_{k=1}^{K} W_{uk} \le 1,\ \forall u \tag{32}$$

$$(1-q_{iu})B_{iuk} \le T-R_k,\ \forall i,k,u \tag{33}$$

$$W_{uk}+X_{ik} \ge 2Y_{iuk},\ \forall i,k,u \tag{34}$$

$$W_{uk}+X_{ik} \le 1+Y_{iuk},\ \forall i,k,u \tag{35}$$

$$\sum_{k=1}^{K} V_k = 1 \tag{36}$$

$$V_k \le S_k,\ \forall k \tag{37}$$

$$V_k \le T, \forall k \tag{38}$$

$$V_k \ge T-(1-S_k)\ \forall k \tag{39}$$

$$B_{iuk} \le Y_{iuk}\ \forall i,k,u \tag{40}$$

$$B_{iuk} \le T,\ \forall i,k,u \tag{41}$$

$$B_{iuk} \ge T - (1 - Y_{iuk}),\ \forall i,k,u \tag{42}$$

$$X_{ik} \le \frac{\sum\limits_{l \in Tl_i} Z_{lk}}{|Tl_i|},\ \forall i,k \tag{43}$$

$$X_{ik}, Z_{lk}, S_k, W_{uk}, Y_{iuk} \in \{0,1\},\ \forall i,k,l,u \tag{44}$$

$$T, V_k, R_k, B_{iuk} \ge 0,\ \forall k \tag{45}$$

In the next section of the paper, some meta-heuristic approaches are developed to solve the formulation (22)-(45) effectively.

## 3 Solution methodology

Generally, the assembly line balancing problems in their simple form are of the NP-hard class of combinatorial optimization problems (see Ogan and Azizoglu 2015). Therefore, we conclude that the linearized formulation (22)-(45) which contains many variables and constraints in addition to classical assembly line balancing problems of the literature, has a high degree of complexity to be solved exactly by optimization solvers. For this aim, first, the classical simulated annealing (SA) and the classical variable neighbourhood search (VNS) algorithms are proposed to solve the formulation (22)-(45). In continue, these algorithms are hybridized by a local search algorithm in order to obtain better solutions. In the rest of this section, first, an encoding/decoding scheme is proposed to construct and evaluate the solutions of the proposed meta-heuristic algorithms. Then the proposed meta-heuristic algorithms are explained.

### 3.1 Encoding/decoding scheme

A special encoding/decoding scheme is proposed to construct and evaluate a feasible solution for the formulation (22)-(45). The proposed meta-heuristic approaches of this study are mainly based on this scheme. This encoding/decoding scheme follows three steps as represented by Fig. 2 and its steps are detailed in continue.

**Step 1. Activity to station assignment.** According to the concept of U-shaped lines, either all of the predecessors or all of the successors of an eligible activity for being assigned to a candidate station should be assigned to earlier stations or that station. On the other hand, for each activity, a set of eligible stations is defined to assign it to one of them. Starting from Station 1, the last station contains one of the predecessors or successors of an activity, and the later stations are eligible for that activity. For example, if we consider Station 6 for assigning an activity and the last station containing its predecessors or successors is Station 3, the set of eligible stations for this activity contains stations 3, 4, 5, and 6. In the beginning, all of the stations, respectively from 1 to $K$ are eligible. Now, to perform this step, the below sub-steps are followed.

*Step 1.1.* A permutation of the activities is generated randomly as a solution vector.

*Step 1.2.* A set of eligible activities starting from the beginning of the solution vector are obtained, respectively. Also, a set of eligible stations is obtained for each eligible activity.

*Step 1.3.* The first eligible activity is assigned to its earliest possible eligible station. The condition for this assignment is that the operating time of the stations does not exceed the cycle time. If an eligible activity is assigned to a station, it is removed from the solution vector, and also it is removed from the predecessor and successor lists of the remaining activities of the solution vector.

*Step 1.4.* The last two steps are repeated until the solution vector becomes empty.

**Step 2. Equipment to station assignment.** In this step, according to the activity assignment plan obtained by Step 1, the required equipment of each opened station is determined from the required equipment of the activities of that station. For the cases that more than one activity from a station requires an equipment, just one unit of that equipment is assigned to that station.

**Step 3. Worker to station assignment.** A random permutation of the available workers is generated. Starting from the beginning of the permutation, the workers are assigned to the opened stations, respectively. Therefore, some of the workers are unassigned if all of the potential stations are not opened.

**Step 4. Solution evaluation.** The solution obtained by steps 1–3 is evaluated in this step. For this aim, the multi-objective function of the formulation (22)-(45) is converted to a single objective function by a typical global criterion. For this aim, the following sub-steps are followed.

*Step 4.1.* For the obtained solution, the objective functions of the formulation (22)-(45) are obtained easily by the Eqs. 1-3. For this aim, the variables $Z_{lk}$, $S_k$, and $Q_k$ are calculated easily from the obtained solution.

*Step. 4.2.* Calculate the following single objective function, which is a typical global criterion. According to the minimization type objective functions (1)-(3), smaller values of $OF$ are favoured.

$$OF = \frac{\sum\limits_{l=1}^{L}\sum\limits_{k=1}^{K} ec_l Z_{lk}}{UB_{OF_1}} + \frac{\sum\limits_{k=1}^{K} S_k}{UB_{OF_2}} + \frac{\sum\limits_{k=1}^{K} 1 - Q_k \Big/ \sum\limits_{k=1}^{K} S_k}{UB_{OF_3}} \tag{46}$$

In this equation, $UB_{OF_1}$ is a proposed upper bound for the value of objective function (1). In this global criterion,

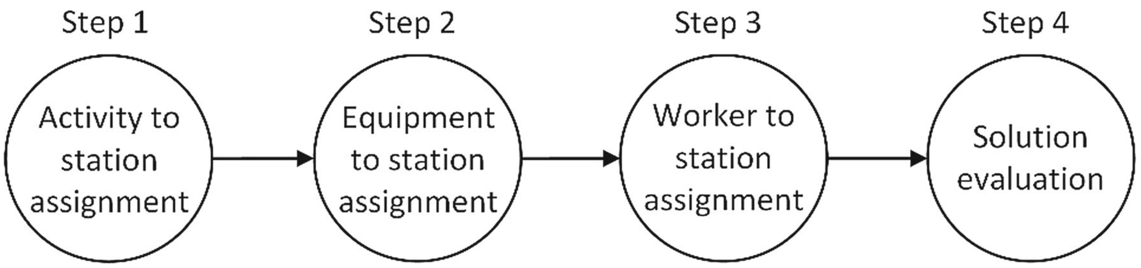
Step 1
Activity to station assignment
Step 2
Equipment to station assignment
Step 3
Worker to station assignment
Step 4
Solution evaluation

**Fig. 2** The proposed encoding/decoding scheme

each objective function value is converted to a scale between 0 and 1. Therefore, their summation has more sense and gives a more accurate evaluation. Regardless of the obtained solution, the following upper bounds are proposed in this study.

$$UB_{OF_1} = K\sum_{l=1}^{L} ec_l \tag{47}$$

$$UB_{OF_2} = K \tag{48}$$

$$UB_{OF_3} = K\max_{i,u}\{1 - q_{iu}\} \tag{49}$$

In Eq. 47, it is assumed that all of the equipment are assigned to all of the potential stations. In Eq. 48, it is assumed that all potential stations are opened. In Eq. 49, it is assumed that the highest possible dis-quality level appears in all of the potential stations.

In order to explain the calculations of this step, the example of Fig. 1 is considered. Some data from the example is shown in Table 1 and others are given below,

- Number of stations is considered $K = 5$.
- Three types of equipment are available where all of them are required for any of the
- activities. We consider $ec_1 = 1000$, $ec_2 = 2000$, and $ec_3 = 3000$.

Based on the given data and the U-shaped configuration of Fig. 1, the following calculations are done to obtain the value of the objective function (46).

- $UB_{OF_1}$ is calculated as $UB_{OF_1} = K\sum_{l=1}^{L} ec_l = 5 \times (1000 + 2000 + 3000) = 30000.$
- $UB_{OF_2} = K = 5.$
- $UB_{OF_3} = K\max_{i,u}\{1 - q_{iu}\} = 5\times$ $\max\{1 - 0.8, 1 - 0.8, \ldots, 1 - 0.9\} = 4 \times 0.4 = 2.$
- In the U-shaped configuration, all 5 stations are established and all of the equipment are assigned to each station because it has been assumed that each activity needs all of the equipment. Therefore, the total equipment cost of the U-shaped configuration is calculated as

$$\sum_{l=1}^{K}\sum_{k=1}^{K} ec_l Z_{lk} = (1000 + 2000 + 3000)$$
$$+ (1000 + 2000 + 3000) + (1000 + 2000 + 3000) = 18000$$

On the other hand, it is assumed that workers 1 to 5 are assigned to stations 1 to 5, respectively. Therefore, based on the activities and workers of the stations,

- $1 - Q_1 = \max\{1 - q_{31}, 1 - q_{41}, 1 - q_{10,1}\} = 0.3,$ $1 - Q_2 = 0.4,$ $1 - Q_3 = 0.4,$ $1 - Q_4 = 0.4,$ and $1 - Q_5 = 0.4.$ss
- Finally, the objective function (46) is calculated as below,
-

**Table 1** Data of the illustrative example

| Activity ($i$) | $t_i$ | $q_{iu}$ | | | | |
|---|---|---|---|---|---|---|
| | | $u = 1$ | $u = 2$ | $u = 3$ | $u = 4$ | $u = 5$ |
| 1 | 2 | 0.8 | 0.8 | 0.6 | 0.8 | 0.9 |
| 2 | 4 | 0.7 | 0.8 | 0.7 | 0.7 | 0.6 |
| 3 | 1 | 0.9 | 0.6 | 0.8 | 0.7 | 0.8 |
| 4 | 6 | 0.7 | 0.9 | 0.7 | 0.8 | 0.6 |
| 5 | 4 | 0.6 | 0.6 | 0.8 | 0.6 | 0.7 |
| 6 | 3 | 0.6 | 0.8 | 0.6 | 0.6 | 0.8 |
| 7 | 5 | 0.8 | 0.7 | 0.6 | 0.9 | 0.6 |
| 8 | 2 | 0.9 | 0.8 | 0.8 | 0.6 | 0.7 |
| 9 | 5 | 0.7 | 0.6 | 0.8 | 0.8 | 0.6 |
| 10 | 5 | 0.8 | 0.9 | 0.6 | 0.8 | 0.9 |

$$
\begin{aligned}
OF &= \frac{\sum_{l=1}^{K}\sum_{k=1}^{K} ec_l Z_{lk}}{UB_{OF_1}} + \frac{\sum_{k=1}^{K} S_k}{UB_{OF_2}} \\
&\quad + \frac{\sum_{k=1}^{K} 1 - Q_k \big/ \sum_{k=1}^{K} S_k}{UB_{OF_3}} \\
&= \frac{18000}{30000} + \frac{5}{5} + \frac{0.3 + 0.4 + 0.4 + 0.4 + 0.4 \big/ 5}{2} = 1.83
\end{aligned}
$$

## 3.2 Meta-heuristic solution approaches

### 3.2.1 Classical simulated annealing (SA-1)

Simulated annealing (SA) is a meta-heuristic algorithm for complex optimization problems based on a single solution, which is introduced by Kirkpatrick et al. (1983) (see also Niroomand and Vizvari, 2015). The objective function of the algorithm is like the energy of materials that should be decreased by decreasing the temperature. The search starts with the initial temperature ($T_0$) and the initial solution ($x_0$). The initial solution is changed to a new solution or neighbor.

solution ($x'$). If this change decreases the objective function (energy), the new solution is accepted, otherwise, it can be accepted by the probability of $\exp\left(\frac{-\Delta OF}{T_0}\right)$ where $\Delta OF = OF(x') - OF(x_0)$. The changing procedure is repeated several times for the accepted solution, and in each repetition, the accepted solution may be replaced by the new solution. After that, the initial temperature is cooled down and the whole procedure is repeated until reaching a stopping criterion like a final temperature, running time, etc. In the end, the best solution is reported. The SA is detailed by the pseudo-code of Fig. 3.

Some clarifications for applying the classical SA to the formulation (22)-(45) are as follow,

- The encoding/decoding scheme of classical SA is exactly the same as Sect. 3.1.
- In the beginning of the algorithm, a random permutation of the workers is generated and it is used in all of the generated solutions. Therefore, the common set of workers is used in all of the generated solutions.
- The neighbourhood search operator is the swap. In the swap operator, the current solution vector is considered. Two integer random numbers from 1 to $I$ are generated. The activities of the positions of these generated numbers are interchanged and a new solution vector is obtained.

### 3.2.2 SA algorithm with random worker assignment (SA-2)

The proposed classical SA of Sect. 3.2.1 is hybridized here in order to obtain a better solution. For this aim, at any generated solution during the SA, a random permutation of the workers is generated and applied to that solution. Therefore, the difference of this version of the SA with the SA of the previous sub-section is that just a common set of workers is not used for the generated solutions, and for each generated solution, a specific set of workers is generated randomly.

**Fig. 3** The classical SA used for the formulation (22)-(45)

```
Set parameters: an initial temperature ($T_0$), final temperature ($T_f$), cooling ratio ($\alpha$), number
of neighborhood in search each temperature $n_{max}$, $T = T_0$, and time limit ($time_{max}$)
Generate an initial solution ($x_0$)
$x = x_0$
Evaluate the fitness values ($OF$)
While time limit is not exceeded or $T < T_f$
        $w = 1$
        For $w = 1{:}n_{max}$
                Randomly select $x'$ from neighborhood structure
                If $OF(x') < OF(x)$
                   $x = x'$
                  Else
                  If $rand(0,1) \le \exp(-\Delta OF/T)$
                     $x = x'$
                  End
                End
                $w = w + 1$
        End
        $T = \alpha T$
End
Report the best obtained solution
```

### 3.2.3 SA algorithm with local search for worker assignment (SA-3)

In this sub-section, another hybrid version of the classical SA is proposed to improve the solutions found for the formulation (22)-(45). For this aim, the SA of Sect. 3.2.2 is considered and improved. The improvement is done by applying a local search in each worker assignment step of the encoding/decoding scheme (Step 3). For this aim, the following are done in the worker assignment step when a new solution (neighbor solution) is generated,

- The worker assignment step of the encoding/decoding scheme (Step 3) is repeated for a number of iterations ($n_w$).
- As in each iteration, a random permutation of the workers is generated, this permutation is changed by a swap (similar to the swap operator used in the classical SA) operator for a number of iterations ($n_{it}$) in order to improve the worker assignment as much as possible.
- Among the $n_w n_{it}$ generated set of workers, the set that results in less value of objective function (3) obtained by formulation

$$\sum_{k=1}^{K} 1-Q_k \Big/ \sum_{k=1}^{K} S_k$$

  is selected for worker assignment of the neighbor solution.

### 3.2.4 Classical variable neighbourhood search (VNS-1)

The VNS is a single solution meta-heuristic algorithm to solve the combinatorial optimization problems effectively (see Niroomand and Vizvari, 2015). It contains a set of neighbourhood search structures (this set is shown by NS). The procedure starts with an initial solution and takes one of the neighbourhood search structures (NSS) and tries to improve the initial solution for a number of iterations. The improved solution (if improved) or initial solution (if not improved) is transferred to the next neighbourhood search structure (NSS). The procedure is repeated for all neighbourhood search structures (NSS). The whole procedure may be repeated for a number of iterations. Finally, the best-obtained solution is reported.

Some clarifications for applying the classical VNS to the formulation (22)-(45) are as follow,

- The encoding/decoding scheme is exactly the same as Sect. 3.1.
- At the beginning of the algorithm, a random permutation of the workers is generated and it is used in all of the generated solutions. Therefore, the common set of workers is used in all of the generated solutions.
- Three neighbourhood search structure are used as follow,
- NSS-1: In the solution vector, an activity is selected randomly and all activities before the selected activity is transferred to the end of the vector in the same order.
- NSS-2: In the solution vector, an activity is selected randomly and is transferred to another randomly selected position.
- NSS-3: The neighbourhood search structure is the swap introduced in Sect. 3.2.1.

The pseudo-code of the VNS-1 is shown in Fig. 4.

### 3.2.5 VNS algorithm with random worker assignment (VNS-2)

The proposed classical VNS of Sect. 3.2.4 is hybridized here in order to obtain a better solution. For this aim, at any generated solution during the VNS, a random permutation of the workers is generated and applied to that solution. Therefore, the difference of this version of the VNS with the VNS-1 is that just a common set of workers is not used for the generated solutions, and for each generated solution, a specific set of workers is generated randomly.

### 3.2.6 VNS algorithm with local search for worker assignment (VNS-3)

In this sub-section, another hybrid version of the classical VNS is proposed to improve the solutions found for the formulation (22)-(45). For this aim, the VNS-2 is considered and improved. The improvement is done by applying a local search in each worker assignment step of the encoding/decoding scheme (Step 3). For this aim, the followings are done in the worker assignment step when a new solution (neighbor solution) is generated,

- The worker assignment step of the encoding/decoding scheme (Step 3) is repeated for a number of iterations ($n_w$).
- As in each iteration, a random permutation of the workers is generated, this permutation is changed by a swap (similar to the swap operator used in the classical VNS) operator for a number of iterations ($n_{it}$) in order to improve the worker assignment as much as possible. Among the $n_w n_{it}$ generated set of workers, the set that results in less value of objective function (3) obtained by formulation

$$\sum_{k=1}^{K} 1-Q_k \Big/ \sum_{k=1}^{K} S_k$$

  is selected for worker assignment of the neighbor solution.

**Fig. 4** The classical VNS used for the formulation (22)-(45)

```
Set parameters: set of neighborhood search structures $NS = \{NSS-1, NSS-2, NSS-3\}$,
iterations in each NSS which is $n_1$, $n_2$, and $n_3$ for NSS-1, NSS-2, and NSS-3 respectively,
algorithm iterations W
Generate an initial solution ($x_0$)
$x = x_0$
Evaluate the fitness values ($OF$)
While time limit is not exceeded or $w \le W$
    $w = 1$
    For $i = 1:3$
        While $w_i \le n_i$
        Randomly select $x'$ from $NSS - i$
        If $OF(x') < OF(x)$
            $x = x'$
        End
        $w_i = w_i + 1$
    End
    $w = w + 1$
End
Report the best obtained solution
```

### 3.2.7 Classical genetic algorithm and its improved versions

For more comparison of the proposed algorithms, the GA is also considered in this study (similar to the GA applied by Gen et al. (1997) and Niroomand et al. (2016)). The classical and modified versions of the GA is briefly explained as below,

- GA-1: In this GA, a population of solutions is generated randomly (the number of solutions of the population is denoted by *N*). Each solution is generated and evaluated by the encoding–decoding scheme of Sect. 3.1. A roulette wheel mechanism (see Gen et al. 1997) is used and two solutions (parents) from the population are selected. The partial mapped crossover (PMX) (see Niroomand et al. 2016) is applied to the sequences of the first step of the solutions to form a new solution (offspring). Then, with a mutation rate ($p_r$) a swap operator (see Niroomand et al. 2016) is applied on the sequence of the first step of the new solution for possible improvement. The remaining $N-2$ solutions of the population are regenerated randomly and are added to the population. The new solution is also added to the population while the parents remain in the population. Then the best $N$ solutions among all solutions of the population are selected to form a new population. The procedure is repeated for a number of iterations ($n_{max}$) and the best-obtained solution among the last population is introduced as the best solution of the GA. Some clarifications for applying the classical SA to the formulation (22)-(45) are as follow,
- The encoding/decoding scheme of the GA-1 is exactly the same as Sect. 3.1.
- At the beginning of the algorithm, a random permutation of the workers is generated and it is used in all of the generated solutions. Therefore, the common set of workers is used in all of the generated solutions.
- GA-2: The proposed GA-1 is hybridized here in order to obtain a better solution. For this aim, in any generated solution during the procedure of GA-1, a random permutation of the workers is generated and applied in that solution. Therefore, the difference of this version of the GA with the GA-1 is that just a common set of workers is not used for the generated solutions, and for each generated solution, a specific set of workers is generated randomly.
- GA-3: In this sub-section, another hybrid version of the GA-1 is proposed to improve the solutions found for the formulation (22)-(45). For this aim, the GA-2 is considered and improved. The improvement is done by applying a local search in each worker assignment step of the encoding/decoding scheme (Step 3). For this aim, the followings are done in the worker assignment step when a new solution (neighbor solution) is generated,
- The worker assignment step of the encoding/decoding scheme (Step 3) is repeated for a number of iterations ($n_w$).
- As in each iteration, a random permutation of the workers is generated, this permutation is changed by a swap (similar to the swap operator used in the GA-1) operator for a number of iterations ($n_{it}$) in order to improve the worker assignment as much as possible. Among the $n_w n_{it}$ generated set of workers, the set that results in less value of objective function (3) obtained by formulation

$$\sum_{k=1}^{K} 1 - Q_k / \sum_{k=1}^{K} S_k$$

is selected for worker assignment of the neighbor solution.

## 4 Computational study

The formulation (22)-(45) and the proposed meta-heuristic solution approaches are numerically evaluated in this section. For this aim, some test problems are generated. Using these test problems, the behavior of the parameters of the meta-heuristic algorithms is studied experimentally and the best value for each parameter is determined. Then the final experiments using the obtained values of the parameters are performed. It is notable to mention that the meta-heuristic solution approaches are coded in MATLAB and are run on a computer Core i7- 16.00 GB RAM—3.88 GHz CPU. The rest of this section describes the details of the computational study.

### 4.1 Test problems

As mentioned earlier, some test problems are generated to evaluate the proposed meta-heuristic solution approaches. In this section, some characteristics of these test problems are detailed. The interested readers can contact the corresponding author for the complete set of data of the test problems.

**Table 2** Some characteristics of the test problems

| Test problem | *I* | *K* | *U* | *L* | *CT* |
|---|---|---|---|---|---|
| TP1 | 11 | 11 | 11 | 4 | 55 |
| TP2 | 11 | 11 | 11 | 8 | 55 |
| TP3 | 25 | 25 | 25 | 5 | 14 |
| TP4 | 25 | 25 | 25 | 10 | 14 |
| TP5 | 35 | 35 | 35 | 5 | 50 |
| TP6 | 35 | 35 | 35 | 10 | 50 |
| TP7 | 53 | 53 | 53 | 5 | 2000 |
| TP8 | 53 | 53 | 53 | 10 | 2000 |
| TP9 | 70 | 70 | 70 | 5 | 200 |
| TP10 | 70 | 70 | 70 | 10 | 200 |
| TP11 | 89 | 89 | 89 | 5 | 90 |
| TP12 | 89 | 89 | 89 | 10 | 90 |
| TP13 | 111 | 111 | 111 | 5 | 580 |
| TP14 | 111 | 111 | 111 | 10 | 580 |
| TP15 | 148 | 148 | 148 | 5 | 140 |
| TP16 | 148 | 148 | 148 | 10 | 140 |

Except the characteristics represented by Table 2, the following data are also considered for the test problems which are not reported in the paper.

- Activity operating time, which is randomly generated between 1 and *CT*.
- Equipment purchasing cost, which is randomly generated for each equipment.
- Quality level of performing each activity by each worker which is randomly generated between 0 and 1.
- Set of equipment used for each activity.

In order to study the performance of the proposed meta-heuristics deeply, the size of the test problems are considered from a small size to a large size.

### 4.2 Parameter tuning

A very important step in implementing any meta-heuristic algorithm is to determine the best value of its parameters. A very simple but time-consuming and even impossible way is to consider all possible values of the parameters of a meta-heuristic algorithm and determine all of the combinations of the parameters. These combinations determine the number of experiments to be performed for studying the effect of the parameters on the performance of the algorithm.

As an effective method to deal with such difficulty and reduce the number of required experiments, Taguchi (1986) proposed Taguchi experimental design method which has been used in many optimization-based studies like Taassori et al. (2015), Niroomand et al. (2015), Mirzaei et al. (2016), Niroomand et al. (2016), Sanei et al. (2016), Boros et al. (2016), Salehi et al. (2020), etc. The process of the Taguchi method divides the parameters into two groups of uncontrollable and controllable parameters. During the process, it tries to decrease the impact of uncontrollable parameters and find the best value of the controllable parameters. For this aim, the Taguchi method uses orthogonal arrays to decrease the number of required experiments. The number of rows (experiments) of the orthogonal array of an algorithm with four parameters and three levels in each parameter is at least $(4 \times (3 - 1)) + 1 = 9$. If there is no orthogonal array with exactly nine number of rows, the larger arrays are modified. The result of experiments of the orthogonal array is a signal-to-noise ratio (S/N ratio) to be minimized in the case of best values of the parameters. For a minimization type objective function, this ratio is as follow,

$$S/N = -10\left(\log_{10}(objective\,function)^2\right) \tag{50}$$

**Table 3** The levels considered for the parameters of the proposed meta-heuristics

| Algorithm | Levels of the parameters | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $T_0$ | $\alpha$ | $n_{\max}$ | $T_f$ | $n_w$ | $n_{it}$ | $W$ | $n_1$ | $n_2$ | $n_3$ | $N$ | $p_r$ | $n_{\max}$ |
| SA-1 | 50, 100, 250 | 0.8, 0.9, 0.99 | 20, 80, 200 | 5 | – | – | – | – | – | – | – | – | – |
| SA-2 | 50, 100, 250 | 0.8, 0.9, 0.99 | 20, 80, 200 | 5 | – | – | – | – | – | – | – | – | – |
| SA-3 | 100, 250 | 0.85, 0.99 | 40, 80 | 5 | 25, 50 | 25, 50 | – | – | – | – | – | – | – |
| VNS-1 | – | – | – | – | – | – | 50, 100, 200 | 50, 100, 200 | 50, 100, 200 | 50, 100, 200 | – | – | – |
| VNS-2 | – | – | – | – | – | – | 50, 100, 200 | 50, 100, 200 | 50, 100, 200 | 50, 100, 200 | – | – | – |
| VNS-3 | – | – | – | – | 25, 50 | 25, 50 | 50, 100, 200 | 50, 100, 200 | 50, 100, 200 | 50, 100, 200 | – | – | – |
| GA-1 | – | – | – | – | | | – | – | – | – | 50, 100, 150 | 0.4, 0.6, 0.8 | 50, 150, 300 |
| GA-2 | – | – | – | – | 25, 50 | 25, 50 | – | – | – | – | 50, 100, 150 | 0.4, 0.6, 0.8 | 50, 150, 300 |
| GA-3 | – | – | – | – | 25, 50 | 25, 50 | – | – | – | – | 50, 100, 150 | 0.4, 0.6, 0.8 | 50, 150, 300 |

For the meta-heuristics of this study, two or three values for each parameter is proposed which are mainly taken from the literature. These values are reported in Table 3 and the total required number of experiments is also obtained there to show the complexity of simple parameter tuning for these algorithms. For the algorithms based on SA, in final temperature, only one value is considered, which means that this parameter is not studied for tuning purposes. On the other hand, the most suitable orthogonal array for each meta-heuristic is obtained and its experiments using values of Table 3 are designed.

The experiments of the orthogonal arrays are performed on TP6 test problem. Each experiment is run five times to obtain more reliable results. The average objective function values is used to obtain the S/N ratio of each parameter level and finally, the best value of each parameter is obtained and reported by Table 4.

**Table 4** The best level obtained for the parameters of the proposed meta-heuristics

| Algorithm | Levels of the parameters | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $T_0$ | $\alpha$ | $n_{\max}$ | $T_f$ | $n_w$ | $n_{it}$ | $W$ | $n_1$ | $n_2$ | $n_3$ | $N$ | $p_r$ | $n_{\max}$ |
| SA-1 | 250 | 0.99 | 200 | 5 | – | – | – | – | – | – | – | – | – |
| SA-2 | 250 | 0.99 | 200 | 5 | – | – | – | – | – | – | – | – | – |
| SA-3 | 250 | 0.99 | 80 | 5 | 50 | 50 | – | – | – | – | – | – | – |
| VNS-1 | – | – | – | – | – | – | 200 | 100 | 100 | 200 | – | – | – |
| VNS-2 | – | – | – | – | – | – | 200 | 100 | 100 | 200 | – | – | – |
| VNS-3 | – | – | – | – | 50 | 50 | 200 | 100 | 100 | 200 | – | – | – |
| GA-1 | – | – | – | – | | | – | – | – | – | 150 | 0.6 | 300 |
| GA-2 | – | – | – | – | 50 | 50 | – | – | – | – | 150 | 0.6 | 300 |
| GA-3 | – | – | – | – | 25 | 50 | – | – | – | – | 150 | 0.8 | 300 |

Table 5 The results obtained for the test problems by the proposed meta-heuristic approaches SA-1, SA-2, and SA-3

| Test problem | SA-1 | SA-2 | SA-3 | VNS-1 | VNS-2 | VNS-3 | GA-1 | GA-2 | GA-3 | Exact solution (CPLEX) |
|---|---|---|---|---|---|---|---|---|---|---|
| TP1 | 1.411 | 1.238 | 1.185 | 1.429 | 1.240 | 1.185 | 1.360 | 1.126 | 1.096 | 1.011 |
| TP2 | 1.342 | 1.204 | 1.166 | 1.407 | 1.181 | 1.149 | 1.378 | 1.211 | 1.114 | 1.019 |
| TP3 | 1.192 | 1.202 | 1.126 | 1.259 | 1.250 | 1.118 | 1.351 | 1.132 | 1.125 | 1.112 |
| TP4 | 1.191 | 1.204 | 1.136 | 1.294 | 1.257 | 1.155 | 1.359 | 1.166 | 1.122 | 1.049 |
| TP5 | 1.221 | 1.177 | 1.112 | 1.229 | 1.210 | 1.117 | 1.433 | 1.176 | 1.115 | 1.101 |
| TP6 | 1.242 | 1.191 | 1.114 | 1.259 | 1.201 | 1.123 | 1.354 | 1.159 | 1.117 | 1.018 |
| TP7 | 1.194 | 1.090 | 1.032 | 1.203 | 1.218 | 1.035 | 1.397 | 1.201 | 1.086 | – |
| TP8 | 1.217 | 1.093 | 1.031 | 1.204 | 1.212 | 1.055 | 1.379 | 1.180 | 1.089 | – |
| TP9 | 1.192 | 1.185 | 1.135 | 1.195 | 1.198 | 1.136 | 1.344 | 1.165 | 1.097 | – |
| TP10 | 1.202 | 1.195 | 1.141 | 1.205 | 1.203 | 1.147 | 1.333 | 1.201 | 1.083 | – |
| TP11 | 1.293 | 1.133 | 1.067 | 1.224 | 1.181 | 1.098 | 1.308 | 1.212 | 1.090 | – |
| TP12 | 1.204 | 1.217 | 1.070 | 1.342 | 1.198 | 1.139 | 1.344 | 1.149 | 1.122 | – |
| TP13 | 1.317 | 1.206 | 1.082 | 1.257 | 1.134 | 1.107 | 1.235 | 1.205 | 1.083 | – |
| TP14 | 1.241 | 1.198 | 1.117 | 1.394 | 1.202 | 1.128 | 1.364 | 1.244 | 1.125 | – |
| TP15 | 1.376 | 1.199 | 1.120 | 1.206 | 1.104 | 1.136 | 1.280 | 1.152 | 1.109 | – |
| TP16 | 1.299 | 1.120 | 1.092 | 1.224 | 1.217 | 1.124 | 1.284 | 1.246 | 1.092 | – |

## 4.3 Final experiments

The best level of the parameters obtained by Sect. 4.2 are used to run the proposed algorithms for all test problems. In order to have a fair comparison among the proposed algorithms, the following conditions are considered when running the proposed meta-heuristic algorithms,

- A common CPU time equal to the CPU time of the longest algorithm in each test problem is considered to run that test problem by all of the algorithms. This issue considers two points, (1) all algorithms are run for equal CPU time, and (2) as the considered CPU time is depended on the size of the test problems, for larger test problems, larger CPU time is considered.
- In order to obtain more reliable results, each test problem is solved by each algorithm 20 times.

In addition, the problem (22)-(45) is solved by CPLEX solver of GAMS for obtaining the optimal solution of each test problem. For this aim, each objective function is divided by its upper bound explained by Eqs. (47)-(49) and the obtained fractions are summed up. The obtained single objective function is identical to the objective function (46), which is used to evaluate the solutions generated by the meta-heuristic solution approaches. The CPU running time of 30,000 s is considered for all test problems while they are solved by GAMS. The obtained results by exact experiments is depicted in the last column of Table 5. It can be seen that only test problems TP1 to TP6 can be solved exactly in the given CPU running time. For other test problems, no optimal solution is obtained at that running time.

The results obtained by final experiments are reported in Table 5, Figs. 5, 6, and 7. In these table and figures, the minimum, maximum and average of the objective function values obtained by each meta-heuristic algorithm for each test problem are reported.

The results of Table 5 are represented based on the average of the objective function values obtained by each meta-heuristic algorithm. The graphs of the objective function values are depicted in Figs. 5, 6, and 7 where minimum obtained value, maximum obtained value, and average of values of all 20 runs of each meta-heuristic algorithm for each test problem are represented, respectively. According to the obtained results, the following conclusions are drawn from the results of Table 5 and Figs. 5, 6, 7.

- In all test problems, among the SA-based algorithms, the best average of the obtained objective function values is related to the SA-3 algorithm. After this algorithm, the SA-2 algorithm results in a better average of the obtained objective function values compared to the SA-1 algorithm for most of the test problems. Among the VNS-based algorithms, the VNS-3 performs better than the VNS-2 and VNS-1 algorithms. Among the GA-based algorithms, the GA-3 performs better than the GA-2 and GA-1 algorithms.
- In all test problems, the best minimum value of the obtained objective function values is related to either GA-3 or SA-3 algorithms. After these algorithms, the

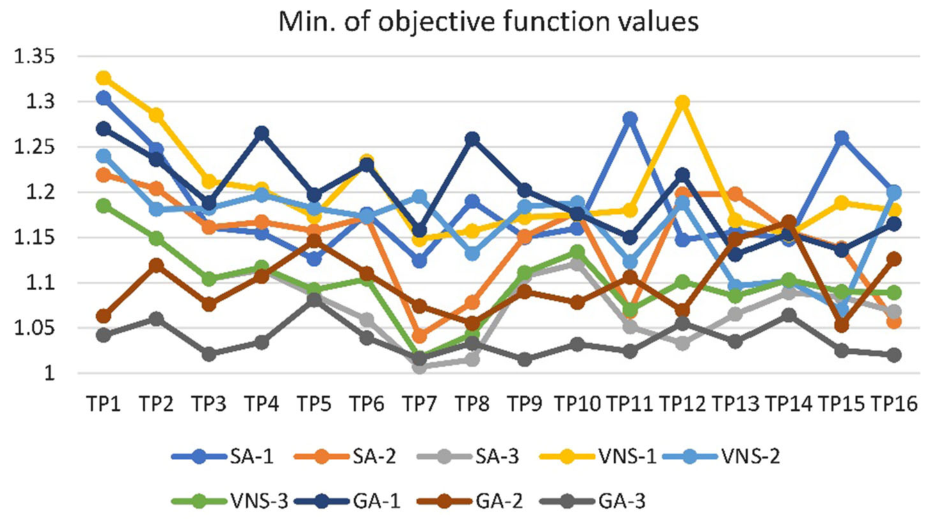

**Fig. 5** Minimum objective function values obtained by the metaheuristic algorithms for all test problems

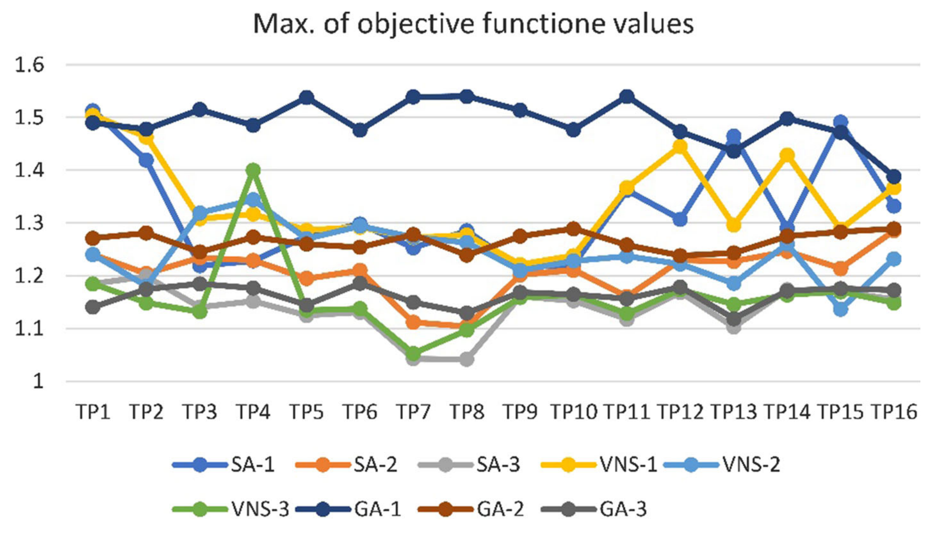

**Fig. 6** Maximum objective function values obtained by the metaheuristic algorithms for all test problems

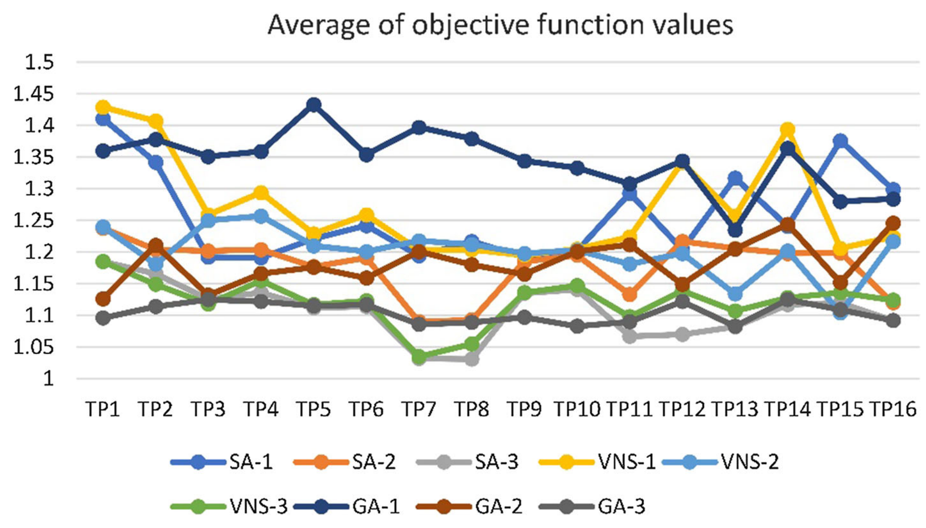

**Fig. 7** Average of objective function values obtained by the metaheuristic algorithms for all test problems

**Table 6** The results of one-way ANOVA for comparison of the algorithms overall test problems (the sign " < / = / > " means that the algorithm of row performs better/equal/worse than the algorithm of the column in 0.05 $\alpha$-level)

| Algorithm | SA-1 | SA-2 | SA-3 | VNS-1 | VNS-2 | VNS-3 | GA-1 | GA-2 | GA-3 |
|---|---|---|---|---|---|---|---|---|---|
| SA-1 | – | < | < | = | < | < | = | < | < |
| SA-2 | – | – | < | > | = | < | > | = | < |
| SA-3 | – | – | – | > | > | = | > | > | = |
| VNS-1 | – | – | – | – | < | < | = | < | < |
| VNS-2 | – | – | – | – | – | < | > | = | < |
| VNS-3 | – | – | – | – | – | – | > | > | = |
| GA-1 | – | – | – | – | – | – | – | < | < |
| GA-2 | – | – | – | – | – | – | – | – | < |
| GA-3 | – | – | – | – | – | – | – | – | – |

**Table 7** CPU running times of the algorithms and common run time used for the final experiments in seconds

| Test problem | SA-1 | SA-2 | SA-3 | VNS-1 | VNS-2 | VNS-3 | GA-1 | GA-2 | GA-3 | Common run time |
|---|---|---|---|---|---|---|---|---|---|---|
| TP1 | 25 | 25 | 1100 | 30 | 30 | 1400 | 25 | 25 | 1300 | 1400 |
| TP2 | 25 | 25 | 1100 | 25 | 25 | 1400 | 25 | 25 | 1300 | 1400 |
| TP3 | 100 | 100 | 2300 | 140 | 140 | 2900 | 130 | 130 | 2700 | 2900 |
| TP4 | 100 | 100 | 2300 | 140 | 140 | 2900 | 130 | 130 | 2700 | 2900 |
| TP5 | 210 | 210 | 3500 | 300 | 300 | 4300 | 250 | 250 | 4000 | 4300 |
| TP6 | 210 | 210 | 3500 | 300 | 300 | 4300 | 250 | 250 | 4000 | 4300 |
| TP7 | 660 | 660 | 5800 | 900 | 900 | 7400 | 810 | 810 | 7000 | 7400 |
| TP8 | 660 | 660 | 5800 | 900 | 900 | 7400 | 810 | 810 | 7000 | 7400 |
| TP9 | 1500 | 1500 | 8500 | 2100 | 2100 | 11,500 | 1900 | 1900 | 10,500 | 11,500 |
| TP10 | 1500 | 1500 | 8500 | 2100 | 2100 | 11,500 | 1900 | 1900 | 10,500 | 11,500 |
| TP11 | 3600 | 3600 | 10,800 | 4100 | 4100 | 12,600 | 3900 | 3900 | 11,700 | 12,600 |
| TP12 | 3600 | 3600 | 10,800 | 4100 | 4100 | 12,600 | 3900 | 3900 | 11,700 | 12,600 |
| TP13 | 5300 | 5300 | 11,600 | 6200 | 6200 | 13,800 | 5800 | 5800 | 13,100 | 13,800 |
| TP14 | 5300 | 5300 | 11,600 | 6200 | 6200 | 13,800 | 5800 | 5800 | 13,100 | 13,800 |
| TP15 | 6900 | 6900 | 13,200 | 7900 | 7900 | 15,000 | 7200 | 7200 | 14,200 | 15,000 |
| TP16 | 6900 | 6900 | 13,200 | 7900 | 7900 | 15,000 | 7200 | 7200 | 14,200 | 15,000 |

SA-2 and GA-2 algorithms result in better minimum value of the obtained objective function values compared to others.

- In all test problems, the best maximum value of the obtained objective function values is related to either SA-3 or GA-3 algorithms. After these algorithms, the SA-2 algorithm results in a better maximum value of the obtained objective function values compared to the SA-1 algorithm is most of the test problems. Among the VNS-based algorithms, the VNS-3 performs better than the VNS-2 and VNS-1 algorithms.
- In the case of the minimum obtained objective function value, the GA-3 algorithm performs definitely better than the other algorithms.
- The obtained results prove the superiority of the SA-3, VNS-3, and GA-3 algorithms over others. This means that, applying a local search on the worker assignment step in the classical algorithms, makes them a powerful solution approach compared to the classical SA, VNS, and GA (the SA-1, VNS-1, and GA-1 algorithms).

For more exact comparison of the obtained results of Table 5, a one-way ANOVA is used to statistically compare them in 0.05 $\alpha$-level. The result of ANOVA shows that there is a significant difference among the average of objective function values of Table 5. In continue, the Tukey test is performed to exactly compare the algorithms based on the average of objective function values of Table 5. The result of the Tukey test is shown by Table 6 where the sign " $< / = / >$ " means that the algorithm of row performs better/equal/worse than the algorithm of column in 0.05 $\alpha$-level. According to the results of Table 6, the SA-1, VNS-1, and GA-1 algorithms are strongly worse than the others while the algorithms SA-3, VNS-3, and GA-3 are strongly better than the others. Interestingly, the algorithms SA-3, VNS-3, and GA-3 are equal in all of the test problems. This means that, applying a local search on the worker assignment step in the classical algorithms makes them a powerful solution approach compared to the classical SA, VNS, and GA (the SA-1, VNS-1, and GA-1 algorithms).

### 4.4 Managerial insights

The mathematical formulation, solution procedure, and obtained results show the following implications from the managerial point of view,

- For industrial holdings, the formulation and solution approaches can be used to establish the assembly line of a new product for the first time.
- The proposed solution approaches can help the managers to overcome multi-objective nature of real-world multi-objective optimization problems.
- The proposed improvement procedure of the proposed algorithms can be modified for other assembly line balancing problems.

## 5 Conclusions

In this paper, a new U-shaped assembly line balancing problem was studied. Comparing to the literature, a new multi-objective formulation for U-shaped assembly line balancing problem was presented. The problem considers the criteria such as equipment purchasing cost, number of stations, and activity performing quality simultaneously and optimizes them by activity and worker allocations. Since the proposed U-shaped assembly line balancing problem is an NP-hard problem, the use of meta-heuristic algorithms for the problem is inevitable. As solution approach, the classical simulated annealing (SA), variable neighborhood search (VNS), and genetic algorithm (GA) with a novel encoding/decoding scheme were used and hybridized according to the characteristics of the problem. In order to evaluate the proposed meta-heuristics, because the problem is new, some test problems were generated randomly. Computational study of the paper including sensitivity analysis of the proposed meta-heuristics and final comparison of them, proved the effectiveness of the hybrid versions of the SA, VNS, and GA comparing to other approaches. It means that in the problem of this study applying the local search on the worker assignment phase (the hybrid algorithms SA-3, VNS-3, and GA-3) dramatically improved the obtained solutions (Table 7).

As the future study on the problem of this paper, the problem can be tackled by multi-objective meta-heuristic solution approaches. Furthermore, considering the problem in an uncertain environment and applying the methods such as stochastic programming, fuzzy programming, interval programming, etc. can be of interest.

**Funding** This study was not funded by any organization.

## Declarations

**Conflict of interest** Author **Morteza Khorram** declares that he has no conflict of interest. Author **Mahmood Eghtesadifard** declares that he has no conflict of interest. Author **Sadegh Niroomand** declares that he has no conflict of interest.

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

## References

Ağpak K, Yegül MF, Gökçen H (2012) Two-sided U-type assembly line balancing problem. Int J Prod Res 50(18):5035–5047. https://doi.org/10.1080/00207543.2011.631599

Aigbedo H, Monden Y (1997) A parametric procedure for multicriterion sequence scheduling for Just-In-Time mixed-model assembly lines. Int J Prod Res 35(9):2543–2564. https://doi.org/10.1080/002075497194651

Ajenblit DA, Wainwright RL (1998) Applying genetic algorithms to the U-shaped assembly line balancing problem. In: *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on* (pp. 96–101). IEEE. https://doi.org/10.1109/ICEC.1998.699329

Alavidoost MH, Zarandi MF, Tarimoradi M, Nemati Y (2017) Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times. J Intell Manuf 28(2):313–336. https://doi.org/10.1007/s10845-014-0978-4

Baybars I (1986) A survey of exact algorithms for the simple assembly line balancing problem. Manage Sci 32(8):909–932. https://doi.org/10.1287/mnsc.32.8.909

Baykasoglu A (2006) Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. J Intell Manuf 17(2):217–232. https://doi.org/10.1007/s10845-005-6638-y

Boros P, Fehér O, Lakner Z, Niroomand S, Vizvári B (2016) Modeling supermarket re-layout from the owner's perspective. Ann Oper Res 238(1–2):27–40

Charnes A, Cooper WW (1962) Programming with linear fractional functionals. Naval Res Logist Q 9(3–4):181–186. https://doi.org/10.1002/nav.3800090303

Chica M, Cordon O, Damas S (2011) An advanced multiobjective genetic algorithm design for the time and space assembly line balancing problem. Comput Ind Eng 61(1):103–117. https://doi.org/10.1016/j.cie.2011.03.001

Delice Y, Aydoğan EK, Özcan U, İlkay MS (2017) Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm. 4OR. https://doi.org/10.1007/s10288-016-0320-4

Erel E, Sabuncuoglu I, Aksu BA (2001) Balancing of U-type assembly systems using simulated annealing. Int J Prod Res 39(13):3003–3015. https://doi.org/10.1080/00207540110051905

Foroughi A, Gökçen H, Tiacci L (2016) The cost-oriented stochastic assembly line balancing problem: a chance constrained programming approach. Int J Ind Eng, 23(6)

Foroughi A, Gökçen H (2018) A multiple rule-based genetic algorithm for cost-oriented stochastic assembly line balancing problem. Assem Autom. https://doi.org/10.1108/AA-03-2018-050

Gen M, Cheng R, Wang D (1997) Genetic algorithms for solving shortest path problems. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97) (pp. 401–406). IEEE

Gökçen H, Kara Y, Atasagun Y (2010) Integrated line balancing to attain Shojinka in a multiple straight line facility. Int J Comput Integr Manuf 23(5):402–411. https://doi.org/10.1080/09511921003642162

Hadi-Vencheh A (2011) A new nonlinear model for multiple criteria supplier-selection problem. Int J Comput Integr Manuf 24(1):32–39. https://doi.org/10.1080/0951192X.2010.527372

Hadi-Vencheh A, Mohamadghasemi A (2013) An integrated AHP–NLP methodology for facility layout design. J Manuf Syst 32(1):40–45. https://doi.org/10.1016/j.jmsy.2012.07.009

Hamta N, Ghomi SF, Jolai F, Shirazi MA (2013) A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. Int J Prod Econ 141(1):99–111. https://doi.org/10.1016/j.ijpe.2012.03.013

Hamta N, Ghomi SF, Tavakkoli-Moghaddam R, Jolai F (2014) A hybrid meta-heuristic for balancing and scheduling assembly lines with sequence-independent setup times by considering deterioration tasks and learning effect Scientia Iranica. Trans E, Ind Eng 21(3):963

Kara Y, Paksoy T, Chang CT (2009) Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. Eur J Oper Res 195(2):335–347. https://doi.org/10.1016/j.ejor.2008.01.003

Khorram M (2018) A cost based mathematical formulation for U-type assembly line balancing problem. Ann Optim Theory and Pr 1(1):11–21

Kim YK, Kim Y, Kim YJ (2000) Two-sided assembly line balancing: a genetic algorithm approach. Prod Plan Control 11(1):44–53. https://doi.org/10.1080/095372800232478

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680. https://doi.org/10.1126/science.220.4598.671

Kovács G, Nagy B, Vizvári B (2017) An integer programming approach to characterize digital disks on the triangular grid. In: International Conference on Discrete Geometry for Computer Imagery (pp. 94–106). Springer, Cham. https://doi.org/10.1007/978-3-319-66272-5_9

Li Z, Kucukkoc I, Tang Q (2017) New MILP model and station-oriented ant colony optimization algorithm for balancing U-type assembly lines. Comput Ind Eng 112:107–121. https://doi.org/10.1016/j.cie.2017.07.005

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

Li Z, Janardhanan MN, Ashour AS, Dey N, Li Z, Janardhanan MN, Ashour AS, Dey N (2019) Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem. Neural Comput Appl 31(12):9095–9111. https://doi.org/10.1007/s00521-018-3957-4

Li Z, Janardhanan MN, Rahman HF (2021) Enhanced beam search heuristic for U-shaped assembly line balancing problems. Eng Optim 53(4):594–608

Make MRA, Rashid MFFA, Razali MM (2017) A review of two-sided assembly line balancing problem. Int J Adv Manuf Technol 89(5–8):1743–1763. https://doi.org/10.1007/s00170-016-9158-3

Manavizadeh N, Hosseini NS, Rabbani M, Jolai F (2013) A simulated annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach. Comput Ind Eng 64(2):669–685. https://doi.org/10.1016/j.cie.2012.11.010

McMullen PR, Frazier GV (1998) Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. Int J Prod Res 36(10):2717–2741. https://doi.org/10.1080/002075498192454

Miltenburg J (2002) Balancing and scheduling mixed-model U-shaped production lines. Int J Flex Manuf Syst 14(2):119–151. https://doi.org/10.1023/A:1014434117888

Miltenburg GJ, Wijngaard J (1994) The U-line line balancing problem. Manage Sci 40(10):1378–1388. https://doi.org/10.1287/mnsc.40.10.1378

Mirzaei N, Niroomand S, Zare R (2016) Application of statistical process control in service industry. J Model Manag 11(3):763–782

Niroomand S, Vizvari B (2015) Exact mathematical formulations and metaheuristic algorithms for production cost minimization: a case study of the cable industry. Int Trans Oper Res 22(3):519–544. https://doi.org/10.1111/itor.12096

Niroomand S, Hadi-Vencheh A, Şahin R, Vizvári B (2015) Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. Exp Syst Appl 42(19):6586–6597

Niroomand S, Hadi-Vencheh A, Mirzaei N, Molla-Alizadeh-Zavardehi S (2016) Hybrid greedy algorithms for fuzzy tardiness/earliness minimisation in a special single machine scheduling problem: case study and generalisation. Int J Comput Integr Manuf 29(8):870–888

Nourmohammadi A, Zandieh M (2011) Assembly line balancing by a new multi-objective differential evolution algorithm based on TOPSIS. Int J Prod Res 49(10):2833–2855. https://doi.org/10.1080/00207540903473367

Ogan D, Azizoglu M (2015) A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. J Manuf Syst 36:46–54. https://doi.org/10.1016/j.jmsy.2015.02.007

Özcan U, Toklu B (2009a) Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models. Comput Oper Res 36(6):1955–1965. https://doi.org/10.1016/j.cor.2008.06.009

Özcan U, Toklu B (2009b) A tabu search algorithm for two-sided assembly line balancing. Int J Adv Manuf Technol 43(7–8):822. https://doi.org/10.1007/s00170-008-1753-5

Pattanaik LN, Jena A (2019) Tri-objective optimisation of mixed model reconfigurable assembly system for modular products. Int

J Comput Integr Manuf 32(1):72–82. https://doi.org/10.1080/0951192X.2018.1550673

Ponnambalam SG, Aravindan P, Naidu GM (2000) A multi-objective genetic algorithm for solving assembly line balancing problem. Int J Adv Manuf Technol 16(5):341–352. https://doi.org/10.1007/s001700050166

Rabbani M, Moghaddam M, Manavizadeh N (2012) Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout. Int J Adv Manuf Technol 59(9–12):1191–1210. https://doi.org/10.1007/s00170-011-3545-6

Roshani A, Fattahi P, Roshani A, Salehi M, Roshani A (2012) Cost-oriented two-sided assembly line balancing problem: a simulated annealing approach. Int J Comput Integr Manuf 25(8):689–715. https://doi.org/10.1080/0951192X.2012.664786

Saif U, Guan Z, Wang B, Mirza J, Huang S (2014) A survey on assembly lines and its types. Front Mech Eng 9(2):95–105. https://doi.org/10.1007/s11465-014-0302-1

Saif U, Guan Z, Zhang L, Mirza J, Lei Y (2017) Hybrid Pareto artificial bee colony algorithm for assembly line balancing with task time variations. Int J Comput Integr Manuf 30(2–3):255–270. https://doi.org/10.1080/0951192X.2016.1145802

Salehi M, Maleki HR, Niroomand S (2020) Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms. Neural Comput Appl 32(12):8217–8243

Sanei M, Mahmoodirad A, Niroomand S (2016) Two-stage supply chain network design problem with interval data. Int J e-Navigation and Maritime Econom 5:74–84

Scholl A, Klein R (1999) ULINO: optimally balancing U-shaped JIT assembly lines. Int J Prod Res 37(4):721–736. https://doi.org/10.1080/002075499191481

Suresh G, Sahu S (1994) Stochastic assembly line balancing using simulated annealing. Int J Prod Res 32(8):1801–1810. https://doi.org/10.1080/00207549408957042

Taassori M, Taassori M, Niroomand S, Vizvári B, Uysal S, Hadi-Vencheh A (2015) OPAIC: an optimization technique to improve energy consumption and performance in application specific network on chips. Measurement 74:208–220

Taguchi G (1986) Introduction to quality engineering: designing quality into products and processes (No. 658.562 T3).

Yavari M, Marvi M, Akbari AH (2019) Semi-permutation-based genetic algorithm for order acceptance and scheduling in two-stage assembly problem. Neural Comput Appl. https://doi.org/10.1007/s00521-019-04027-w

Yuan B, Zhang C, Shao X, Jiang Z (2015) An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. Comput Oper Res 53:32–41. https://doi.org/10.1016/j.cor.2014.07.011

Zhang W, Gen M (2011) An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. J Intell Manuf 22(3):367–378. https://doi.org/10.1007/s10845-009-0295-5

Zhou BH, Tan F (2018) Electric vehicle handling routing and battery swap station location optimisation for automotive assembly lines. Int J Comput Integr Manuf 31(10):978–991. https://doi.org/10.1080/0951192X.2018.1493229