International Journal on Interactive Design and Manufacturing (IJIDeM) (2024) 18:4209–4222
https://doi.org/10.1007/s12008-024-01929-5

ORIGINAL ARTICLE

Assembly line balancing with product and operator oriented sequence dependent task times using tabu search algorithm

R. G. Özdemir[1] · U. Kula[1] · M. Helal[1]

Received: 11 October 2023 / Accepted: 2 June 2024 / Published online: 1 July 2024

Abstract

The problem is formulated as a mixed-integer nonlinear programming model, with the objective of minimizing sequence-dependent setups while meeting specified cycle time and station count constraints. A tabu search algorithm is proposed to solve the resulting mathematical model, and its performance is assessed through numerical experiments. The developed numerical examples demonstrate significant improvements in setup time reduction. Notably, a substantial enhancement—a 91% reduction in setup time—is observed in moderate-sized problems ($n = 50$) characterized by high precedence density (OS = 0.6). Subsequently, the developed tabu search algorithm is applied to address the assembly line balancing problem encountered by a leading air conditioner manufacturer. Numerical experiments indicate that the algorithm yields an approximate 10% reduction in both product and operator-oriented setups, showing its effectiveness in enhancing assembly line efficiency. Future research could explore how the developed approach to be applied in various industrial settings including multi-model or mixed-model assembly lines.

Keywords Assembly-line balancing · Sequence dependent task times · Tabu search · Meta-heuristics

1 Introduction

An assembly line consists of a series of workstations arranged to produce a set of finished products. Workpieces are transferred from one station to another, and special operations are performed depending on the technological requirements of the product. Generally, assembly lines are balanced under a cycle time constraint by observing the precedence relationships between tasks. A precedence relationship between two tasks, i and j, means that task i must be finished before task j starts. Precedence relationships between tasks are usually derived from the technological constraints of assembly operations. However, some precedence relationships may not represent strict technological constraints in practice. These types of predecessors are added to precedence relationship diagrams to reduce assembly times or worker load. Therefore, they are soft constraints.

In this paper, an assembly line balancing problem faced by a leading air conditioner manufacturer located in Turkey is considered. In the air conditioner assembly process, the parts and components produced in the company's job shop are assembled in 21 different workstations. However, some assembly tasks defined in the precedence diagram can be assembled in a different sequence than prescribed. This provides flexibility to the assembly line workers to change the order of assembly tasks as needed, which at times increases the assembly cycle times.

In air conditioner assembly operations, several components need to be installed at the same location or at locations with close proximity on the main body. Installing, for example, component 2 before component 1 may prolong the latter task. This is because the existence of component 2 makes it harder for component 1 to be installed, as additional operations may be required or may prevent the worker from using the most efficient installation procedure. In such cases, the air conditioner manufacturer desires to have the component 1 task precede the component 2 task due to the interaction between component 1 and component 2 assembly times.

Interaction between tasks may be either product or operator-related. Product-related interaction refers to the increase in task times due to factors related to the product, such as product characteristics, shape, and its orientation

✉ R. G. Özdemir
rifat.ozdemir@aum.edu.kw

1 College of Engineering and Technology, American University of the Middle East, Kuwait City, Kuwait

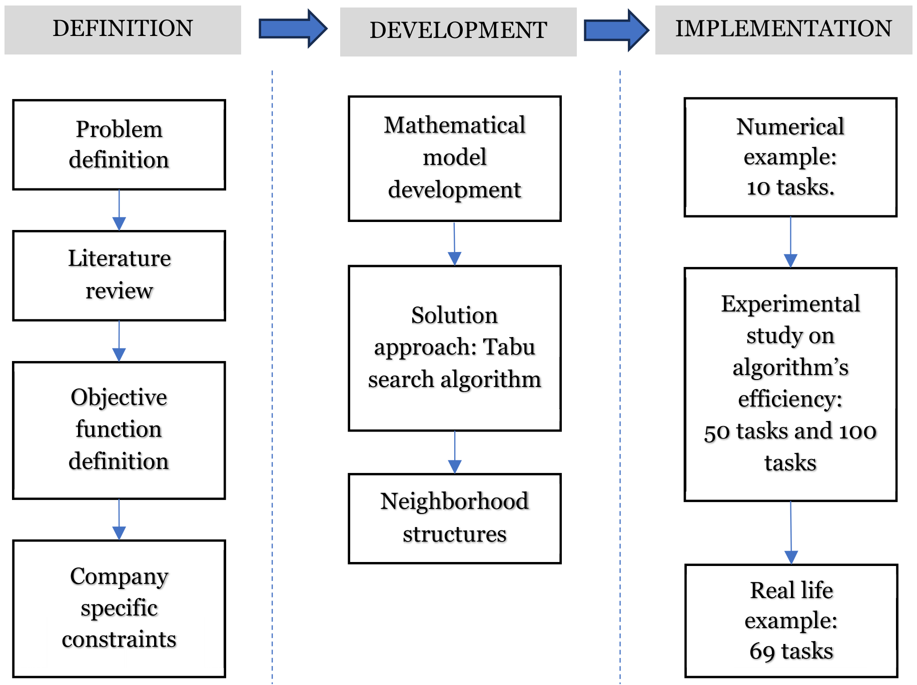
DEFINITION
DEVELOPMENT
IMPLEMENTATION
Problem definition
Literature review
Objective function definition
Company specific constraints
Mathematical model development
Solution approach: Tabu search algorithm
Neighborhood structures
Numerical example: 10 tasks.
Experimental study on algorithm's efficiency: 50 tasks and 100 tasks
Real life example: 69 tasks

Fig. 1 Workflow of the proposed study

during assembly, among others. On the other hand, operator-related interactions occur due to walking, material usage, and equipment preparation times. The study is summarized on a workflow diagram in Fig. 1.

In this paper, the extra task time caused by product characteristics and by operator preferences are distinguished and referred to as product-oriented setups (POS) and operator-oriented setups (OOS), respectively.

Figure 1a displays the precedence diagram, where solid lines indicate essential precedence relationships between tasks, while dashed lines indicate interactions among the tasks. For instance, task 1 is an immediate predecessor of tasks 2 and 3, while the connection between tasks 2 and 3 indicates that a setup activity exists. This setup activity occurs because additional effort is required if one of the tasks is performed earlier than the other in the assembly process.

Figure 2b shows two stations in an assembly line with a feasible task assignment based on the precedence diagram in (a). As Fig. 2b illustrates, tasks 1, 2, and 3 are assigned to Station 1, while the remaining tasks 4, 5, and 6 in the precedence diagram are assigned to Station 2. The arcs between tasks 2 and 3 in Fig. 2b represent the setup, i.e., extra times that depend on the sequence in which these tasks are performed. $\delta 23$ denotes the operator-related setup time, OOS, and $\theta 23$ denotes the required product-related setup time, POS. Thus, the duration of task 3 increases by the total sum of setup times $\delta 23$, and the operator-related 23 if task 2 is executed before task 3 in the station. s executed before task 3 in the station. However, setup times related to the operator do not occur between tasks assigned to different stations. Each operator is dedicated to their own station, eliminating the need for operators to alter the task sequence to minimize walking time, material usage, or tool changes. Therefore, only POS appears between task 2 and task 4, and the task time of task 4 increases by the operator-related 24 if task 2 is assigned to an earlier station, as shown in the figure. Product-related setups affect task time even if the interaction occurs between two distant tasks along the line because changing the order of tasks increases the task times. Because tasks are performed in a cyclic manner by an operator in a station, the last task in a station always interacts with the first task, resulting in an operator-related setup. This occurs because the operator needs to adjust the equipment, walk, or calibrate the tools to perform the first task in the station. Conversely, the initial and final tasks within a station never interact in terms of product-related setups, as these tasks are executed sequentially and independently.

2 Literature review

The literature on assembly balancing and the range of problems encountered are extensive. For a recent classification of assembly line balancing problems, types of assembly lines, and variations in problem types, see Eghtesadifard et al. [1]. Scholl et al. [2] were the first to introduce the concept of the sequence-dependent assembly line balancing problem (SDALBP) and discussed various solution approaches by adapting existing methods used for

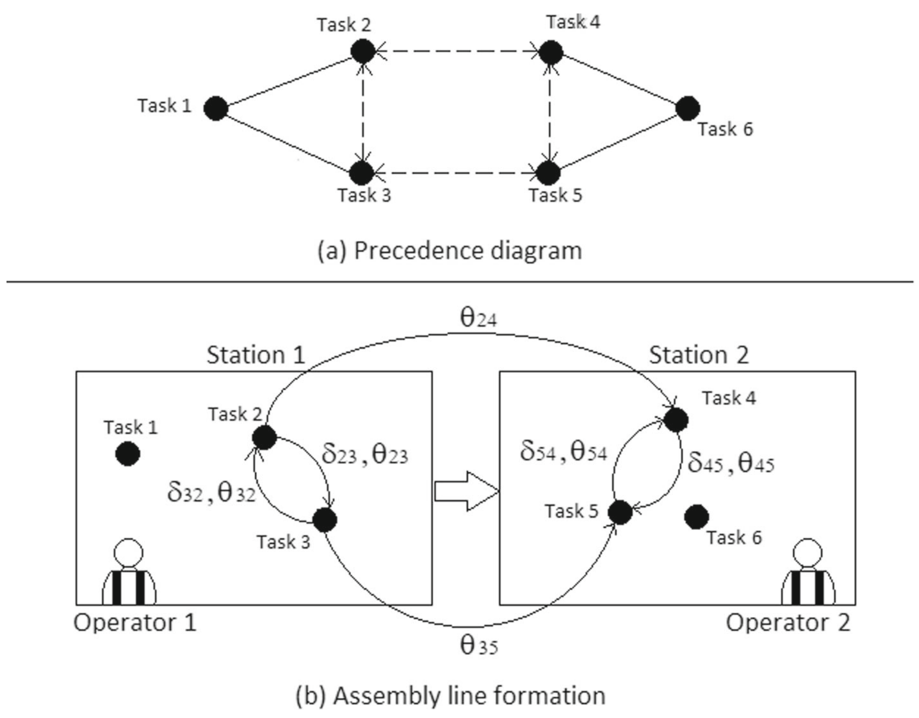

Fig. 2 An example of assembly line formation illustrating interaction between tasks due to product and operator-oriented setups

the Simple Assembly Line Balancing Problem (SALBP) to address the sequence-dependent assembly line balancing problem. They also conducted preliminary computational experiments. They demonstrated that SALBP-based search procedures are highly effective in solving the sequence-dependent assembly line balancing problem. For an early review of SALBP and its exact and heuristic solution methods, refer to Baybars [3] and Scholl and Becker [4]. In a more recent study, Dolgiu et al. [5] developed mathematical models for SALBP-1 and SALBP-2 and discussed the application of simulated annealing, tabu search, and genetic algorithms for both problem types.

In a very recent study, Boysen et al. [6] reviewed the developments in the SALB problem and its variants that have appeared in the literature over the last 15 years since the publication of [4]. They also suggested ways to collect data on precedence diagrams in real industrial settings and outlined a future research agenda.

Andrés et al. [7] were the first to consider sequence-dependent setups in a realistic setting. They observed that scheduling tasks assigned to a workstation by solving the Simple Assembly Line Balancing Problem was considered an operational issue in practice, needing to be resolved by the operator assigned to the station. They formulated a mathematical model that simultaneously considers task assignments to stations, intra-station decision-making and task scheduling, and inter-station decision-making. The authors named the problem the Generalized Assembly Line Balancing Problem with Setups (GALBPS), solved it using a metaheuristic called GRASP, and compared its results with various heuristic rules. Martino and Pastor [8] developed several priority-based heuristics to solve the problem defined by Andrés et al. [7]. They conducted a computational experiment comparing the suggested heuristics with the metaheuristic GRASP and some existing heuristics in the literature. Their results showed that some of the developed heuristics outperformed the existing ones, including the metaheuristic GRASP. Özcan and Toklu [9] studied a two-sided assembly line with sequence-dependent setups. They formulated the problem as a mixed-integer program to minimize the number of stations for a given cycle time and solved a set of test problems using a heuristic approach.

In another study, Giard and Jeunet [10] also considered sequence-dependent setups in a mixed-model car assembly line with zoning restrictions. They aimed to jointly minimize setup and temporary utility worker costs as their objective.

Akpinar and Baykasoğlu [11] studied an assembly line with sequence-dependent setups and formulated the problem as a mixed-integer program to solve the assembly line balancing problem with sequence-dependent tasks. Their mixed-integer programming formulation includes aspects of real-world assembly systems such as parallel workstations, zoning constraints, and sequence-dependent setup times between tasks. Since the problem defined in [11] is NP-complete, the same authors developed a metaheuristic based on ant colony optimization in Akpinar and Baykasoglu [12]. They addressed the sequence-dependent assembly line problem using a multiple colony bees algorithm and tested the performance of the algorithm for low, medium, and high variability setup times. Their computational results indicate that the multi-colony algorithm outperforms the single colony bees algorithm.

More recently, Özcan [13] considered line-switching setups occurring in a workstation in two parallel assembly lines and formulated the problem as a binary linear programming model, solving it using a simulated annealing algorithm. Yang and Cheng [14] addressed a similar problem involving forward and backward setups, developing a mixed-integer programming model and solving it using a neighboring search algorithm. Lopes et al. [15] proposed flexible frontiers to minimize the line length instead of commonly used fixed frontiers for multi-manned paced assembly lines. They developed lower bounds for a new formulation of a mixed-integer linear program and demonstrated that flexible frontiers may reduce assembly line length by 42%.

In a very recent study, Didden et al. [16] addressed a real-world assembly balancing problem in automobile assembly lines by considering all relevant factors, such as mixed-model lines, sequence-dependent setups, variable operator workplaces, and multiple assignment constraints. They solved the resulting model using a genetic algorithm to be utilized as a decision support system in a real-world setting. Results of their real-life case study showed that the proposed genetic algorithm increased the efficiency of the line and decreased the variability of operator times across all stations.

To the best of our knowledge, studies addressing sequence-dependent task times in the literature are categorized into two main groups: (i) studies considering setups that occur due to the sequence of operations applied to the base product; and (ii) studies considering setups that occur due to the sequence of tasks performed by the same operator. As a contribution to the field, this study proposes the simultaneous consideration and differentiation of these two types of setups to address the problem of sequence-dependent assembly line balancing. Such joint consideration results in a new mathematical formulation of the problem as a non-linear programming model, which incorporates product and operator-oriented setups. Our model distinguishes between product and operator-oriented setups, enabling the determination of the duration of each setup type. This differentiation may aid in focusing efforts separately on each setup type to enhance assembly line operations.

Tabu search (TS) was first proposed by Glover [17] and has since been successfully applied in a wide variety of combinatorial optimization problems, including assembly line balancing. Chiang [18] implemented tabu search for the Type I simple assembly line balancing problem and demonstrated its ability to identify the optimal solution across all test problems, with only a few exceptions. Pastor et al. [19] presented a real-life assembly line balancing problem involving multiple products and multiple objectives and developed a tabu search algorithm to improve commonly used heuristics in the literature. For applications of tabu search to other variations of assembly line balancing problems, see [20–22].

The paper is organized as follows: Sect. 2 defines and models the problem mathematically. Section 3 presents the proposed tabu search method to solve the problem. Following that, an illustrative example is provided, and the performance of the proposed algorithm is tested with generated data. In Sect. 4, the methodology is implemented in an air-conditioner producer firm. Finally, the study is concluded in Sect. 5.

3 Problem definition and mathematical formulation

This section presents the detailed explanation of the problem addressed in the study along with necessary assumptions and formulation. The objective of sequence dependent assembly line balancing problem is to assign n tasks to the minimum number of m stations with minimum endured setup times, which is the sum of product and operator-oriented setup times.

The notation used throughout the study is given below.

i and k	Task indexes.
j	Station index.
s	Position index.
n	Number of tasks.
m	Number of stations.
t_i	Operation time of task$i$.
c	Cycle time.
E_i, L_i	Earliest and latest workstation where task i can be assigned.
T_j	Set of all tasks can be assigned to workstation j.
$P^*{}_k$	Set of tasks (i, k) where i is immediate predecessor of k.
$F^*{}_k$	Set of all successors of task k
PT_i	Set of predecessors of task i, including non-immediate predecessors
SD^p	Set of interacting pairs in terms of product oriented setups.
SD^o	Set of interacting pairs in terms of operator oriented setups.
Nm_j	Maximum number of tasks that can be assigned to workstation j
θ_{ik}	product oriented setup times between tasks i and k
δ_{ik}	operator oriented setup times between tasks i and k

$$X_{ijs} = \left\{ \begin{array}{l} 1,\ \text{if task } i \text{ is assigned to station } j \text{ at } s^{th} \text{ position} \\ 0,\ \text{otherwise} \end{array} \right\},$$
$$\left(i = 1, \ldots n;\ j = E_i, \ldots, L_i;\ s = 1, \ldots Nm_j\right)$$

$$Y_j = \begin{cases} 1, \text{ if station } j \text{ is opened} \\ 0, \text{ otherwise} \end{cases}, \quad (j = 1, ..., m)$$

$$Z_{ikj} = \begin{cases} 1, \text{ if task } i \text{ is performed just before task } k \text{ in the station } j \\ 0, \text{ otherwise} \end{cases}, \\ \left(\forall j;\ \forall(i, k)|(i, k \in SD^o) \text{ and } (i, k \in T_j)\right)$$

$$N_{ik} = \begin{cases} 1, \text{ if task } i \text{ is performed before task } k \\ 0, \text{ otherwise} \end{cases}, \\ \left(\forall j;\ \forall(i, k)|(i, k \in SD^p)\right)$$

$$W_{ij} = \begin{cases} 1, \text{ if task } i \text{ is the last task assigned to station } j \\ 0, \text{ otherwise} \end{cases}, \\ (\forall i;\ j = E_i, ..., L_i)$$

All tasks are sequentially completed at the assembly stations according to a predetermined sequence. The position of task k in the sequence, determines product and operator oriented setup times endured by k^{th} task, PST_k and OST_k, respectively. PST_k is the sum of all product oriented setup times between task k and all others which appear before k^{th} task in the sequence:

$$PST_k = \sum_{\forall i, i \neq k} \theta_{ik} N_{ik}, \tag{1}$$

whereN_{ik} is the control variable which counts θ_{ik} in total product oriented setup time when task i comes before task k in the sequence. Note that, it is not necessary that the tasks i and k are in the same station or consecutive in the sequence.

OST_k is the sum of all operator oriented setup times between task k and the task performed just before the k^{th} task in the same station as given below.

$$OST_k = \sum_{\forall i, i \neq k, (i, k \in T_j)} \delta_{ik} Z_{ikj}, \tag{2}$$

whereZ_{ikj} is the control variable which counts δ_{ik} in total operator oriented setup time when task i comes just before task k in the sequence of performing the tasks of station j. Unlike product-oriented setups, it is imperative that tasks i and k are assigned to the same station and are consecutive in the sequence. Sequence of the tasks in the same station are performed cyclically which makes tasks i and k consecutive if i is the last task and k is the first task in the sequence.

The assumptions of the model are as follows:

i. Task times and set-up times are known with certainty.
ii. A single model of one product is assembled on the line.
iii. Buffers are not allowed between stations.
iv. All workstations are equally equipped and any task can be assigned to any workstation.
v. Parallel stations and workers are not allowed.
vi. Setup times are independent of the worker types.

In addition to above assumptions, the earliest and latest stations for task k are defined by considering product-oriented setup times as follows:

$$E_k = \lceil (t_k + \sum_{i \in P_k^*} (\theta_{ik} + t_i)/c \rceil \text{for } k = 1, ..., n \tag{3}$$

$$L_k = m + 1 - \lceil (t_k + \sum_{h \in F_k^*} (\theta_{ih} + t_i)/c \rceil \text{for } k = 1, ..., n \tag{4}$$

The Eqs. 3 and 4 are derived under the assumption that certain tasks interact in terms of both precedence relationships and product-oriented setups. That is, task k may be predecessor of task i and accomplishing task k prior to task i may increase the time required to perform task i due to product oriented setups. Assignable task set for each station, T_j is then determined based on the calculated earliest and latest stations of tasks as follows:

$$T_j = \{i | E_i \leq j \leq L_i\}, \forall j \tag{5}$$

The predetermined values of E_i, L_i and sets of T_j restrict the number of variables. The mathematical model is as follows:

Objective function

$$\min z = \sum_{j=1}^{m} c.Y_j + \sum_{i=1}^{n} (PST_i + OST_i) \tag{6}$$

Subject to

$$\sum_{j=E_i}^{L_i} \sum_{s=1}^{Nm_j} X_{ijs} = 1, \forall i \tag{7}$$

$$\sum_{\forall i \in T_j} X_{ijs} \leq 1, \forall j;\ s = 1, ..., Nm_j \tag{8}$$

$$\sum_{\forall i \in T_j} X_{ijs+1} \leq \sum_{\forall i \in T_j} Xijs, \forall j;\ s = 1, ..., Nm_j \tag{9}$$

$$R_i = \sum_{j=E_i}^{L_i} \sum_{s=1}^{Nm_j} \left[(Nm_j.(j-1) + s \right].X_{ijs},\ i \in T_j \tag{10}$$

$$R_i - R_k \leq 0,\ \ (i, k) \in P \tag{11}$$

$$R_i - R_k \leq M(1 - N_{ik}), \forall i, kL_i \geq E_k \tag{12}$$

$$N_{ik} \leq N_{ip} + N_{pk} \leq N_{ik} + 1, \forall i,\ p,\ k \in SD^p \tag{13}$$

$$X_{ijs} + X_{kj,s+1} \leq 1 + Z_{ikj},\ \forall j;\ s = 1,\ .....,\ Nm_j - 1;\quad \forall (i,\ k) | (i \neq k) \wedge (i,\ k \in T_j) \wedge (k \neq PT_i) \tag{14}$$

$$X_{ijs} - \sum_{\forall k\, \in T_j | (i \neq k) \wedge (k \notin PT_i)} X_{kj,s+1} \leq W_{ij},\quad \forall j;\ s = 1,\ .....,\ Nm_j - 1;\ \forall i \in T_j \tag{15}$$

$$W_{ij} + X_{kj1} \leq 1 + Z_{ikj},\ \forall j;\ \forall (i,\ k) | (i \ / = k) \wedge (i,\ k \in T_j) \wedge (i \notin PT_k) \tag{16}$$

$$OST_k = \sum_{(\forall i, (i,k) \in SD^o; i \in T_j)} \delta_{ik} Z_{ikj},\ \forall k \tag{17}$$

$$PST_k = \sum_{(\forall i, (i,k) \in SD^p)} \theta_{ik} N_{ik},\ \forall k \tag{18}$$

$$\sum_{\forall i \in T_j} \sum_{s=1}^{Nm_j} (t_i + PST_i + OST_i) X_{ijs} \leq c.Y_j\ ,\ \vee j \tag{19}$$

$$X_{ijs},\ Y_j,\ Z_{ikj},\ W_{ij},\ N_{ik} \in \{0,\ 1\}\ ,\ \vee i,\ j,\ k,\ s \tag{20}$$

$$R_i,\ PST_i,\ OST_i \geq 0,\ \ \vee i \tag{21}$$

The objective function (6) minimizes both the total time per cycle allocated to the stations and the total setup times incurred by all tasks allocated to the stations. It is worth noting that minimizing the total cycle time in each station is equivalent to minimizing the number of stations in the line.

Constraint set (7) implies that each task can only be assigned to only one position in one station. Constraint set (8) assures that only 1 task can be assigned to a position in a station. Constraint set (9) implies that tasks should be assigned in increasing order to positions within any workstation. Constraints sets (10) and (11) ensure that precedence relations between tasks are obeyed when assigning tasks to positions within stations. The rank of task i in the sequence of assignment is held by R_i values. If R_i value is smaller than R_k, then task i is assigned into an earlier position in the sequence than task k. Thus, constraint set (12) ensures that task i is prior to task k in the sequence of assignment if N_{ik} equals to 1, ($N_{ik} = 1 \rightarrow R_i \leq R_k$).

Constraint set (13), based on the formulation of Scholl et al. [1], can be explained as follows: Considering three, $i < p < k$, interacting tasks, in order not to trap into a cycle between those three tasks the following two cases should be avoided; $N_{ip} = N_{pk} = N_{ki} = 1$ and $N_{kp} = N_{pi} = N_{ik} = 1.$, which means that $N_{ip} + N_{pk} + N_{ki} \leq 2$, and can be reduced to $N_{ip} + N_{kp} \leq N_{ik} + 1$. The second case is avoided the same way as it is in the first case which is then transformed into $(1 - N_{ip}) + N_{ik} + (1 - N_{pk}) \leq 2 \rightarrow N_{ik} \leq N_{ip} + N_{pk}$. In constraint set (14), Z_{ikj} is 1 whenever task i is assigned to position s and task k is assigned to position $s + 1$ of station j, thus, Z_{ikj} can take a value of 1 only if the tasks are consecutive in the same station. Constraint set (15) forces the variable W_{ij} equal to 1 if i is the last task of station j. In constraint set (16), if i is the last task and k is the first task in station j, then they are considered to be consecutive, and thus Z_{ikj} is equal to 1. Constraint set (17) determines the total operator oriented setup times endured by task k. Constraint set (18) determines the total product oriented setup times endured by task k. Constraint set (19) ensures that the load of each station, consisting of task times, product-related, and operator-related setup times, remains below a predetermined cycle time dictated by the production rate of the line. Constraints (20) and (21) define the variables to be binary and nonnegative.

4 Proposed tabu search algorithm

Tabu search, first introduced and elucidated by Glover [17], and is a meta-heuristic neighborhood search methodology. The main idea of Tabu search is to search for a new candidate solution that lies in the neighborhood of the current solution. To overcome the issue of local optimality, Tabu search maintains a list of prohibited moves, called a tabu list, at each iteration of the search procedure.

More specifically, tabu search explores the search space beyond the local optimum by remembering a list of recent moves that should not be repeated when generating a new solution. The tabu restriction can be overridden if a forbidden (tabu) move meets aspiration criteria. These criteria are specifically designed to authorize a tabu move, ensuring that a potentially better solution that has not yet been visited is not overlooked.

A commonly used aspiration criterion in tabu search is that if, at any iteration of the search process, a tabu move can generate a solution that is better than the best solution found so far, the tabu restriction is revoked. The proposed methodology adopts the same criterion.

The objective function of the tabu search algorithm is defined as:

$$\max z = \sum_{j=1}^{m} \left(\sum_{i \in T_j} t_i \right)^2 - \left(\sum_{i=1}^{n} \sum_{k=1}^{n} (\theta_{ik}.N_{ik}) + \sum_{j=1}^{m} \left(\delta_{ik}.Z_{ikj} \right) + \delta_{L_j, F_j} \right)^2 \tag{22}$$

Table 1 gives the notation used in the pseudocodes used for single, double, and triple task change TS algorithms.

Table 1 Notation for tabu search objective function

j = workstation
i,k = index of tasks
m = number of stations
n = number of tasks
F_j = first task of station j
L_j = last task of station j
t_i = operation time of task$i$
T_j = set of tasks that are assigned station$j$
θ_{ik} = product oriented setup times between tasks$i$ and$k$
δ_{ik} = operator-oriented setup times between tasks$i$ and$k$
$Z_{ikj} = \begin{cases} 1, iftaskiisexecutedjustbeforetaskkatstationj \\ 0, otherwise \end{cases}$, for all $(i,k) \in SD$
$N_{ik} = \begin{cases} 1, iftaskiisexecutedpriortotaskk \\ 0, otherwise \end{cases}$, for all $(i,k) \in SD$

As seen above, the objective function aims to maximize the workload of the stations, in contrast to the global objective function, which seeks to minimize the number of workstations. Maximizing the workload of the stations ensures that they are optimally loaded, thus minimizing the number of stations required [18].

The algorithm employs a well-known heuristic called the “Largest Candidate Rule,” as demonstrated in Groover [22]. However, the method is modified accordingly. The algorithm starts by arranging the tasks in descending order based on their task time values to find an initial solution. It then selects the first available task that has the greatest task time and no precedence constraint. After assigning the first task, it searches for the next assignable tasks under the same conditions, while also considering setup times between tasks. In addition to the task time itself, it accounts for any setup time to determine the workstation time. If a task cannot be added to a station due to the cycle time limit, the algorithm opens a new workstation and continues until all tasks are assigned accordingly.

4.1 TS neighborhood structures

After the initialization process, the TS procedure begins with a single task change. Alternatives for every task, station, and position are explored. The algorithm attempts to change the station of a specific task while adhering to cycle time and precedence constraints. When a single task is assigned to another station, a virtual position is opened in that station. If the assignment is feasible, the objective value is calculated. After considering all possible positions of the station for task assignment, if the task cannot be moved, the process advances to the double exchange strategy. Table 2 depicts the algorithm for single task change.

After the single task change process, the tabu search procedure proceeds with double task change. In the double task change strategy, the algorithm explores alternatives for every possible change between two tasks. In addition to the single task change process, tasks can be assigned to the same station while considering double task change. When changing the positions of the two tasks is feasible with respect to cycle time and precedence constraints, the objective value of that change is calculated. Table 2 illustrates the algorithm for double task change (Table 3).

After completing the single and double task change processes, the algorithm proceeds to the triple task change process. In that procedure the algorithm tries alternatives for every possible change between three tasks. Two approaches can be adopted for the triple task change process, assuming three tasks (i, k and l) that can be potentially altered while adhering to both cycle time and precedence constraints. One change can occur if task i takes the place of task k, task k takes the place of task l and task l takes the place of task i. ($i \rightarrow k$), ($k \rightarrow l$), ($l \rightarrow i$). In addition, another change is also possible when task l takes the place of task k, task k moves to the place of task i and task i moves to the place of task l. ($l \rightarrow k$), ($k \rightarrow i$) and ($i \rightarrow l$). The algorithm of triple task change is given in Table 4.

4.2 Complementary TS mechanisms

After obtaining an initial solution, the entire solution is designated as both the current and best solution thus far. Subsequently, an initial value for the objective function is set to zero, followed by the execution of single, double, and triple exchanges. The solution that yields the maximum objective value is then designated as the current solution. After identifying the change that maximizes the objective value, the origin of the task, its position, and its station are added to the tabu list. If this change is already in the tabu list, its objective value is compared to the previous objective value. If the objective value with the selected change surpasses the previous best objective value, it is removed from the tabu list and designated as both the best assignment so far and the current assignment. If the change that was in the tabu list does not result in an objective value greater than the best solution so far, it is disregarded. Instead, the second maximum solution among task changes is selected. Conversely, if the change is not in the tabu list, it is designated as the current solution, and the algorithm proceeds from that task assignment without altering the best solution so far. The algorithm continues searching for changes until the stopping criterion is met, which is defined as the maximum allowed number of

Table 2 Pseudo code for single task change

```
1:  for task i=0,..., N do
2:     for station j=0,....., m do
3:          for position s=0,...,Nm_j do
4:             move task i as
                  Station number of task i→ j    for j≠ station number of task i
                  Position number of task i→ s
5:             if (all precedence relations are obeyed) then
6:                 calculate Station time(St_j) for all j as
                   $St_j = \sum_{i \in T_j} t_i + \sum_{k \in T_j} (\theta_{ik} N_{ik}) + \sum_{(i,k) \in T_j} (\delta_{ik} Z_{ikj}) + \delta_{L_j,F_j}$
7:                 if (St_j ≤ cycle time) for all j then
8:                    calculate (objective value) as
                      $\max z = \sum_{j=1}^{m} \left( \sum_{i \in T_j} t_i \right)^2 - \left( \sum_{i=1}^{n} \sum_{k=1}^{n} (\theta_{ik} . N_{ik}) + \sum_{j=1}^{m} (\delta_{ik} . Z_{ikj}) + \delta_{L_j,F_j} \right)^2$
9:             end for
10:    end for
11:  end for
```

Table 3 Pseudo code for double task change

```
1:  for task i=0,..., N-1 do
2:     for task k=0,....., N do
3:         change i and k positions
4:          if (all precedence relations are obeyed) then
5:              calculate St_j for all j as
                $St_j = \sum_{i \in T_j} t_i + \sum_{k \in T_j} (\theta_{ik} N_{ik}) + \sum_{(i,k) \in T_j} (\delta_{ik} Z_{ikj}) + \delta_{L_j,F_j}$
6:              if (St_j ≤ cycle time) for all j then
7:                 calculate (objective value) as
                   $\max z = \sum_{j=1}^{m} \left( \sum_{i \in T_j} t_i \right)^2 - \left( \sum_{i=1}^{n} \sum_{k=1}^{n} (\theta_{ik} . N_{ik}) + \sum_{j=1}^{m} (\delta_{ik} . Z_{ikj}) + \delta_{L_j,F_j} \right)^2$
8:     end for
9:  end for
```

Table 4 Pseudo code for triple task change

```
1:  for task i=0,..., N-2 do
2:     for task k=0,....., N-1 do
3:         for task l=0,...,N do
4:             change tasks i, k and l as
                  ( i →k , k →l, l → i ) and ( l →k , k →i, i → l )
5:              if (all precedence relations are obeyed) then
6:                  calculate St_j for all j as
                    $St_j = \sum_{i \in T_j} t_i + \sum_{k \in T_j} (\theta_{ik} N_{ik}) + \sum_{(i,k) \in T_j} (\delta_{ik} Z_{ikj}) + \delta_{L_j,F_j}$
7:                  if (St_j ≤ cycle time) for all j then
8:                     calculate (objective value) as
                       $\max z = \sum_{j=1}^{m} \left( \sum_{i \in T_j} t_i \right)^2 - \left( \sum_{i=1}^{n} \sum_{k=1}^{n} (\theta_{ik} . N_{ik}) + \sum_{j=1}^{m} (\delta_{ik} . Z_{ikj}) + \delta_{L_j,F_j} \right)^2$
9:          end for
10:    end for
11:  end for
```

iterations. Finally, the best solution so far is stored, and the algorithm terminates.

5 Numerical study and implementation

To evaluate the algorithm's performance, random sample problems were generated with pre-specified parameters. For testing purposes, the following example (illustrated in Figs. 3, 4, and 5) was developed, consisting of 10 tasks. In this scenario, a cycle time of 15 s is assumed.

Figure 3 shows the precedence diagram. Here, arrows represent precedence relations, the circles represent tasks and the rectangles represent task times of each task in seconds.

Figure 4 shows the product related setup times between tasks. The presence of an arrow indicates the existence of a setup between tasks, while the numbers in rectangles represent the setup times in seconds.

Figure 5 displays operator-oriented setup times, analogous to the product-oriented setups depicted in Fig. 4. However, these setups are only relevant when two tasks are assigned to the same station and executed consecutively. If tasks were assigned without considering the setup times, the initial solutions would be as follows:

As illustrated in Fig. 6, the algorithm successfully assigns task 10 to a single station, resulting in a total station time of nine seconds. However, a product-oriented setup time exists between tasks 5 and 10. Assigning task 5 earlier would consequently increase the station time. The assignment considering setup times is as follows:

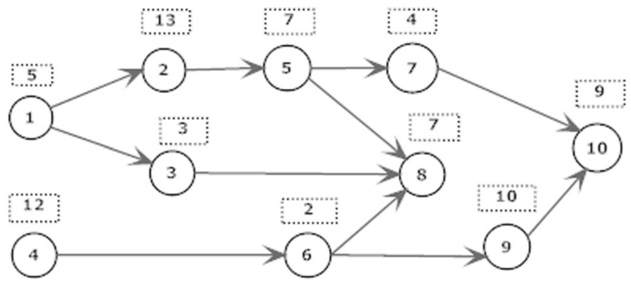

Fig. 3 Precedence diagram of the example problem

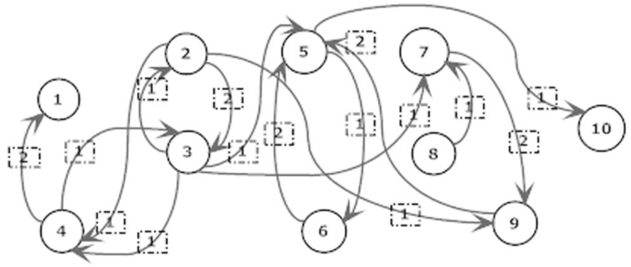

Fig. 4 Product oriented setup times (POS) of the example problem

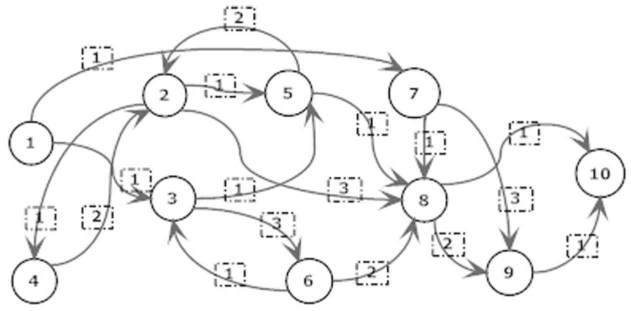

Fig. 5 Operator oriented setup times (OOS) of the example problem

Figure 7 shows that indeed there should be seven stations to assign the tasks with setups. Additionally, it is not feasible to assign tasks 1 and 9 to the same station (station two) due to a product-oriented setup time between tasks four and one (i.e., two seconds), which would exceed the cycle time limit in total. Therefore, another station is opened for task 1. Product-oriented setup times (POS) and operator-oriented setup times (OOS) are separately depicted in the initial solution. For instance, in station 6, the product-oriented setup time is two seconds, while the operator-oriented setup time is one second. The initial objective value, which is also set as the best so far value, is determined to be 750.

After finding an initial solution, neighborhood search strategies are performed. Single, double, and triple task moves are attempted, and the change that yields the best value is set as the best change so far. The algorithm first moves task 6 from position 2 of station 1 to first position of station 2 (6,1,2→6,2,1). The new objective value is 742. Double task change afterwards changes tasks 1 and 9 ((1,3,1→1,2,1)and(9,2,1→9,3,1)). The algorithm finally performs triple task change but fails to find one. The last task change results in an objective value of 781, which is better than the initial solution's objective value. Consequently, the best so far value is updated, and the origins of the tasks (i.e., (1,3,1) and (9,2,1)) are added to the tabu list. For each sample problem, the tabu list size was set to seven. The algorithm iterates until the stopping criterion is satisfied.

To the best of current knowledge, no existing sample problems in the literature feature such a distinction of setup times. Therefore, random examples have been generated. One characteristic that may influence the problem is the number of tasks (n), as it increases possible combinations among tasks.

Besides, another characteristic of the problem is precedence relations. The quantity and structure of these relations are important aspects to consider when generating the problems. Additionally, setup times are considered a specific characteristic of this problem type.

In the problem, the number of tasks and the order strength (OS) are utilized. The order strength is defined as the number of arcs in the precedence graph divided by $n(n-1)/2$ (where n is the number of tasks) as a complexity measures developed by Bukchin and Tzur [22]. Additionally, another measure

Fig. 6 Initial assignment of tasks without setup times by the Tabu search algorithm

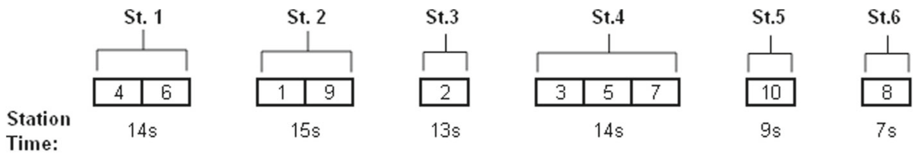

Fig. 7 Initial assignment of tasks with setup times by the Tabu search algorithm

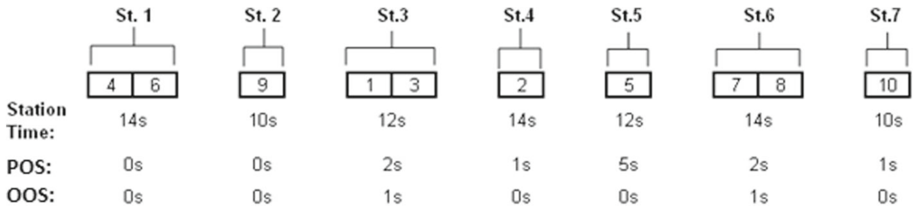

Table 5 Performance of the algorithm with sample problems

Task Number (n)	Order Strength (OS)	Setup Density (SD)	Initial Solution	Best So Far	Initial Setup	BSF Setup
50	0.3	0.2	61761	67795	556	441
		0.8	- 84367	- 69586	2182	1638
	0.6	0.2	72757	77262	386	32
		0.8	1734	6936	1318	124
100	0.3	0.2	47050	58295	2115	1990
		0.8	- 832186	- 791225	8435	8225
	0.6	0.2	89688	95345	1491	1421
		0.8	- 238986	- 224531	5122	5008

called setup density (SD) is defined as follows:

$$SD = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}}{n(n-1)} \tag{23}$$

where,

$$S_{ij} = \left\{ \begin{matrix} 1, if\theta_{ij} > 0and\delta_{ij} > 0 \\ 0, otherwise \end{matrix} \right\} \text{for}\forall i,\ j$$

Task times were randomly generated to range between 1 and 20 s, while setup times were randomly generated to range between 0 and 5 s. The algorithm was tested with different combinations of order strength (OS) and sequence-dependent (SD) setup times, as well as varying numbers of tasks (50 and 100). This resulted in eight combinations, with ten data points generated for each combination. The table below displays the average performances obtained from each group of ten samples.

As shown in Table 5, the algorithm successfully increases the objective solution and decreases setup times for each of the eight combinations. However, it was observed that when the setup density is high, the algorithm may yield negative solutions. These instances can be unrealistic, as total setup times may exceed total task times, especially since setup times are randomly generated between 0 and

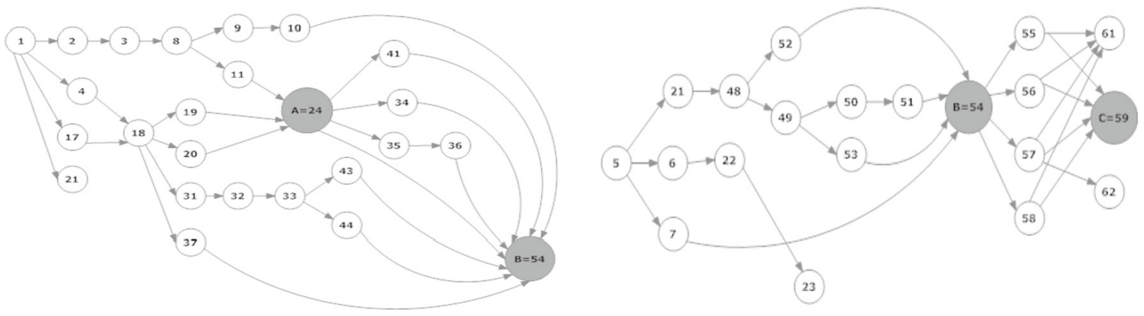

Fig. 8 (Left): Inital part of the precedence diagram of the air conditioner. (Right): Second part of the precedence diagram of the air conditioner

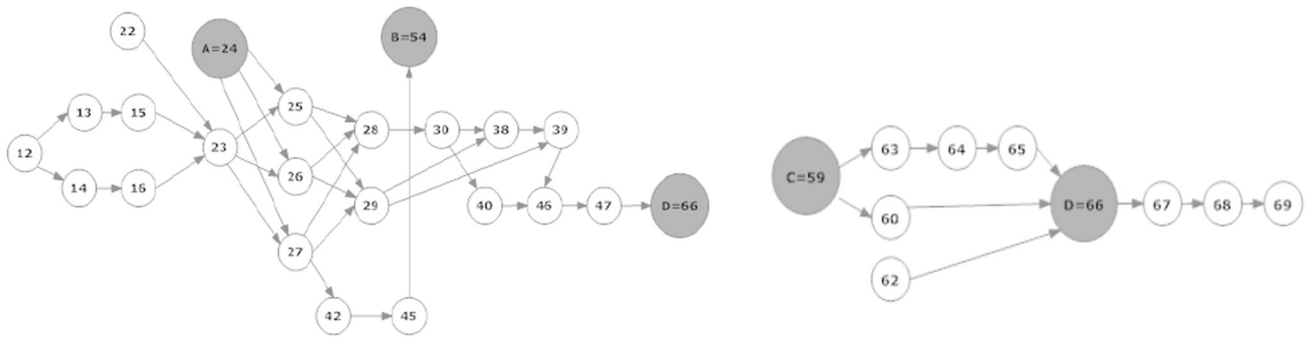

Fig. 9 (Left) Third part of the precedence diagram of the air conditioner. (Right): Last part of the precedence diagram of the air conditioner

Table 6 Product orinted setup times (secs.) matrix of the air conditioner assembly

	7	10	11	13	18	24	27	33	34	42	47	49	58
8	4.3												
9	4.3												
10	4.3		1.2			0.6							
11		1.2											
14				0.3									
24		3.3											
26									4.2				
31					4.5								
32					0.6								
34							4.1						
37										2.1			
42									0.5				
43								1.2					
48											3.5		
52												2.5	
57													2.2

5 s. Additionally, it is evident that after the TS algorithm is executed, the objective function significantly increases for some combinations due to the reduction in setup time usage. This reduction is primarily attributed to "operator-oriented setup times" occurring between consecutive tasks. The TS algorithm assigns tasks to minimize these times, aiming to generate substantial differences between the initial solutions and the best solution obtained thus far.

5.1 Case study

The air conditioner assembly process comprises 69 tasks. Due to its complexity, the assembly precedence diagram was divided into four separate parts. Tasks connecting each part were depicted in grey.

Figures 8 and 9 depict the precedence diagram for the assembly of the air conditioner at the manufacturer's facility. Grey circles in the precedence diagrams show the main components in the assembly process.

Table 6 shows the product-oriented setup times of the tasks given in the precedence diagram.

According to the precedence diagram task 8 is performed before task 7. However, due to the task characteristics it is possible to perform task 7 before task 8. Table 7 shows that changing the sequence of tasks 7 and 8 results in a 4.3 s increase in assembling tasks 7 and 8. A similar interaction exists between tasks 7 and 9. Table 7 shows task interactions due to the operator oriented setups.

Below are the results for the air conditioner's case. The algorithm has been executed 10,000 times with a tabu list size (TS) set to 7, and the closeness of the algorithm's result is compared to the theoretical maximum value.

Figure 10 shows the total workload of every station in the air conditioner firm where cycle time is set to 15 s. Total number of workstations is 21 and the total setup endured in that order is 27.5 s.

Figure 11 shows that rearranging the tasks, the algorithm has left the last station (station 21) empty, thereby reducing the total number of stations to 20. In comparison to the current situation in the AC manufacturer, the total setup time is reduced to 24.6 s from 27.5, representing approximately a 10% reduction in product and operator-related setup times.

6 Conclusion

In this study, a new Tabu Search (TS) algorithm was developed to address the Sequence Dependent Assembly Line Balancing problem with the aim of minimizing total setup time and consequently reducing the number of required workstations. Setup activities were categorized into product and operator-oriented tasks, and a modification to a well-known rule-based heuristic was proposed to generate initial solutions for the TS algorithm. Additionally, a novel neighborhood structure was introduced, incorporating single, double, and triple task exchange strategies to explore optimal solution alternatives. Furthermore, a mixed-integer nonlinear programming model was formulated to minimize newly defined product and operator-oriented setup times.

Table 7 Operator oriented setup times (secs.) matrix of the air-conditioner assembly

	6	7	9	10	11	13	14	18	19	20	21	22	23	24	28	29	33	34	35	39	40	43	44	49	50	51	55	56	57	58	60	61	62	63	67	69
6		0.5			0.5																															
7	0.3																																			
8			2.2		0.6																															
11				0.5																																
12						1.4	1.4																													
17								0.5																												
18									0.6	0.6																										
21												0.2																								
22											0.2																									
23														0.7																						
24													0.7					0.7																		
28																0.3																				
29															0.3																					
32																0.6																				
33																		0.7				0.5	0.5													
34																			0.1																	
39																					0.1															
40																				0.1																
43																							0.5													
48																								0.3												
49																									0.5											
50																											0.5	0.5	0.5	0.5		0.7				
54																																0.7				
55																																0.7				
56																																0.7				
57																																0.7				
58																												0.2								
59																															0.3					
60																																1.1				
61																																	0.2			
62																																		0.5		
63																																			0.1	
64																																				0.2

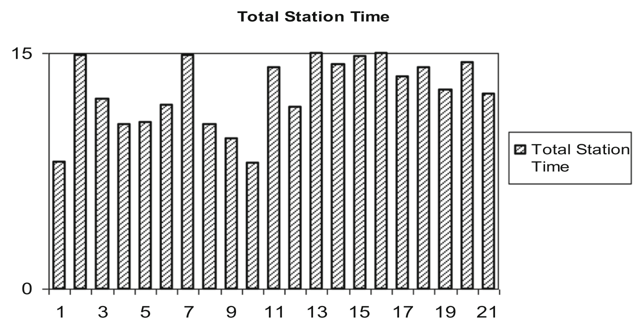

Fig. 10 Current station loads of the AC manufacturer

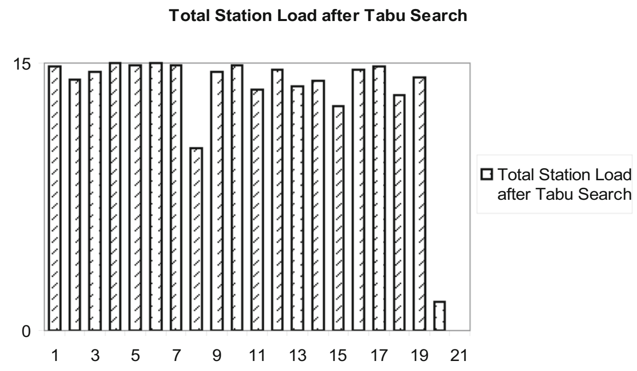

Fig. 11 Total Station Load in AC assembly after Tabu Search Algorithm Applied

The efficacy of the developed algorithm was assessed across assembly line balancing problems of various sizes, with differing precedence and setup density between tasks. The results demonstrate the robustness and efficiency of the algorithm, particularly for large-scale problems. Notably, the most significant improvement—a 91% reduction in setup time—was observed in moderate-sized problems ($n = 50$) with high precedence density (OS $= 0.6$).

To verify the practical applicability of our approach, we applied the developed algorithm to an air conditioner assembly line consisting of 69 tasks with minimal setup intensity between tasks. Although the algorithm effectively optimized the efficiency of the assembly line by reducing its length by one workstation, its performance in reducing setup time was relatively modest, achieving only a 10% reduction due to the low setup density.

Future research should prioritize the application of the developed algorithm across diverse industrial contexts in different assembly line systems, such as multi-model or mixed-model assembly lines.

Data availability The authors declare that no new data generated or analysed other than the data displayed in this manuscript.

Declarations

Conflict of interest The authors declare that they have no affiliations with or involvement in any organization or entity with any financial interest, or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Eghtesadifard, M., Khalifeh, M., Khorram, M.: A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017. Comput. Indus. Eng. **1**(139), 106182 (2020)
2. Scholl, A., Boysen, N., Fliedner, M.: The sequence-dependent assembly line balancing problem. OR Spectrum **30**, 579–609 (2008)
3. Baybars, I.: A survey of exact algorithms for the simple assembly line balancing problem. Manag. Sci. **32**(8), 909 (1986)
4. Scholl, A., Becker, C.: State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur. J. Oper. Res. **168**(3), 666–693 (2006)
5. Dolgui, A., Proth, J.M.: Design and Balancing of Paced Assembly Lines. In Supply Chain Engineering. Springer, London (2010)
6. Boysen, N., Schulze, P., Scholl, A.: Assembly line balancing: what happened in the last fifteen years? Eur. J. Oper. Res. **301**(3), 797–814 (2022)
7. Andres, C., Miralles, C., Pastor, R.: Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. Eur. J. Oper. Res. **187**, 1212–1223 (2008)
8. Martino, L., Pastor, R.: Heuristic procedures for solving the general assembly line balancing problem with setups. Int. J. Prod. Res. **48**(6), 1787–1804 (2010)
9. Özcan, U., Toklu, B.: Balancing two-sided assembly lines with sequence-dependent setup times. Int. J. Prod. Res. **48**(18), 5363–5383 (2010)
10. Giard, V., Jeunet, J.: Optimal sequencing of mixed models with sequence-dependent setups and utility workers on an assembly line. Int. J. Prod. Econ. **123**, 290–300 (2010)
11. Akpinar, Ş, Baykasoğlu, A.: Modeling and solving mixed-model assembly line balancing problem with setups. Part I: a mixed integer linear programming model. J. Manuf. Syst. **33–1**, 177–187 (2014)
12. Akpinar, Ş, Baykasoğlu, A.: Modeling and solving mixed-model assembly line balancing problem with setups. Part II: a multiple colony hybrid bees algorithm. J. Manuf. Syst. **33**(4), 445–461 (2014)
13. Özcan, U.: Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. Int. J. Prod. Econ. **213**, 81–96 (2019)
14. Yang, W., Cheng, W.: Modelling and solving mixed-model two-sided assembly line balancing problem with sequence-dependent setup time. Int. J. Prod. Res. **58**(21), 6638–6659 (2020)
15. Lopes, T.C., Pastre, G.V., Michels, A.S., Magatão, L.: Flexible multi-manned assembly line balancing problem: model, heuristic procedure, and lower bounds for line length minimization. Omega **1**(95), 102063 (2020)
16. Didden, J.B.H.C., Lefeber, E., Adan, I.J.B.F., Panhuijzen, I.W.F.: Genetic algorithm and decision support for assembly line balancing in the automotive industry. Int. J. Prod. Res. **61**(10), 3377–3395 (2023)
17. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)

18. Chiang, W.C.: The application of a Tabu search metaheuristic to the assembly line balancing problem. Ann. Oper. Res. **77**, 209–227 (1998)
19. Pastor, R., Andrés, C., Duran, A., Perez, M.: Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. J. Oper. Res. Soc. **53**, 1317–1323 (2002)
20. Lapierre, D.S., Ruiz, A., Soriano, P.: Balancing assembly lines with Tabu search. Eur. J. Oper. Res. **168**, 826–837 (2006)
21. Özcan, U., Toklu, B.: A Tabu search algorithm for two-sided assembly line balancing. Int. J. Adv. Manuf. Technol. **43**, 822–829 (2009)
22. Yilmaz, H., Yilmaz, M.: A mathematical model and tabu search algorithm for multi-manned assembly line balancing problems with assignment restrictions. Eng. Optim. **52**(5), 856–874 (2020)