Soft Computing (2021) 25:12899–12914
https://doi.org/10.1007/s00500-021-06107-3

OPTIMIZATION

Modeling and solving assembly line worker assignment and balancing problem with sequence-dependent setup times

Hamid Yilmaz[1]

Accepted: 31 July 2021 / Published online: 12 August 2021

Abstract
Assembly lines appear with various differentiations in order to better include the disabled in the labor market and to increase production efficiency. In this way, the optimal workforce assignment problem that emerges heterogeneously is called assembly line worker assignment and balancing problem (ALWABP). This paper addresses the ALWABP where the simple version is enriched by considering sequence-dependent setup times between tasks. A mixed integer linear programming model is presented, and a simulated annealing algorithm is developed such as an NP-hard problem. In order to test the proposed solutions, 640 benchmark problems in the literature were combined and used. The solutions obtained through using the proposed algorithm are compared with the mixed integer programming model on the small-size test problems. Experimental results show that the proposed algorithm is more effective and robust for a large set of benchmark problems.

Keywords Mixed integer linear programming · Assembly line balancing · Simulated annealing · Sequence-dependent setup times

1 Introduction

Assembly lines have been used extensively to produce high-volume products in the industry and one of the well-known research areas in the production environment (Serin et al. 2019). Assembly line balancing problem can be defined as assigning tasks to workstations, considering the constraints on the line. In addition, it has to be optimized with specific objectives, such as cycle time minimization for a predetermined number of the workstations or workstation minimization for a predetermined cycle time (Ct). Also, assembly processes include between 15 and 70% of manufacturing lead time and 40% of manufacturing costs (Gökçen et al. 2006; Yadav et al. 2019). A considerable amount of research on solving conventional ALB problems is available in the literature (Baykasoğlu and Dereli 2008; Çevikcan and Durmusoğlu 2020). In addition, conventional ALBP is known as the NP-hard problems (Ege et al. 2009; Yeh and Kao 2009). The detailed reviews of such studies are given by Scholl and Becker (2006) and Becker and Scholl (2006).

Another variant of ALBP considers worker assignments. This is a major problem found in sheltered work centers for disabled. Some employees may need more time to perform certain tasks or may not be able to do so. Because of the growing interest and respect for the disabled, many industries employing them, and companies are always concerned with assigning workers on assembly lines (Blum and Miralles 2011). In this case, one of the most critical issues is that there are workers with different skill levels. There are various models created for this purpose, and such lines are called assembly line worker assignment and balancing problems (ALWABP) in the literature. In cases where task times depend on the worker, it is important to decide on assigning tasks (Miralles et al. 2007). For this reason, a high-skill level worker can perform the task quickly, while the worker with a lower skill level could more time to complete the task. ALWABP tries to determine the most appropriate assignment for both workers and workstations as seen in Fig. 1 (Shin et al. 2019). ALWABP is an assembly line balancing problem involving worker

✉ Hamid Yilmaz
hamidyilmaz@bayburt.edu.tr

[1] Faculty of Engineering, Industrial Engineering Department, Bayburt University, Bayburt, Turkey

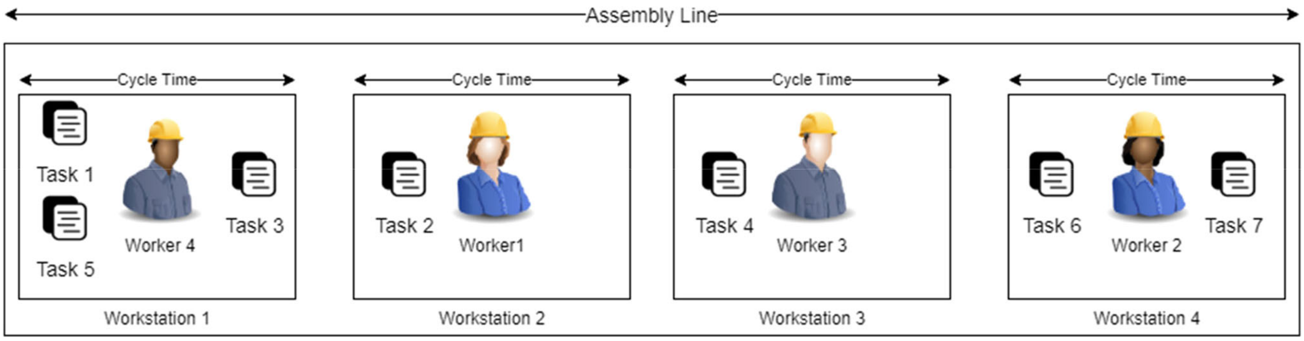

Fig. 1 Assignment example of workers and tasks for ALWABP

assignment and task allocation sub-problems. Worker assignment aims to distribute tasks evenly across workstations by satisfying different constraints such as precedence and cycle time constraint, while determining the assignment of workers to workstations (Dolgui et al. 2018).

In the real-world planning environment, more are occurring than the constraints involved in assembly lines. For this reason, it has become a necessity to offer solutions considering these additional constraints.

For most industrial assembly lines, setups are assumed to be negligible because their time is very short compared to task time. As a result of this situation, it is not necessary to specify task execution sequences in a workstation, but task execution order is an important issue in order to minimize station time in case of sequence-dependent setup times. Therefore, in the real-world planning environment, it may be necessary to consider sequence-dependent setup times in ALWBP. Performing one task directly before another task can affect the completion time of the next task within the same station, since a setup may be required to perform the second task. Also, if a task is assigned to a worker in a station as the last one, it may need setup time to perform the first task assigned to that station (Özcan and Toklu 2010). Andres et al. first introduced balancing and scheduling tasks in assembly lines with sequence-dependent setup times (Andres et al. 2008).

Most of the published papers about sequence-dependent setup times have focused extensively on various line balancing problems except ALWABP (Dolgui et al. 2018; Özcan and Toklu 2010; Andres et al. 2008; Scholl et al. 2008, 2013; Martino and Pastor 2010; Nazarian et al. 2010; Seyed-Alagheband et al. 2011; Yazdanparast et al. 2011; Yolmeh and Kianfar 2012; Kalayci and Gupta 2013; Akpınar et al. 2013, 2017; Hamta et al. 2013; Akpınar and Baykasoğlu 2014a, b; Şahin and Kellegöz 2017; Özcan 2019). ALWABP has been criticized for lacking real-life application since the problem structure in a real-world planning environment is much more complex. To the authors' best knowledge, no study has been reported on ALWABP with the objective of minimizing cycle time under sequence-dependent setup times. Building on the significance and relevance of the study to sequence-dependent setup times in ALWABP, this study presents several contributions, as follows:

(1) The first contribution is to present the ALWABP problem with sequence-dependent setup times for the first time. Also, a new mixed integer programming model is formulated and solved by GUROBI to solve small-sized problems.
(2) The second contribution is that a metaheuristic algorithm developed to solve the large-sized problems.
(3) A comprehensive comparative study is conducted to test the performance of the proposed algorithm. However, the result quality and CPU times of the proposed heuristic method were also compared with the mathematical model.

This paper is organized as follows. After the introduction, the ALWABP literature review is presented in Sect. 2. In Sect. 3, problem description and the MIP formulation of the problem are presented. In Sect. 4, the proposed simulated annealing algorithm for solving ALWABP is given in detail. The computational results with the application of MIP formulation and the results of the metaheuristic algorithm on a set of test problems are presented and discussed in Sect. 5 and 6. Finally, some conclusions and implications for further research are presented in the last section.

2 Literature review

In this section, firstly, the literature on assembly lines with setup time is given. Then, the literature on assembly line worker assignment and balancing problems are presented. Dolgui et al. reported that the study of assembly line balancing, scheduling and design problems is widely studied in the literature (Dolgui et al. 2018). However, there are various studies on assembly line problems with sequence-dependent setup times in the literature. Andres et al. first introduced balancing and scheduling tasks in assembly lines with sequence-dependent setup times. A binary integer programming model and greedy randomized adaptive search procedure are presented (Andres et al. 2008). Scholl et al. defined a new problem and formulated a mixed integer program for the sequence-dependent assembly line balancing problem. Also, they generated test data for computational experiments (Scholl et al. 2008). Martino and Pastor (2010) presented a heuristic to solve the same problem as Andres et al. (2008). Özcan and Toklu presented a mixed integer program and a COMSOAL-based heuristic algorithm to solve two-sided assembly line balancing problems with sequence-dependent setup times (Özcan and Toklu 2010). A mathematical formulation using mixed integer programming for sequence-dependent task times in multi-model production has been developed by Nazarian et al. (2010). Seyed-Alagheband et al. have developed a mathematical model and a simulated annealing (SA) algorithm to solve type II assembly line balancing problems with sequence-dependent setup times (Seyed-Alagheband et al. 2011). Yazdanparast et al. discussed the general assembly line balancing problem with setups and formulated a mathematical model to solve the problem (Yazdanparast et al. 2011). Yolmeh and Kianfar considered setup assembly line balancing and scheduling problem and proposed a hybrid genetic algorithm to solve the problem (Yolmeh and Kianfar 2012). Scholl et al. extended the problem by introducing backward setups. Also, they have proposed a mathematical program and heuristics algorithms in order to solve the problem (Scholl et al. 2013). Kalayci and Gupta considered sequence-dependent disassembly line balancing problem. However, a particle swarm optimization algorithm has been proposed to solve the problem (Kalayci and Gupta 2013). Akpınar et al. have presented a new hybrid ant colony algorithm with genetic algorithm for mixed-model assembly line balancing problem type I with real-world constraints such as zoning constraints, parallel workstations and sequence-dependent setup times between tasks (Akpınar et al. 2013). Hamta et al. have presented multi-objective optimization of single model assembly line balancing problem with sequence-dependent setup time exists between tasks and have proposed a particle swarm optimization algorithm with variable neighborhood search to solve it (Hamta et al. 2013). Akpınar and Baykasoğlu have presented a mixed-model assembly line balancing problem with sequence-dependent setup times. Besides, a mixed integer linear programming formulation and hybrid bee algorithm have been proposed to solve the problem (Akpınar and Baykasoğlu 2014a, b). Şahin and Kellegöz have developed a mixed-binary integer programming model, a simulated annealing algorithm and a genetic algorithm for the U-type assembly lines with sequence-dependent setup times (Şahin and Kellegöz 2017). Akpınar et al. have proposed an exact solution algorithm based on Benders decomposition for setup assembly line balancing problem (Akpınar et al. 2017). Özcan have introduced parallel assembly line balancing problem with sequence-dependent setup times. Also, they have proposed mathematical programming model and a simulated annealing algorithm to solve the problem (Özcan 2019). Yang and Cheng have examined the multi-manned assembly line balancing problem, which is widely used in large-sized productions, with sequence-dependent setup times. A mixed integer programming is presented, and a simulated annealing algorithm is also proposed to solve it (Yang and Cheng 2020).

Regarding worker assignment and balancing problems, Miralles et al. have introduced sheltered work centers in assembly line balancing and formulated a mathematical model (Miralles et al. 2007). Chaves et al. have proposed a hybrid heuristic based on clustering search to solve ALWABP (Chaves et al. 2007). Miralles et al. have defined a mathematical model for ALWABP and a branch and bound approach with different search strategies to solve the problem (Miralles et al. 2008). Chaves et al. have proposed a hybrid method based on clustering search to solve the ALWABP (Chaves et al. 2009). Blum and Miralles have developed a beam search method to solve assembly worker assignment and balancing problems (Blum and Miralles 2011). Moreira et al. have introduced a constructive heuristic based on priority rules of task-worker (Moreira et al. 2012). Mutlu et al. have developed a genetic algorithm to solve the problem. Also, results of the algorithm show that the proposed method is very robust and effective for large test instances (Mutlu et al. 2013). Vila and Pereira derived new lower bounds for the ALWABP. Nevertheless, they have developed an exact algorithm to solve the problem. This proposed branch and bound algorithm yields state-of-the-art results for ALWABP (Vila and Pereira 2014). Ramezanian and Ezzatpanah presented a mixed-model assembly line balancing and worker assignment problem. A goal programming solution procedure has been proposed to solve the problem (Ramezanian and Ezzatpanah 2015). Akyol and Baykasoğlu proposed a constructive heuristic approach for the ALWABP. Performance of the algorithm is compared with the relevant literature on test instances. Experimental results show that the proposed approach is very effective on test instances (Akyol and Baykasoğlu 2016). Janardhanan et al. studied worker assignment and line balancing problem in two-sided assembly lines. They have developed a mixed integer programming model and a migrating birds algorithm to solve the problem (Janardhanan et al. 2019).

In order to adapt the solutions offered to assembly lines to real-world systems, the problem should be examined together with the constraints such as sequence-dependent setup times encountered in the real world. When the literature is examined, it is seen that the ALWABP has not been studied together with the sequence-dependent setup time constraints. The presented study is the first study in the literature in terms of analyzing the assembly line worker assignment and balancing problem with sequence-dependent setup times (ALWABPS). Also, in this study, two main elements of setup times are considered between tasks: forward and backward setups, and the objective is to minimize the cycle time for workstations. Another contribution to the literature of the paper is to formulate a new mixed integer programming model. Furthermore, a computational study using a simulated annealing (SA) algorithm as a metaheuristic approach for test instances, involving up to 2 setup group (low and high) and total 640 test instances, is presented.

3 Assembly line worker assignment and balancing problem with sequence-dependent setup times (ALWABPS)

3.1 Problem definition

This paper will focus on proposing algorithm and approaches for solving ALWABP with the sequence-dependent setup time, based on conclusions from the literature review. This paper tackles the problem by aiming to minimize the cycle time and integrate forward and backward setup times into the problem. ALWABP, which minimizes the cycle time, can be formally expressed as: A set "I" of tasks, a set "W" of workers, and a set "WS" of stations are given by a line designer to perform these tasks. Some workers are unable to perform a number of tasks because they do not have some skills or they have a disability. Let "$W_i \subseteq W$" be the subset of workers that can perform task "$i \in I$". However, let "$I_w \subseteq I$" be the subset of tasks that can be performed by worker "$w \in W$". For each task "i" that can be performed by worker "w", "Z_{iw}" expresses the time required by the worker "w" to perform the task "i" and it is called the task time. Some tasks have precedence relationships where the current task cannot be performed before the previous one. The objective of the problem is to ensure that all work pieces (tasks) assigned to the workstation can be performed by the workers at that station, precedence relations between tasks are fulfilled and the cycle time of workstations is minimized (Pereira 2018).

Between tasks, there are two types of sequence-dependent setup times. These are forward setup and backward setup, respectively. At the same workstation, if task "i" is performed just before another task "h", a forward setup occurs for the same workpiece to perform task "h". It is called forward setup time, and "μ_{ih}" is added to the workstation time. In a regular workstation, if task "i" is the last assigned task in a workstation and task "h" is the first assigned task at the same workstation, then it is called backward setup. It is necessary to perform task h, and a backward setup time, t_{ih}, is added to compute workstation cycle time.

A detailed illustration of assembly line worker assignment and balancing with sequence-dependent setup times is given in Fig. 1. As seen in Fig. 2, Worker 3 is assigned to Workstation 1 and "1–5–3" tasks are assigned to him. At the same time, these assignments are "2–4" tasks for the Worker 1, and "6–7" for the Worker 2. Setup times, i.e., forward setups; μ_{15} and μ_{24}, and backward setups; t_{31} and t_{42}, are also shown in Fig. 2.

In this paper, it is assumed that the problem considered under the following conditions:

- The task times are deterministic and known in advance.
- Not all workers may be able to perform all tasks.
- A task can be assigned to only one workstation.
- A task can be assigned to only one worker.
- A single model product is produced.
- The precedence relationships are known in advance.
- All components are available with no quality problems.
- The forward and backward setup times deterministic and known in advance.
- Each task must be performed.
- No breakdowns occur in the assembly line.

4 Notations

The notations used in MILP of the problem are given as follows.

Indıces

i, h	A task
j	A workstation
w	A worker
s	A position inside the task operation sequence of a workstation

Parameters and sets

c	Cycle time
t_{iw}	Task time of "i" at worker "w"
SN	Maximum number of tasks that can be assigned to any station
P	Set of immediate predecessors of tasks
WS	Number of workstations

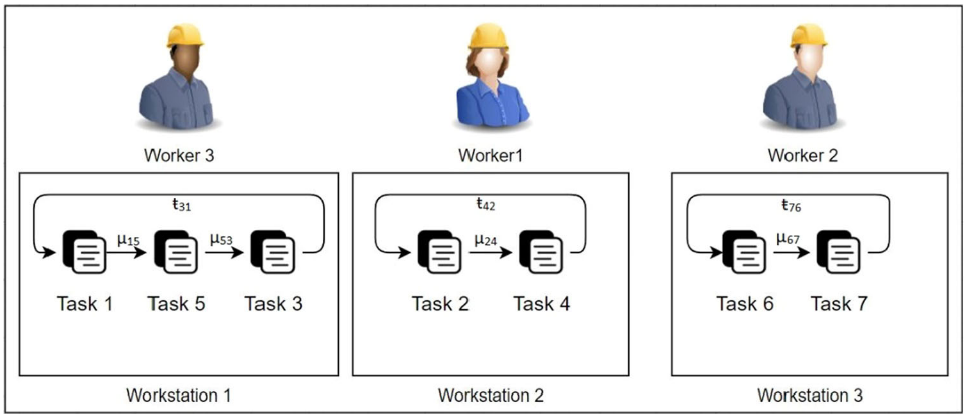

Fig. 2 Illustration of ALWABP with sequence-dependent setup times

WW Number of workers
SN Number of positions.
T Number of tasks
I Set of tasks. $I = \{1, 2, …., i, …T\}$.
W Number of assignable workers on the line. $W = \{1, 2, …., w, …WW\}$.
J Number of workstations on the line. $J = \{1, 2, …., j, …WS\}$.
S Number of positions in a station. $S = \{1, 2, …., s, … SN\}$.
M A big number.

Decision variables

x_{ijws} 1, if task "i" in workstation "j" is assigned to worker "w" in position "s" of its operation sequence; 0, otherwise.
y_{jw} 1, if worker "w" is assigned to workstation "w"; 0, otherwise.
pf_{ihjw} 1, if task "i" is the immediate processor of task "h" in workstation "j" at worker "w"; 0, otherwise.
pb_{ihjw} 1, if task "i" is the last task and "h" is the first task assigned to workstation "j" at worker "w"; 0, otherwise.

l_{ijw} 1, if task is the last task assigned to workstation "j" at worker "w"; 0, otherwise.

4.1 Mixed integer programming model of ALWABPS

In this study, presented mathematical model is as follows:

$$\text{Minimize}\, c \tag{1}$$

$$\sum_{j\in J}\sum_{w\in W}\sum_{s\in S} x_{ijws} = 1, \forall(i) \in I \tag{2}$$

$$\sum_{i\in I} x_{ijws} \le 1, \forall(j) \in J, \forall(w) \in W, \forall(s) \in S \tag{3}$$

$$\sum_{j\in J} y_{jw} = 1, \forall(w) \in W \tag{4}$$

$$\sum_{w\in W} y_{jw} = 1, \forall(j) \in J \tag{5}$$

$$\sum_{i\in I}\sum_{s\in S} x_{ijws} \le M * y_{jw}, \forall(j) \in J, \forall(w) \in W \tag{6}$$

$$\sum_{i\in I} x_{ijw(s+1)} - \sum_{i\in I} x_{ijws} \le 0, \forall(j) \in J, \forall(w) \in W, \forall(s) \in \{1, 2, \ldots\ldots\ldots(SN-1)\} \tag{7}$$

$$\sum_{j\in J}\sum_{w\in W}\sum_{s\in S}(SN*(j-1)+s)*x_{ijws} - \sum_{j\in J}\sum_{w\in W}\sum_{s\in S}(SN*(j-1)+s)*x_{hjws} \le 0, \forall(j) \in J, \forall(w) \in W, \forall(s) \in S, (i,h) \in P \tag{8}$$

$$\sum_{i\in I}\sum_{s\in S}\sum_{w\in W} t_{iw} * x_{ijws} + \sum_{(i,h)\in I, i\ne k}\sum_{s\in S}\sum_{w\in W} tsf_{ih} * pf_{ihjw} + \sum_{(i,h)\in I}\sum_{s\in S}\sum_{w\in W} tsb_{ih} * pb_{ihjw} \le c * \sum_{w\in W} y_{jw}, \forall(j) \in J \tag{9}$$

$$x_{ijws} + x_{hjw(s+1)} - pf_{ihjw} \le 1, \forall(j) \in J, \forall(i,h) \in I, i \ne h, \forall(s) \in \{1, 2, \ldots\ldots\ldots(SN-1)\} \tag{10}$$

$$x_{ijws} - \sum_{h\in I}\sum_{w\in W} x_{hjw(s+1)} \le l_{ijw}, \forall(j) \in J, \forall(i,h) \in I, i \ne h, \forall(s) \in \{1, 2, \ldots\ldots\ldots(SN-1)\} \tag{11}$$

$$l_{ijw} + x_{hjw1} - pb_{ihjw} \le 1, \forall(j) \in J, \forall(i,h) \in I, \forall(s) \in S \tag{12}$$

$$\sum_{w \in W} y_{(j+1)w} - \sum_{w \in W} y_{jw} \leq 0, \forall (j) \in \{1, 2, \ldots .(J-1)\} \quad (13)$$

$$x_{ijws} \in \{0,1\}, \forall (i) \in I, \forall (j) \in J, \forall (w) \in W, \forall (s) \in S \quad (14)$$

$$y_{jw} \in \{0,1\}, \forall (j) \in J, \forall (w) \in W \quad (15)$$

$$pf_{ihjw} \in \{0,1\}, \forall (i,h) \in I, \forall (j) \in J, \forall (w) \in W \quad (16)$$

$$pb_{ihjw} \in \{0,1\}, \forall (i,h) \in I, \forall (j) \in J, \forall (w) \in W \quad (17)$$

$$l_{ijw} \in \{0,1\}, \forall (i) \in I, \forall (j) \in J, \forall (w) \in W \quad (18)$$

$$c \geq 0 \quad (19)$$

The objective of the model is to minimize the cycle time of workstations of the assembly line (1). Constraints 2 ensures that each task is assigned to a worker in a workstation to a position. Constraint 3 ensures that not more than one task is assigned to a position in a workstation. Constraints 4 and 5 ensure that each worker is assigned to a single workstation and a workstation can only have one worker. Constraint 6 checks whether a station has been opened. The constraint 7 allows the task positions (time period) to be opened sequentially. Constraint 8 ensures that all precedence relations among tasks of all assembly lines are satisfied, regarding not only the assignment to different workstations, but also the time period inside the same workstations. Constraint 9 ensures that the workstation global time in every workstation, including forward setup times, backward setup times and processing times of tasks, is under the cycle time. If "i" and "h" tasks are assigned to "s" and "$s+1$" positions, respectively, in workstation "j" at worker "w", it is ensured that pf_{ihjw} takes the value of 1 by constraint 10. If the task "i" is the last task at worker "w" in workstation "j", l_{ijw} takes the value "1" by constraint 11. If task "i" is the last job in workstation "j" at worker "w" and job "h" is the first job in the same station, then pb_{ihjw} gets the value "1" by constraint 12. The constraint 13 allows the stations to be opened sequentially. Constraints 14–19 are variables.

5 Proposed algorithm

Simulated annealing is an iterative random search algorithm, and it has been used in many optimization problems including ALBP (Li et al. 2016; Roshani and Giglio 2017). Originally, simulated annealing was introduced as an optimization method by Kirkpatrick et al. (1983). In this paper, a simulated annealing algorithm is proposed to ALWABPS that is able to solve problem sizes of real-world environment.

The proposed simulated annealing algorithm is described by the following procedure: SA is initialized with predetermined parameter settings such as initial temperature "T_i", cooling rate "Cr", the number of iterations for each temperature level (TL), and the stop criteria as finish temperature "T_f". SA starts with an initial solution "S_i" with the cost, "CO_i". "S_i" can be obtained with a constructive heuristic using specific or random priority rules for the problem. "CO_i" is the objective function value for the solution of "S_i". In the case, "S_i" value is assigned to the current solution "CS" and the best solution "BS". The cost of "CS" and "BS" is calculated as "CO_c" and "CO_b", respectively. SA iterations begin with obtaining the initial solution by a constructive heuristic. Using a move operator, a neighbor solution "NS" is generated. The neighbor solution cost "CO_{NS}" is calculated and compared to "CO_c". Afterward, the changes in the objective function values are calculated by "$\Delta CO = CO_{NS} - CO_c$". If "$CO_{NS}$" is better than "$CO_c$", then "NS" is assigned to "CS". If "CO_{NS}" is worse than "CO_c", there are two different conditions called as the Metropolis criterion. The first condition is accepting "NS" as "CS" with the probability of $e^{-\Delta CO/\mathrm{temp}}$ ("T_c" is the current temperature and initially "T_c" is equal to "T_i"). The second condition is that "CS" remains unchanged, if the first condition is not met. Thus, accepting worse solutions by the algorithm increases the capability of jumping out of local optima. If "CO_c" is better than "CO_b", then "CS" is accepted as "BS"; otherwise, "BS" remains unchanged. The algorithm repeats this process "TL" times at each temperature level. The parameter "T_c" is slowly decreased as in "$T_c = T_c \cdot Cr$" by a cooling function until the stopping condition "$T_c < T_f$" is met. The algorithm will run, until the stopping criteria are satisfied.

Initial solution: In this paper, a worker-oriented strategy to get a feasible solution is presented. The workers are assigned to the workstations. Then, tasks are assigned to workers sequentially, taking the priority value of workers and tasks into consideration. In order to explain the initial solution generation is as follows:

1. Generate a random number between 0 and 1 by uniform distribution for each worker and task for assembly line.
2. Select the workers from highest priority to lowest and assign them to the workstations.
3. Determine the set of assignable tasks for the current workstation $j = 1$.
4. Sort the tasks in decreasing (priority rule) order.
5. Select the assignable first task "i", then assign the task to workstation "j". Otherwise close the workstation "j" and select the next workstation as $j = j + 1$. If workstation is the last station, go to step 6; otherwise, go to Step 4.

6. If all tasks are assigned to workers, then stop and save the solution as the initial solution. Otherwise, change worker priorities in turn and go to Step 2.

Neighbor generation: New line balance is created by reassign tasks to workers in workstations by using a move operator. In this algorithm, a swap operator is used. The operator ensures that the priority values of the two selected tasks are changed as seen in Fig. 3. This operator has been applied to both tasks and workers. A new neighbor solution is created from the current solution by using the swap operator in each step of the algorithm.

Objective function: Maximum cycle time duration created in workstations on assembly line worker assignment and balancing problem is used as the objective function to evaluate the quality of solutions generated. The objective of the proposed algorithm for the ALWABPS is minimizing the cycle time for a given number of workers.

Stopping criteria: The proposed algorithm terminates when the T_f or maximum iteration number ($Iter_{max}$) is obtained.

6 Numerical experiment

The main objective of this study is to introduce and characterize the assembly worker assignment and balancing problem with sequence-dependent setups. With this design, a new mathematical model is presented for the problem. As a solution approach, a metaheuristic approach is proposed. Proposed algorithm is a computationally simple approach, and within a short computational time, it can find good solutions. In this paper, the problem of assembly line worker assignment and balancing problem with sequence-dependent setup times is presented for the first time. There is no study on the problem for comparison with the presented algorithm in the literature. Hence, the results of the proposed simulated annealing algorithm are compared with the presented mathematical model solution on small-size test instances, and solutions of large test instances are enclosed in this paper.

The proposed mathematical model is solved by GUROBI 8.1.1; algorithm is coded by C# and executed on a PC with Intel(R) Core(TM) i7, 2.20 GHz processor and 8.00 GB of RAM. Benchmark data are considered from the ALWABP benchmark data set (Chaves et al. 2007). Test samples are derived through using five experimental factors with low and high levels. These factors are; the number of workers, the number of tasks, the variability of the task execution time, the order strength and the number of tasks that cannot be performed by workers. The details of test instance characteristics are given in Table 1. There are 320 test instances which are named into Heskia, Roszieg, Tonge and Wee-Mag.

Each one of the test instance families contains 80 problems. There are 32 task groups and each of them contains 10 test instances. Each task group is defined by the family name and a number between 1 and 8. However, the results of 80 test problems are given in detail for each

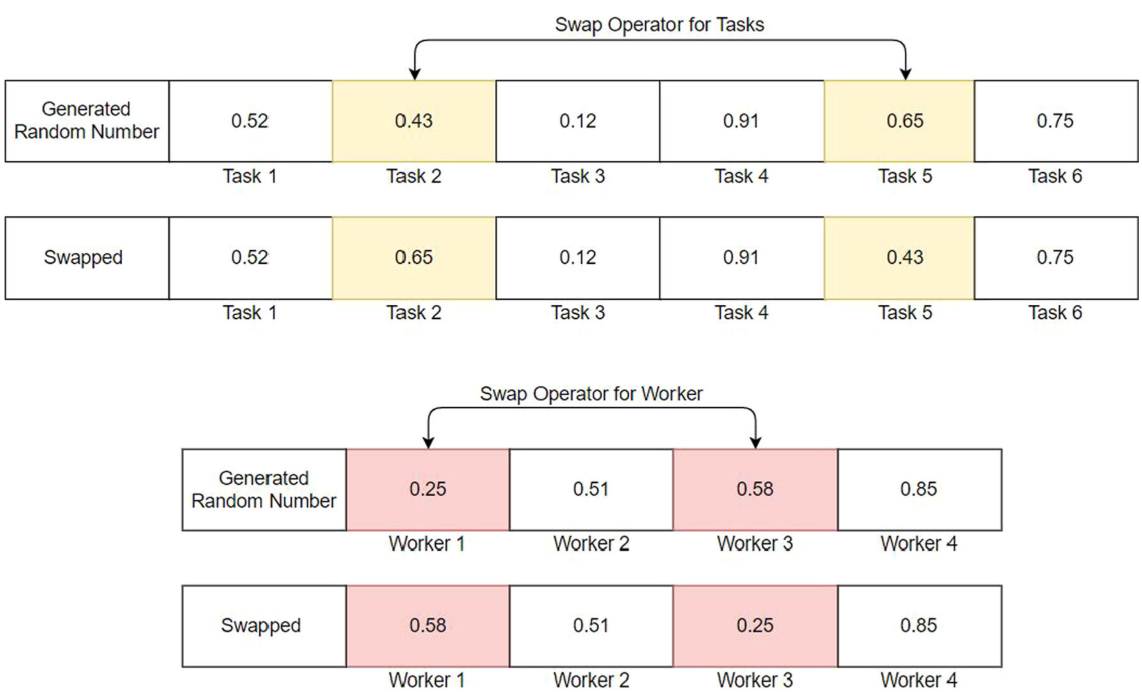

Fig. 3 Swap operator for the worker and tasks

Table 1 Characteristics of test instances

Family	Number of tasks	Number of workers	Order strength
Roszieg	25 (low)	Group 1–4 (4 Workers)/Group 5–8 (6 Workers)	71.67 (high)
Heskia	28 (low)	Group 1–4 (4 Workers)/Group 5–8 (7 Workers)	22.49 (low)
Tonge	70 (high)	Group 1–4 (10 Workers)/Group 5–8 (17 Workers)	59.42 (high)
Wee-Mag	75 (high)	Group 1–4 (11 Workers)/Group 5–8 (19 Workers)	22.67 (low)

Table 2 Characteristics of test instances for setup times

Family	Test instance code	Setup time level (low)	Test instance code	Setup time level (high)
Roszieg	heskia_c = 138.alb	0.25	heskia_c = 138.alb	1.00
Heskia	roszieg_c = 16.alb	0.25	roszieg_c = 16.alb	1.00
Tonge	Tonge70_c = 160.alb	0.25	Tonge70_c = 168.alb	1.00
Wee-Mag	wee-mag_c = 38.alb	0.25	wee-mag_c = 38.alb	1.00

family. In addition, the test instances presented by Scholl in the literature were used in terms of sequence-dependent setup times. Scholl grouped the sequence-dependent setup times of tasks into 4 groups (0.25, 0.50, 0.75 and 1.00) as seen in Table 2 (Scholl et al. 2013). Setup times used in this paper are 0.25 (low) and 1.00 (high) and available for download at "http://www.assembly-line-balancing.de". Thus, a total of 640 test instances are presented for ALWABPS.

Each test instance was run with the proposed mathematical model for 900 s (Ritt et al. 2015) and the results were reported. Thus, the model was run a total of 9000 s for just one data set. Since the proposed algorithm is a heuristic method, each instance was run 3 times and best results are reported.

First, Roszieg and Heskia test problem instances were solved, and their results were compared with heuristic algorithm. For Roszieg test instances, although 32 problems were solved optimally, other test problems in the same problem family were solved with small gap values. In all test problems, the proposed algorithm has yielded the same results created by the mathematical model as seen in Table 3. In addition, when the mathematical model and heuristic algorithm results are compared in the Heskia family, it is seen that the heuristic algorithm produces better results in a shorter time than the mathematical model as seen in Table 4. Thus, the efficiency of the proposed heuristic algorithm has been demonstrated and tested on large-scale test problems. Large-scale test problems belonging to Tonge and Wee-Mag families were solved with a mathematical model and algorithm, but the mathematical model could not find any solution despite the removal of the time limitation. The heuristic algorithm results are presented in Table 5.

7 Results and discussion

In this section, the performance of algorithm on some well-known test problems taken from the ALBP literature in terms of solution quality and running time is going to be examined. Since the problem has been presented for the first time in the literature, no comparable study is reported. For this reason, the presented algorithm and mathematical model have been compared on various problems. Algorithm and mathematical model were coded in C# language and GRUBI solver, and both were executed on a PC with Intel(R) Core(TM) i7, 2.20 GHz processor and 8.00 GB of RAM under a Microsoft Windows 10 environment.

Chaves et al. applied a clustering search algorithm and generated ALWABP benchmark test instances (Chaves et al. 2007). Hereby, the presented benchmark data sets, which are composed of four families (Roszieg, Heskia, Tonge and Wee-Mag), have been used in every ALWABP research study. In addition, the test instances presented by Scholl in the literature were used in terms of sequence-dependent setup times. Setup times used in this paper are 0.25 (low) and 1.00 (high) (Yolmeh and Kianfar 2012).

Small-size test instances used in the presented study are Roszieg and Heskia. These problems are used to compare the performance of the proposed algorithm with the mixed integer programming described in Sect. 2. The proposed mathematical model is solved using the GUROBI solver to find the optimal solution of the problems.

In Roszieg problem, 80 test instances with 0.25 setup time were examined, and the mathematical model optimally solved 32 of these problems. 48 of the feasible solutions found with an average Gap value of 3.91%. At the same time, the proposed algorithm yielded the same results compared with the mathematical model in all test

Table 3 Detailed results obtained by mathematical model and heuristic approach for Roszieg family

	Family	Mathematical model						Heuristic solutions			
	ROSZIEG	Setup level 1 (0.25)			Setup level 4 (1.00)			Setup level 1 (0.25)		Setup level 4 (1.00)	
	Inst	C	GAP (%)	CPU(s)	C	GAP(%)	CPU(s)	C	CPU(s)	C	CPU(s)
Group 1	1	21	0.00	244.79	26	3.84	900.00	21	9.21	26	9.97
	2	23	0.00	297.41	26	0.00	520.88	23	8.79	26	7.82
	3	19	0.00	233.89	24	12.50	900.00	19	8.08	24	8.59
	4	19	0.00	405.01	22	0.00	117.34	19	12.38	22	10.95
	5	18	0.00	251.78	21	0.00	497.52	18	7.04	21	7.49
	6	25	0.00	743.88	28	0.00	351.00	25	7.40	28	8.51
	7	22	0.00	175.55	25	0.00	226.85	22	7.47	25	10.17
	8	21	4.76	900.00	25	0.00	336.83	21	7.09	25	9.20
	9	23	0.00	383.35	26	0.00	152.28	23	10.25	26	7.69
	10	20	0.00	164.43	25	0.00	754.25	20	7.06	25	11.56
	Avg	21.10	0.48	380.01	24.80	1.63	475.70	21.10	8.48	24.80	9.19
Group 2	11	31	0.00	450.72	36	0.00	349.47	31	9.52	36	8.71
	12	28	0.00	309.86	34	0.00	431.21	28	7.15	34	8.00
	13	77	2.59	900.00	84	0.00	78.43	77	9.03	84	7.63
	14	26	0.00	127.64	29	0.00	654.58	26	15.37	29	8.92
	15	27	0.00	108.76	33	0.00	266.32	27	11.89	33	9.96
	16	23	0.00	397.19	28	0.00	545.72	23	7.09	28	10.97
	17	23	0.00	298.99	27	0.00	451.34	23	8.24	27	8.12
	18	21	0.00	203.72	25	12.00	900.00	21	7.05	25	13.47
	19	28	0.00	149.72	31	0.00	572.72	28	7.53	31	11.06
	20	41	0.00	460.91	45	0.00	170.21	41	11.03	45	7.43
	Avg	32.50	0.26	340.75	37.20	1.20	442.00	32.50	9.39	37.20	9.43
Group 3	21	29	0.00	329.37	35	14.28	900.00	29	12.44	35	7.41
	22	31	0.00	190.27	35	0.00	551.67	31	10.95	35	8.94
	23	27	0.00	745.54	31	0.00	900.00	27	7.13	31	12.30
	24	34	0.00	317.81	38	0.00	390.21	34	8.86	38	13.05
	25	29	0.00	405.01	34	14.71	514.25	29	10.09	34	10.19
	26	28	0.00	244.79	32	0.00	410.62	28	7.46	32	8.39
	27	22	0.00	625.91	27	0.00	545.81	22	12.98	27	8.64
	28	29	0.00	236.29	33	0.00	44.45	29	9.43	33	12.76
	29	28	0.00	218.95	31	0.00	169.86	28	7.60	31	7.61
	30	34	0.00	354.61	36	2.77	900.00	34	7.94	36	7.01
	Avg	29.10	0.00	366.86	33.20	3.18	532.69	29.10	9.49	33.20	9.63
Group 4	31	32	0.00	257.43	36	0.00	417.25	32	7.78	36	8.12
	32	30	0.00	294.33	34	2.94	900.00	30	7.49	34	7.08
	33	33	0.00	442.87	37	5.41	900.00	33	11.08	37	9.15
	34	28	3.57	900.00	33	3.03	900.00	28	7.29	33	7.29
	35	28	0.00	134.65	32	0.00	133.72	28	7.54	32	7.30
	36	30	0.00	218.55	33	0.00	218.07	30	9.74	33	11.13
	37	28	0.00	451.83	31	0.00	452.85	28	7.34	31	7.52
	38	29	0.00	354.31	33	0.00	353.92	29	10.35	33	14.54
	39	22	0.00	192.07	25	0.00	192.44	22	8.31	25	8.29
	40	30	0.00	749.95	33	0.00	746.71	30	8.06	33	7.54
	Avg	29.00	0.36	399.60	32.70	1.14	521.50	29.00	8.50	32.70	8.80

Table 3 (continued)

	Family	Mathematical model						Heuristic solutions			
	ROSZIEG	Setup level 1 (0.25)			Setup level 4 (1.00)			Setup level 1 (0.25)		Setup level 4 (1.00)	
	Inst	C	GAP (%)	CPU(s)	C	GAP(%)	CPU(s)	C	CPU(s)	C	CPU(s)
Group 5	41	11	9.09	900.00	14	28.57	900.00	11	9.52	14	11.37
	42	11	18.18	900.00	18	44.44	900.00	11	12.76	18	14.70
	43	11	0.00	505.26	16	56.25	900.00	11	10.12	16	10.74
	44	10	10.00	900.00	17	50.25	900.00	10	12.67	17	15.54
	45	13	7.69	900.00	17	64.70	900.00	13	9.54	17	11.22
	46	10	10.00	900.00	17	58.88	900.00	10	12.50	17	9.34
	47	11	9.09	900.00	14	66.66	900.00	11	14.72	14	9.06
	48	9	0.00	451.83	14	75.00	900.00	9	10.93	14	12.01
	49	11	0.00	749.95	15	46.13	900.00	11	9.07	15	11.52
	50	10	10.00	900.00	14	42.85	609.46	10	12.38	14	12.36
	Avg	10.70	7.41	800.70	15.60	53.37	870.95	10.70	11.42	15.60	11.79
Group 6	51	12	8.33	900.00	26	84.61	900.00	12	14.41	18	10.87
	52	11	18.18	900.00	15	66.66	900.00	11	9.84	15	12.24
	53	11	9.09	900.00	17	70.51	900.00	11	12.72	15	10.80
	54	11	9.09	900.00	17	60.00	900.00	11	16.95	17	17.41
	55	12	8.33	900.00	19	57.89	900.00	12	15.60	16	10.93
	56	14	7.14	900.00	23	56.21	900.00	14	14.00	20	16.57
	57	14	7.14	900.00	27	74.07	900.00	14	12.54	19	12.38
	58	12	8.33	900.00	20	75.00	900.00	12	13.45	19	14.60
	59	13	7.69	900.00	22	72.22	900.00	13	10.59	16	14.45
	60	10	10.00	900.00	19	73.84	900.00	10	12.18	14	13.36
	Avg	12.00	9.33	900.00	20.50	69.10	900.00	12.00	13.23	16.90	13.36
Group 7	61	17	5.88	900.00	25	68.00	900.00	17	10.23	23	10.42
	62	14	14.29	900.00	26	73.68	900.00	14	9.88	19	11.28
	63	20	5.00	900.00	28	53.57	900.00	20	10.89	26	11.38
	64	17	5.88	900.00	27	59.25	900.00	17	16.31	23	10.42
	65	15	6.67	900.00	23	56.21	900.00	15	15.89	22	12.14
	66	18	5.56	900.00	26	50.00	900.00	18	9.23	26	9.99
	67	18	5.56	900.00	24	66.66	900.00	18	9.24	24	17.22
	68	17	5.88	900.00	23	62.21	900.00	17	14.54	21	13.99
	69	16	6.25	900.00	28	71.42	900.00	16	11.22	19	9.39
	70	18	5.56	900.00	24	62.50	900.00	18	17.36	23	9.37
	Avg	17.00	6.65	900.00	25.40	62.35	900.00	17.00	12.48	22.60	11.56
Group 8	71	16	6.25	900.00	22	72.72	900.00	16	10.24	22	9.75
	72	17	11.76	900.00	24	70.83	900.00	17	16.80	23	9.42
	73	17	5.88	900.00	23	39.13	900.00	17	10.51	23	11.19
	74	17	5.88	900.00	27	66.66	900.00	17	15.56	21	13.37
	75	17	5.88	900.00	21	52.38	900.00	17	9.16	21	11.08
	76	18	5.56	900.00	22	63.63	900.00	18	13.63	22	12.19
	77	14	7.14	900.00	22	72.72	900.00	14	12.42	19	12.00
	78	15	6.67	900.00	20	65.00	900.00	15	16.35	20	16.37
	79	15	6.67	900.00	19	47.36	900.00	15	9.04	19	9.29
	80	15	6.67	900.00	23	65.21	900.00	15	9.24	23	9.89
	Avg	16.10	6.84	900.00	22.30	61.56	900.00	16.10	12.30	21.30	11.46

Table 4 Detailed results obtained by mathematical model and heuristic approach for Heskia family

	Family	Mathematical model			Heuristic solutions			
	HESKIA	Setup level 1 (0.25)	Setup level 4 (1.00)	CPU	Setup level 1 (0.25)		Setup level 4 (1.00)	
	Inst	C	C		C	CPU	C	CPU
Group 1	1	111	205	900	108	10.01	144	13.08
	2	122	180	900	121	11.12	152	18.48
	3	142	205	900	121	10.30	155	10.18
	4	129	173	900	116	14.02	154	11.86
	5	165	172	900	106	14.40	172	11.33
	6	119	143	900	111	16.80	141	10.31
	7	134	187	900	128	10.81	173	14.97
	8	134	168	900	134	12.53	168	11.07
	9	127	163	900	117	10.87	163	12.84
	10	166	204	900	155	13.01	183	15.21
	Avg	134.9	180	900	121.7	12.39	160.5	12.93
Group 2	11	199	231	900	189	12.23	224	12.44
	12	146	175	900	123	12.25	156	17.20
	13	145	178	900	124	17.92	161	13.23
	14	140	168	900	110	13.40	160	12.71
	15	163	208	900	144	17.56	200	13.96
	16	139	160	900	132	14.79	157	17.95
	17	166	214	900	163	12.77	199	18.24
	18	141	180	900	141	12.16	180	12.64
	19	121	161	900	113	12.02	150	12.46
	20	138	212	900	138	17.73	176	13.87
	Avg	149.8	188.7	900	137.7	14.28	176.3	14.47
Group 3	21	254	410	900	217	11.28	270	11.04
	22	216	252	900	166	10.91	236	10.11
	23	247	419	900	230	18.99	266	10.61
	24	240	256	900	201	10.16	249	10.85
	25	186	244	900	165	12.89	209	10.81
	26	217	270	900	209	15.56	241	18.16
	27	220	272	900	173	10.08	214	19.03
	28	257	292	900	217	10.47	279	14.22
	29	190	266	900	184	10.52	228	10.39
	30	184	266	900	184	11.58	211	12.10
	Avg	221.1	294.7	900	194.6	12.24	240.3	12.73
Group 4	31	291	307	900	220	14.17	264	13.10
	32	181	283	900	162	13.17	226	18.80
	33	300	289	900	235	16.19	283	15.51
	34	146	189	900	144	15.68	174	13.25
	35	200	283	900	200	16.42	250	12.31
	36	211	275	900	192	15.48	245	13.97
	37	270	283	900	211	12.55	237	12.44
	38	226	243	900	167	14.36	225	15.79
	39	190	252	900	190	12.22	232	15.43
	40	231	293	900	186	16.16	262	13.74
	Avg	224.6	269.7	900	190.7	14.64	239.8	14.43

Table 4 (continued)

	Family	Mathematical model			Heuristic solutions			
	HESKIA	Setup level 1 (0.25)	Setup level 4 (1.00)	CPU	Setup level 1 (0.25)		Setup level 4 (1.00)	
	Inst	C	C		C	CPU	C	CPU
Group 5	41	257	236	900	75	22.25	159	22.01
	42	158	252	900	95	29.76	161	19.42
	43	259	344	900	109	22.22	134	19.85
	44	433	208	900	259	22.08	204	30.72
	45	206	267	900	105	33.39	191	17.53
	46	442	356	900	72	21.42	187	17.65
	47	366	317	900	112	24.92	220	27.45
	48	250	359	900	83	17.57	147	21.64
	49	599	250	900	326	17.72	179	22.75
	50	262	INF	900	89	29.60	169	25.72
	Avg	323.2	287.67	900	132.5	24.09	175.1	22.47
Group 6	51	185	209	900	89	17.45	203	18.87
	52	303	352	900	98	26.38	154	18.87
	53	134	346	900	65	18.84	130	28.23
	54	129	257	900	68	26.56	121	33.69
	55	91	207	900	77	17.28	107	22.86
	56	260	192	900	53	25.17	127	25.49
	57	141	322	900	76	28.25	181	32.87
	58	229	243	900	136	24.26	204	20.01
	59	212	185	900	85	23.38	158	20.94
	60	138	227	900	52	41.84	137	17.43
	Avg	182.2	254	900	79.9	24.94	152.2	23.93
Group 7	61	513	282	900	167	24.71	282	17.86
	62	411	302	900	149	18.46	258	20.89
	63	239	307	900	210	24.16	229	23.38
	64	203	298	900	138	18.34	163	19.62
	65	650	444	900	116	22.94	223	23.85
	66	246	342	900	137	19.65	186	17.04
	67	236	245	900	98	21.76	154	25.17
	68	249	411	900	157	22.31	199	22.41
	69	316	347	900	147	19.79	159	20.96
	70	312	359	900	143	29.36	188	19.32
	Avg	337.5	333.7	900	146.2	22.15	204.1	21.05
Group 8	71	284	259	900	127	24.29	186	17.22
	72	544	347	900	176	24.34	303	22.98
	73	426	382	900	216	22.71	196	25.91
	74	208	258	900	112	31.72	169	36.98
	75	515	325	900	104	19.85	189	17.10
	76	172	274	900	120	24.08	164	38.10
	77	327	321	900	150	34.16	214	24.91
	78	319	294	900	112	26.12	144	24.09
	79	666	291	900	164	17.69	147	23.95
	80	296	247	900	127	22.87	161	22.75
	Avg	375.7	299.8	900	140.8	24.78	187.3	25.4

Table 5 Detailed results obtained by heuristic approach for Tonge and Wee-Mag family

	Family	Heuristic solutions				Family	Heuristic solutions			
	TONGE	Setup level 1 (0.25)		Setup level 4 (1.00)		WEE-MAG	Setup level 1 (0.25)		Setup level 4 (1.00)	
	Inst	0.25	CPU	1	CPU	Inst	0.25	CPU	1	CPU
Group 1	1	127	829	153	721	1	49	785	86	777
	2	116	718	178	701	2	55	811	96	831
	3	125	723	163	805	3	44	759	77	812
	4	137	756	179	748	4	55	813	90	754
	5	118	739	168	776	5	53	750	80	829
	6	116	737	158	767	6	54	760	89	772
	7	133	753	153	849	7	52	848	83	901
	8	126	754	167	779	8	50	795	83	772
	9	100	704	150	730	9	52	752	83	795
	10	119	712	144	724	10	49	754	80	763
	Avg	121.70	742.51	161	760.11	Avg	51	782.74	85	800.68
Group 2	11	137	737	197	711	11	54	780	85	773
	12	134	779	183	741	12	49	842	74	750
	13	132	721	176	812	13	51	796	78	769
	14	119	751	151	764	14	57	758	84	785
	15	113	726	153	764	15	61	750	92	820
	16	124	706	162	739	16	58	803	93	831
	17	150	940	165	717	17	54	787	87	863
	18	158	713	186	706	18	59	858	98	784
	19	144	711	185	833	19	56	767	87	788
	20	148	701	184	714	20	62	838	95	822
	Avg	135.90	748.53	174	750.20	Avg	56	797.80	87	798.60
Group 3	21	196	703	219	786	21	78	921	117	825
	22	232	702	231	712	22	67	801	97	776
	23	169	754	217	762	23	77	845	112	798
	24	178	757	204	774	24	73	760	100	774
	25	216	749	271	710	25	70	767	93	754
	26	202	703	235	785	26	77	911	104	752
	27	174	714	215	715	27	75	795	100	776
	28	217	772	259	867	28	74	773	103	758
	29	207	765	250	702	29	72	753	99	756
	30	199	781	220	743	30	86	771	117	841
	Avg	199.00	739.93	232	755.66	Avg	75	809.77	104	780.91
Group 4	31	197	758	229	741	31	77	800	100	817
	32	224	743	254	704	32	68	901	101	762
	33	222	710	246	712	33	76	797	107	770
	34	196	705	228	907	34	75	751	106	754
	35	162	760	204	758	35	64	891	93	788
	36	202	852	235	963	36	72	821	103	901
	37	195	751	231	713	37	63	903	94	752
	38	183	860	221	706	38	73	795	103	787
	39	217	754	259	707	39	78	797	104	795
	40	194	701	223	842	40	67	769	94	791
	Avg	199.20	759.17	233	775.48	Avg	71	822.52	101	791.57

Table 5 (continued)

	Family	Heuristic solutions				Family	Heuristic solutions			
	TONGE	Setup level 1 (0.25)		Setup level 4 (1.00)		WEE-MAG	Setup level 1 (0.25)		Setup level 4 (1.00)	
	Inst	0.25	CPU	1	CPU	Inst	0.25	CPU	1	CPU
Group 5	41	50	942	79	1002	41	28	1033	57	1418
	42	60	1061	98	948	42	28	1176	53	1142
	43	53	903	85	1123	43	27	1213	49	1170
	44	50	1052	82	954	44	26	1101	49	1138
	45	46	989	74	954	45	27	1387	50	1083
	46	52	931	80	907	46	27	1188	53	1203
	47	51	962	87	971	47	32	1131	52	972
	48	56	959	87	1044	48	32	1281	55	1312
	49	52	903	80	908	49	25	1200	46	1236
	50	55	1221	85	997	50	26	1240	47	1074
	Avg	52.50	992.38	84	980.67	Avg	28	1194.99	51	1174.87
Group 6	51	56	969	86	947	51	33	1028	60	1511
	52	54	988	81	935	52	26	1168	46	1426
	53	56	918	90	1011	53	29	1224	50	998
	54	64	925	106	1130	54	26	1072	46	1016
	55	53	1386	77	908	55	34	1305	59	1084
	56	56	995	79	1234	56	30	1388	49	1281
	57	62	908	108	1031	57	30	1263	53	996
	58	64	1044	104	1098	58	29	1121	52	1194
	59	54	1186	88	913	59	32	1029	61	1087
	60	56	1297	87	943	60	29	975	54	1046
	Avg	57.50	1061.73	91	1015.03	Avg	30	1157.38	53	1163.80
Group 7	61	88	955	102	958	61	36	1056	59	1176
	62	82	905	106	942	62	34	1261	59	1519
	63	79	966	99	952	63	40	1099	63	1534
	64	105	1079	125	1078	64	33	1035	49	995
	65	85	941	119	930	65	39	1020	64	1174
	66	84	1202	109	1197	66	35	1052	54	1006
	67	75	948	107	1035	67	42	1012	65	1035
	68	83	985	109	1071	68	37	1061	68	1385
	69	75	962	109	1091	69	39	1028	58	1003
	70	81	1468	119	938	70	42	1097	67	1259
	Avg	83.70	1041.15	110	1019.07	Avg	38	1072.33	61	1208.70
Group 8	71	77	947	103	1171	71	40	1053	63	982
	72	87	1165	125	918	72	43	1550	62	980
	73	92	1075	122	964	73	44	1023	65	1096
	74	97	1045	137	976	74	40	1092	65	1339
	75	81	1032	109	1422	75	38	1170	63	1049
	76	71	1225	100	944	76	28	1104	47	1013
	77	83	1366	123	1114	77	37	1252	60	1083
	78	95	911	131	1013	78	39	1049	64	1061
	79	86	935	118	955	79	44	1099	69	971
	80	80	1018	106	1041	80	40	1012	63	995
	Avg	84.90	1071.89	117	1051.80	Avg	39	1140.33	62	1056.92

instances. In addition, the total CPU time for mathematical model solutions is 49879 s. This solution time is 852 s for the algorithm. For the same test instances, 31 problems were solved optimally with the mathematical model at 1.00 setup time, while 49 test problems produced feasible solutions with a Gap value of 31.69%. Additionally, while the total time spent for mathematical model solutions is 55,428 s, the solution time took 852 s in the algorithm. Furthermore, the same results were found with the mathematical model in all 160 test problems for Roszieg.

In the Heskia problem, all of the maximum times were reached for each test instances defined for the mathematical model. Thus, the mathematical model was able to yield the same result with the algorithm in only 6 of 80 test problems with 0.25 setup time. The algorithm solved the 74 problem by creating better results. The algorithm presented in all 80 test problems with 1.00 setup time and provided better solutions to all problems than the mathematical model.

Thus, the efficiency of the proposed algorithm has been shown on small-sized test problems, and the results of large-sized test problems are given for Tong and Wee-Mag families in Table 5.

8 Conclusions and future research

In this paper, assembly line worker assignment and balancing problem with sequence-dependent setup times is introduced. For this purpose, a mathematical model formulation is developed for the problem. Proposed model is capable of solving some small-size instances optimally using GUROBI solver. As the mathematical model has difficulties finding optimal solutions for the real-life environment problems with a reasonable CPU time, an algorithm based on a simulated annealing is also proposed. To demonstrate the efficiency of the proposed approaches, a computational experiment is conducted. The results show that the proposed algorithm and mathematical model are effective and successful for the ALWABPS. The outcome of the study will help production managers test various possible scenarios for balancing ALWABP with forward and backward setup times and determine a feasible solution within an acceptable computational time for the production line.

According to our best knowledge, the presented paper is the first study for assembly line worker assignment and balancing problem with sequence-dependent setup times. This study is a good starting point for future studies. Future studies might extend the presented problem, like U-shaped assembly line balancing problems with disabled worker and setup time. The problem, which is NP-Hard problem structure, can be solved with exact solution algorithms. For this purpose, lower bound studies might be done to be used in the algorithm. Beside these, some real-life constraints, such as zoning constraints, positional constraints and so on, should be studied. Thus, the applicability of the studies in the literature to real-life problems will be easier.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Human and animal rights statement Humans/animals are not involved in this work.

References

Akpınar Ş, Baykasoğlu A (2014a) 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

Akpınar Ş, Baykasoğlu A (2014b) 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

Akpınar Ş, Bayhan GM, Baykasoğlu A (2013) Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Appl Soft Comput 13(1):574–589

Akpınar Ş, Elmi A, Bektaş T (2017) Combinatorial Benders cuts for assembly line balancing problems with setups. Eur J Oper Res 259(2):527–537

Akyol SD, Baykasoğlu A (2016) A multiple-rule based constructive randomized search algorithm for solving assembly line worker assignment and balancing problem. J Intell Manuf 30:557–573

Andres C, Miralles C, Pastor R (2008) Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. Eur J Oper Res 187(3):1212–1223

Baykasoğlu A, Dereli T (2008) Two-sided assembly line balancing using an ant-colony-based heuristic. Int J Adv Manuf Technol 36(5/6):582–588

Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 168(3):694–715

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

Çevikcan E, Durmusoğlu MB (2020) A novel optimization approach for segmented rabbit chase oriented U-type assembly line design: an application from lighting industry. Assem Autom 40(3):483–510

Chaves AA, Lorena LAN, Miralles C (2009) Hybrid metaheuristic for the assembly line worker assignment and balancing problem. Lect Notes Comput Sci 5818:1–14

Chaves AA, Miralles C, Lorena LAN (2007) Clustering search approach for the assembly line worker assignment and balancing problem. In: Proceedings of the 37th international conference on computers and industrial engineering, Alexandria, Egypt, pp 1469–1478

Dolgui A, Kovalev S, Kovalyov MY, Malyutin S, Soukhal A (2018) Optimal workforce assignment to operations of a paced assembly line. Eur J Oper Res 264:200–211

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

Gökçen H, Ağpak K, Benzer R (2006) Balancing of parallel assembly lines. Int J Prod Econ 103(2):600–609
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
Janardhanan MN, Li Z, Nielsen P (2019) Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem. Soft Comput 23:11263–11276
Kalayci CB, Gupta SM (2013) A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. Int J Adv Manuf Technol 69(1–4):197–209
Kirkpatrick S, Gelatt CD, Veechi MP (1983) Optimization by simulated annealing. Science 220:671–679
Li Z, Tang Q, Zhang L (2016) Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm. J Clean Prod 135:508–522
Martino L, Pastor R (2010) Heuristic procedures for solving the general assembly line balancing problem with setups. Int J Prod Res 48(6):1787–1804
Miralles C, Garcıa-Sabater JP, Andres C, Cardos M (2007) Advantages of assembly lines in sheltered work centres for disabled. A case study. Int J Prod Econ 110:187–197
Miralles C, Garcia-Sabater JP, Andres C, Cardos M (2008) Branch and bound procedures for solving the assembly line worker assignment and balancing problem: application to sheltered work centers for disabled. Discret Appl Math 156:352–367
Moreira MCO, Ritt M, Costa AM, Chaves AA (2012) Simple heuristics for the assembly line worker assignment and balancing problem. J Heurist 18:505–524
Mutlu Ö, Polat O, Supciller AA (2013) An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. Comput Oper Res 40:418–426
Nazarian E, Ko J, Wang H (2010) Design of multi-product manufacturing lines with the consideration of product change dependent inter-task times, reduced changeover and machine flexibility. J Manuf Syst 29(1):35–46
Özcan U (2019) Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. Int J Prod Econ 213:81–96
Özcan U, Toklu B (2010) Balancing two-sided assembly lines with sequence-dependent setup times. Int J Prod Res 48(18):5363–5383
Pereira J (2018) The robust (minmax regret) assembly line worker assignment and balancing problem. Comput Oper Res 93:27–40
Ramezanian R, Ezzatpanah A (2015) Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem. Comput Ind Eng 87:74–80
Ritt M, Costa AM, Miralles C (2015) The assembly line worker assignment and balancing problem with stochastic worker availability. Int J Prod Res 54(3):907–922
Roshani A, Giglio D (2017) Simulated annealing algorithms for the multi-manned assembly line balancing problem: minimising cycle time. Int J Prod Res 55(10):2731–2751
Şahin M, Kellegöz T (2017) Increasing production rate in U-type assembly lines with sequence-dependent set-up times. Eng Optim 49(8):1401–1419
Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur J Oper Res 168(3):666–693
Scholl A, Boysen N, Fliedner M (2008) The sequence-dependent assembly line balancing problem. Or Spectrum 30(3):579–609
Scholl A, Boysen N, Fliedner M (2013) The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. Or Spectrum 35(1):291–320
Serin F, Mete S, Çelik E (2019) An efficient algorithm for U-type assembly line re-balancing problem with stochastic task times. Assem Autom 39(4):581–595
Seyed-Alagheband SA, Ghomi SF, Zandieh M (2011) A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. Int J Prod Res 49(3):805–825
Shin J, Lee M, Morrison JR (2019) On the optimization of cycle time in assembly lines with parallel workstations and tasks requiring multiple workers. In: IEEE 15th international conference on automation science and engineering (CASE), Vancouver, BC, Canada, pp 916–921
Vila M, Pereira J (2014) A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Comput Oper Res 44:105–114
Yadav A, Kulhary R, Nishad R, Agrawal SK (2019) Parallel two-sided assembly line balancing with tools and tasks sharing. Assem Autom. https://doi.org/10.1108/AA-02-2018-025
Yang W, Cheng W (2020) A mathematical model and a simulated annealing algorithm for balancing multi-manned assembly line problem with sequence-dependent setup time. Math Probl Eng 2020:1–16
Yazdanparast V, Hajihosseini H, Bahalke A (2011) Cost oriented assembly line balancing problem with sequence dependent setup times. Aust J Basic Appl Sci 5(9):878–884
Yeh DH, Kao HH (2009) A new bidirectional heuristic for the assembly line balancing problem. Comput Ind Eng 57(4):1155–1160
Yolmeh A, Kianfar F (2012) An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. Comput Ind Eng 62(4):936–945