Production Engineering (2024) 18:533–556
https://doi.org/10.1007/s11740-023-01246-y

PROCESSES

A hybrid metaheuristic based on the firefly and bat algorithms to solve the alternative subgraphs assembly line balancing problem

Ümmühan Palamut[1] · Şener Akpinar[1]

Received: 5 August 2023 / Accepted: 23 November 2023 / Published online: 20 December 2023
© The Author(s) under exclusive licence to German Academic Society for Production Engineering (WGP) 2023

Abstract

Assembly line balancing problem is the process of assigning a set of tasks to a group of stations, considering the precedence relations between tasks. Precedence relations are commonly definite; however, tasks may have alternative precedence relations due to different mounting alternatives. In the presence of alternative precedence relations, the classical balancing problem becomes more complicated given that two interdependent problems, namely selection and balancing, must be solved hierarchically. This type of balancing problem is named as the Alternative Subgraph Assembly Line Balancing Problem (ASALBP). This current paper concerns itself to develop an efficient solution procedure for the ASALBP by hybridizing the firefly and bat algorithms. The proposed hybrid algorithm is evaluated on a set of benchmark problems taken from the related literature and numerically compared against the standard firefly and bat algorithms, and some formerly developed heuristic methods. Computational results reveal the satisfactory performance of the proposed algorithm in solving ASALBP instances.

Keywords Alternative subgraph assembly line balancing problem · Hybrid metaheuristic · Firefly algorithm · Bat algorithm

Abbreviations

ASALBP	Alternative subgraph assembly line balancing problem
ALBP	Assembly line balancing problem
SALBP	Simple assembly line balancing problem
GALBP	Generalized assembly line balancing problem
GRASP	Greedy randomized adaptive search procedure
FA	Firefly algorithm
BA	Bat algorithm
HFAB	Hybrid firefly and bat
ALBESP	Assembly line balancing and equipment selection problem

1 Introduction

Assembly lines – a combination of workstations with a material handling system – are one of the most important types of the production systems. Thanks to assembly lines producing large quantities of products become possible and they have been provided efficiently use of scarce resources. To benefit from these advantages of the assembly lines, they are required to be designed as efficiently as possible. The design problem of the assembly lines is known as the balancing problem, which is one of the main topics in the related literature on optimization of the assembly lines. Assembly line balancing problem (ALBP) is known as the decision problem of optimally partitioning the assembly work among the workstations with respect to some objectives [1]. In this problem type, the objective could be either to reduce the number of workstations or cycle time. According to [2], ALBPs consist of two basic types of problems, simple assembly line balancing problem (SALBP) and generalized assembly line balancing problem (GALBP). The simple case (SALBP) is an assembly line where only one standard product is produced on a serial assembly line. If the problem have more complexity and limitations, then it should be considered as GALBP. The reader could be suggested to read the review papers of [3–5] for more detailed information.

✉ Şener Akpinar
sener.akpinar@deu.edu.tr

Ümmühan Palamut
ummuhanpalamut@hotmail.com

[1] Department of Industrial Engineering, Dokuz Eylül University, TR-35397 Buca Izmir, Turkey

Normally, in an assembly line balancing problem, the precedence graph of all tasks is one, while in real-life problems some tasks may have optional precedence relations [6–9]. Within this context, [10, 11] have introduced a new kind of GALBP named as Alternative Subgraph Assembly Line Balancing Problem (ASALBP). This problem is mostly related to the assembly line balancing problem faced by suppliers with a large product range and product number, which enables them to offer more options to their customers in direct proportion to increasing customer needs. The distinctive aspect of the ASALBP is that it has alternative sub-graph precedence relations, not definite precedence relations. The ASALBP mainly considers the mounting alternatives for different parts of an assembly or manufacturing process. Each alternative is represented by a sub-graph that specifies the tasks, task times and task precedence relations required to produce a product. It must be here noted that each subassembly option has a specific task sequence with different precedence relations. Like the other ALBPs, the ASALBP is mainly divided into two groups with regard to its objective: ASALBP–1 seeks the minimum number of stations for a given cycle time while ASALBP–2 tries to minimize the cycle time for a given number of stations [10, 11]. This current paper intends to tackle the ASALBP–1 by developing an effective approximation algorithm.

ASALBP–1 mainly contains two decisions to be resolved: selecting one of the alternative precedence subgraph and then assigning assembly tasks to a minimum number of stations for a given cycle time. As the number of precedence sub-graph alternatives and tasks numbers increases and therefore the solution space expands, ASALBP–1 is evaluated in the NP-hard problem class [10, 11]. Additionally, as the problem data grows, it is required too much time to solve the problem to optimality via exact solution methods. Thereof, designing effective approximation algorithms to tackle the ASALBP–1 turns into a necessity. In the related literature, although, ASALBP–1 has been solved by many heuristic methods, metaheuristics have not been studied extensively on this subject. Consequently, in the related literature, designing metaheuristic algorithms to solve the ASALBP–1 comes out as a research gap, which is going to be exhibited in the literature review section. Therefore, this current paper proposes a novel hybrid metaheuristic that combines the firefly algorithm (FA) [12] with the bat algorithm (BA) [13] to solve the ASALBP–1 as effectively as possible.

The remainder of this paper is organized as follows. Problem definition and the relevant literature on ASALBPs are given in Sect. 2. The developed hybrid metaheuristic is described in Sect. 3. Computational results are given in Sect. 4. Finally, the paper concludes with some remarks in Sect. 5.

2 Problem definition and literature review

This section firstly describes the ASALBP–1 through the mathematical programming model developed by [14]. Afterwards, the literature on ASALBPs is summarized.

2.1 Problem formulation

ASALBP–1 as a general assembly line balancing problem refers to the design problem of a product assembly process where different assembly alternatives exist. In the presence of different assembly alternatives, the balancing problem becomes more complicated, since two interdependent problems must be solved simultaneously. These two problems may be named as selection and assignment problems. The selection problem aims to choose the most appropriate one among the existing assembly alternatives, while, the balancing problem aims to partition the assembly tasks among a minimum number of stations by considering a predetermined cycle time with regard to selected alternative. The ASALBP–1 has been formulated as a binary linear programming model by [10, 11]. Later [14] extended the definition of the ASALBP–1 with the consideration of sub-processes composed of different tasks and formulized the problem as a mathematical programming model. The notations with their definitions belonging to this formulation are given in Table 1. This current paper is interested in this extended variant of the ASALBP–1 [15].

$$MinZ = \sum_{j=m_{min}+1}^{m_{max}} j * y_j \tag{1}$$

$$\sum_{j=E_{i0}}^{L_{i0}} x_{ij0} = 1 \quad \forall i | i \in TR_0 \tag{2}$$

$$\sum_{\forall r \in R_i} \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} = \sum_{\forall r \in R_i} ar_r \quad \forall i | i \notin TR_0 \tag{3}$$

$$\sum_{r=0 \forall i | (r \in R_i)}^{nr} \sum_{(j \in [E_{ir}, L_{ir}])} t_{ir} \times x_{ijr} \leq C \quad j = 1, \ldots, m_{\min} \tag{4}$$

$$\sum_{r=0}^{nr} \sum_{\forall i | (r \in R_i)(j \in [E_{ir}, L_{ir}])} t_{ir} \times x_{ijr} \leq C \quad j = 1, \ldots, m_{\min} \tag{5}$$

$$\sum_{j=E_{p0}}^{L_{po}} j * x_{pj0} \leq \sum_{j=E_{i0}}^{L_{i0}} j * x_{ij0} \quad \forall i \in TR_0, \forall p \in PT_i | p \in TR_0 \tag{6}$$

Table 1 Model notations with their definitions

	n	Number of tasks
Parameters	nr	Number of routes
	nsr	Number of different sets of routes (subgraphs) such that the routes within a set are alternatives to each other
	$m_{max}(m_{min})$	Upper (Lower) bound on the number of stations
	R_i	Set of all assembly routes through which task i can be processed
	C	Cycle time
	t_{ir}	Processing time of task i, if task i processed route r
	TR_r	Set of tasks that are affected by route r
	P_{ir}	Set of the possible immediate predecessors of task i, if task i processed through route r
	PT_i	Set of all possible immediate predecessors of task i
	$E_{ir}(L_{ir})$	Earliest (Latest) station that task i can be assigned to, if task i is processed through route r
	SCR_q	Subset of q assembly routes that are alternative among one other
Decision variables	$x_{ijr} \in \{0,1\}$	Equals 1 if task i is assigned to station j and processed through route r, and 0 otherwise
	$y_j \in \{0,1\}$	Equals 1 if there is any task assigned to station j, and 0 otherwise
	$ar_r \in \{0,1\}$	Equals 1 if there is any task processed through route r, and 0 otherwise

$$\sum_{\forall s \in R_p} \sum_{j=E_{ps}}^{L_{ps}} j * x_{pjs} \leq \sum_{j=E_{io}}^{L_{i0}} j * x_{ij0} \quad \forall i \in TR_0, \forall p \in PT_i | p \notin TR_0 \tag{7}$$

$$\sum_{j \in E_{p0}}^{L_{p0}} j * x_{pj0} \leq \sum_{\forall r \in R_i} \sum_{j=E_{ir}}^{L_{ir}} j * x_{ijr} + m_{max} * \left(1 - \sum_{\forall r \in R_i} ar_r\right) \quad \forall i \notin TR_0, \forall p \in PT_i | p \in TR_0 \tag{8}$$

$$\sum_{j=E_{pr}}^{L_{pr}} j \times x_{pjr} \leq \sum_{j=E_{ir}}^{L_{ir}} j \times x_{ijr} \quad \forall i \notin TR_0, \forall r \in R_i, \forall p \in P_{ir} | \left[p \notin TR_0 r \in R_p\right] \tag{9}$$

$$\sum_{\forall s \in R_p} \sum_{j=E_{ps}}^{L_{ps}} j \times x_{pjs} \leq \sum_{\forall r \in R_i} \sum_{j=E_{ir}}^{L_{ir}} j \times x_{ijr} \quad \forall i \notin TR_0, \forall p \in PT_i | \left[p \notin TR_0 \left(R_i \cap R_p = \emptyset\right)\right] \tag{10}$$

$$\sum_{r \in SCR_q} ar_r = 1 \quad q = 1, \ldots, nsr \tag{11}$$

$$\sum_{\forall i \in TR_r} \sum_{j=E_{ir}}^{L_{ir}} x_{ijr} \leq ar_r * \left|TR_r\right| \quad r = 1, \ldots, nr \tag{12}$$

ASALBP–1 seeks the optimal assignments of tasks to establish the minimum number of stations for definite cycle time. Accordingly, the objective function (1) of the model minimizes the total number of stations. All the tasks must be completed in definite stations, therefore, constraint set (2) and (3) guarantee that every task—belonging to a selected route—is assigned to exactly one station. Constraint sets (4) and (5) ensure the sum of the processing times of the tasks assigned to a station not to exceed the predetermined cycle time, since every stations has a capacity specified with the cycle time. All the assembly line balancing problems has an essential constraint set—must exactly be fulfilled—as the precedence relations between tasks due to some technical requirements. Because of this essential requirement, constraint set (6), (7), (8), (9) and (10) guarantee that a task can only be assigned to station j if all its predecessors have already been assigned to a preceding station of j on the line or to station j. Constraint set (11) constructs a unique assembly route by selecting exactly one route for each subassembly among the possible routes. Finally, constraint set (12) ensures assigning the tasks belonging to a particular subgraph to the same route.

2.2 Literature review

Product assembly therefore the assembly line balancing problems commonly contains certain precedence relations between assembly tasks. Nevertheless, products may have some parts, which could have alternative assembly processes in real life, and this requires to selection of one of the alternative precedence relations before seeking the optimum balance of the related assembly line. In such circumstances, the balancing problem of an assembly line becomes more complicated due to alternative assemblies and therefore the alternative precedence relations between tasks [6] handled such an assembly line balancing problem for the first time in the related literature. The processing alternatives had been occurred due to need to select of limited equipment in the Authors' case; however, the precedence relations between tasks have always

been maintained. The assembly line balancing problem with processing alternatives due to need to select of limited equipment was been studied by also [8]. On the other hand, assembly line balancing problem that contains alternative subgraphs were been handled by [15, 7 and 9] for the industries of power washers manufacturing, toy manufacturing and commercial hand-held drills production, respectively.

Subsequent to afore-mentioned works, [10] and [11] introduced the alternative subgraph assembly line balancing problem (ASALBP) for the first time in the related literature. They gave the character traits of the ASALBP, described the problem with small examples and formulated the problem as a binary linear programming model to solve the problem to optimality. [14] introduced another variant of ASALBP with assembly subprocesses with different tasks. The Authors also formulized this new variant of the ASALBP as a mathematical programming model to solve the problem to optimality. The exact solution methods developed to solve both above-mentioned variants of ASALBP are suffering to solve medium and large-scale problem instances. Therefore, [16] proposed three single-pass and two multi-pass heuristic methods to solve medium and large-scale problem instances of the ASALBP–1. Additionally, in the related literature, 14 different heuristic methods [17], approximation methods [18] and constructive heuristics methods [19] were also been developed to solve the ASALBP–1 as effectively as possible. [20] proposed an alternative representation of the ASALBP and the Authors formulated the problem as a new mixed integer programming model and tried to solve through a SALOME based solution approach. As distinct the previous works, [21] tried to solve the ASALBP with a metaheuristic algorithm, Greedy Randomized Adaptive Search Procedure (GRASP). The proposed GRASP selects the subgraphs, determines initial assignments of tasks in the first phase, and tries to improve the solution through a local search in the second phase. Besides this work, a new variant of the problem – ASALBP with resource selection and parallel stations – has also been solved through a genetic algorithm by [22]. The Authors have also formulated the new variant of the problem as a mathematical programming model. The aforementioned literature review on ASALBPs indicates the lack of metaheuristic algorithms to tackle the problem, since it is difficult to solve the medium and large-scale problem instances through the proposed mathematical programming approaches. However, there are plenty number of heuristics, whose results can potentially be improved, designed to solve the problem in the related literature. Additionally, studies in the related literature have suggested designing metaheuristic algorithms for ASALBP as future research directions. To contribute the research on eliminating this research gap, this current paper aspires to develop a hybrid metaheuristic based on firefly and bat algorithms to solve the ASALBP-1 as effectively as possible.

Both the firefly and bat algorithms were originally designed to solve continuous optimization problems, however, these two algorithms have been employed to solve plenty types of optimization problems. 2 recent review papers of [23] and [24] comprehensively classified the variants and application areas of these two algorithms. The reader can strongly be suggested to read these two review papers to be informed about the firefly and bat algorithms. These two review papers do not contain any research paper on the implementation of these two algorithms to an assembly line balancing problems in the literature. However, [25] compared the performance of BAT for the Assembly Line Balancing and Equipment Selection Problem (ALBESP), in which Product Design Alternatives are considered, againts some other metaheuristic algorithms. The scarcity of the implementations of the BA and FA on the assemble line balancing problem has also been exhibited by a recent review paper on the assembly line balancing problems [26].From this point of view, we can conclude that this current paper will reveal some information about the potential performances of both firefly and bat algorithms in solving assembly line balancing problems, especially for the ASALBP.

As a final note, it must be here noted, combining the firefly and bat algorithms within a hybrid algorithmic framework is a scarce research field, since we could attain only three papers hybridized these two algorithms [27–29] from a Web-of-Science search. This situation also brings an added value to the struggle of this current paper.

3 Hybrid firefly and bat algorithm for solving ASALBP–1

This section presents the proposed hybrid Firefly and Bat (HFAB) Algorithm to solve the ASALBP–1 in detail. As a part of a hybrid algorithm, firefly algorithm has the potentials of improving the poor solutions and providing an accelerated convergence speed [30], while bat algorithm has the potential of providing an improved convergence rate [31]. Therefore, it is projected to combine these two algorithms under a hybrid algorithmic framework would provide an effective solution procedure for ASALBP–1.

The proposed hybrid HFAB algorithm starts its search by tuning the values of the required parameters. The HFAB then generates an initial population of fireflies and updates the population through the search procedure of FA. Afterwards, the BA catches the new population formed because of the FA. At this point, BA reacts as a local search procedure and tries to improve the population through its pulse rate and loudness condition. The HFAB fulfils one single iteration through this hierarchical execution of the FA and BA. Following, The HFAB checks the termination criteria: if it is satisfied the algorithm ends and reports the best solution ever found, otherwise the FA catches the population improved by the BA. Within this context, the proposed HFAB algorithm – depicted in Fig. 1 – repeats itself until a pre-specified number

of iterations reached. The algorithmic components of the proposed HFAB algorithm is going to be explained in the following sub-sections in detailed.

3.1 Solution representation and initial population

A solution representation of the ASALBP–1 must be consisted of the sum of the tasks of the subgraph, chosen by the number of subassemblies that the assembly line balancing problem has. Therefore, the HFAB algorithm employs the task-based representation, as it is the best fit for any assembly line balancing problem in the related literature. Figure 2a visualizes an example precedence diagram consist of two sub-assemblies each one having two subgraphs, while Fig. 2b depicts the solution representation of the assembly process resulted from the selections of subgraph-2 and subgraph-3.

The total number of assembly tasks resulted from the selected subgraphs defines the length of the solution representation. Tasks are assigned to stations by obeying the sequence given as the solution representation (Fig. 2b) to

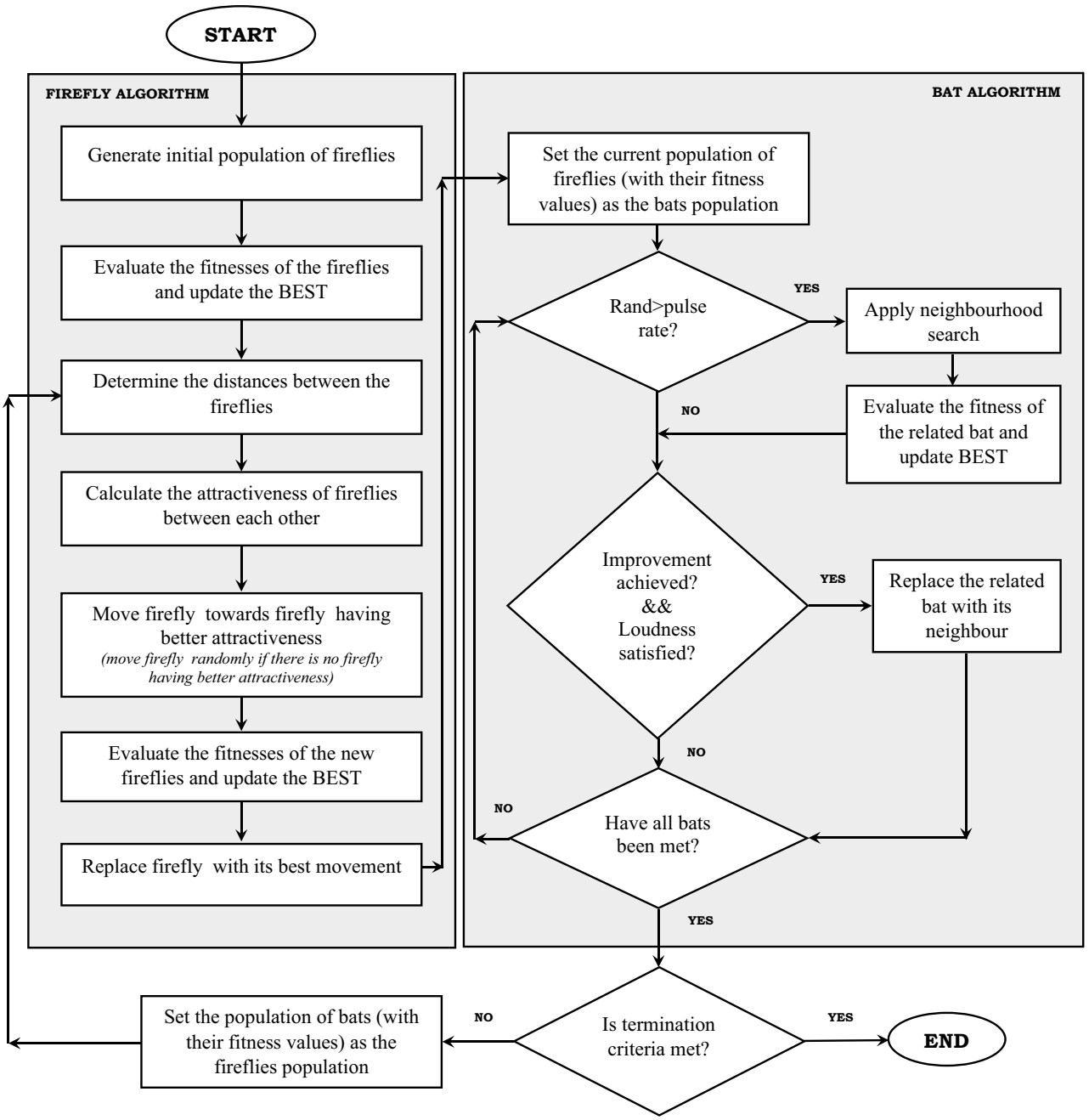


Fig. 1 Flowchart of the proposed HFAB algorithm

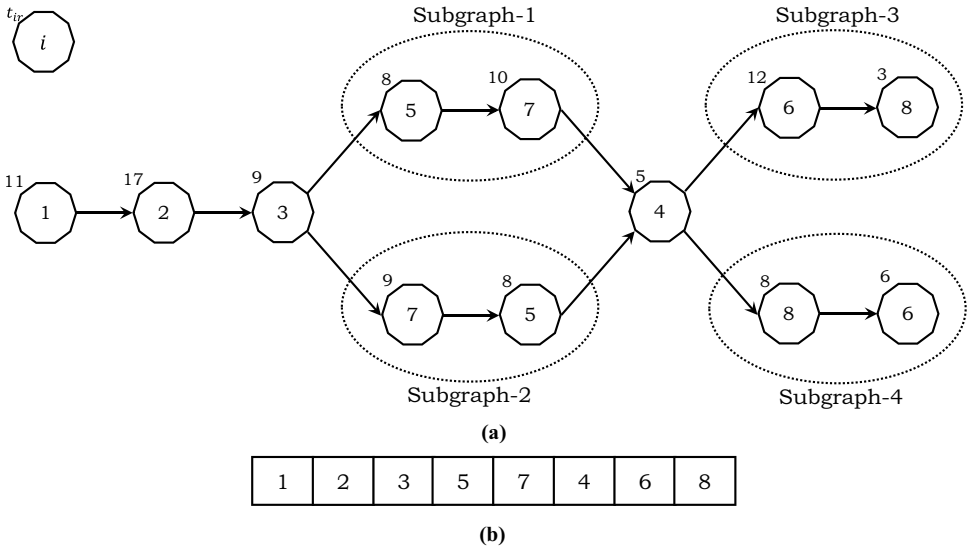


Fig. 2 **a** Precedence diagram for an example problem, **b** Task based solution representation

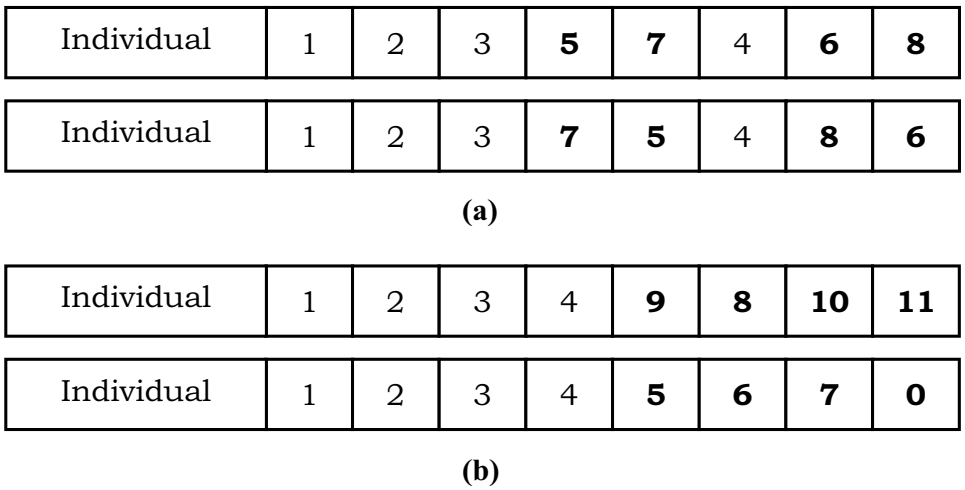


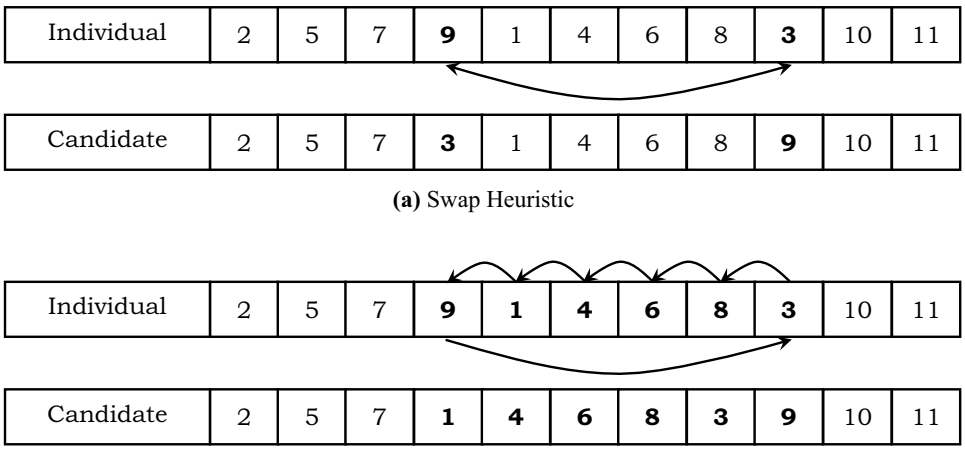
Fig. 3 Hamming distance calculation between two individuals **a** having number of tasks same tasks and **b** having different numbers of tasks

decode it. When a station's idle time does not permit to assign a new task, a new station is opened. This procedure continues until all tasks are assigned to a station.

The proposed HFAB algorithm starts its search by constructing an initial population with a predefined size. Each individual of the population consists of fixed tasks, and different tasks that the alternative subgraphs have of different subassemblies. Therefore, the HFAB firstly selects the subgraphs for every individual randomly. The population is formed from assembly tasks with the same or different subgraphs. All the alternatives option combinations are used at least once for each assembly. After selecting the alternative subgraphs, individuals are created with the ranked positional weight technique [32] and randomly – since previous studies [18] reported good results for these two techniques – to form initial population.

3.2 Distance and attractiveness calculation

After forming the population, the proposed HFAB algorithm employs FA firstly. In the FA phase, the proposed HFAB firstly determines the attractiveness of the individuals between each other. The attractiveness of an individual mainly depends on its performance namely fitness function value. Besides the fitness function, the other component of the attractiveness of an individual is the distance to other individuals. In other words, individual f attracts individual g if f has better fitness value than g, and the level of attractiveness depends on the rate calculated by the dividing the fitness value of f to the distance between the individuals f and g. The proposed HFAB algorithm uses the Hamming distance – the sum of the number of tasks in different orders between the two different



(a) Swap Heuristic

(b) Insertion Heuristic

Fig. 4 Neighbourhood search algorithms

Table 2 Benchmark problems

Problem size	Problem name	Cycle times					Number of subgraphs		
		C1	C2	C3	C4	C5	5	8	11
							Number of tasks		
Small size	Mansor	48	62	94	–	–	11	–	–
	Bowman	20	–	–	–	–	10	–	–
	Buxey	30	36	54	–	–	29	–	–
	Mitchell	14	21	35	–	–	21	21	–
Medium size	Gunther	41	49	49	61	81	37	37	37
	Kilbridge	51	79	92	138	184	45	46	48
	Hahn	2004	2338	2806	3504	4676	56	56	63
	Warnecke	54	62	74	92	111	63	63	67
	Tonge	160	176	207	251	320	73	75	75
Large size	Arcus2	5785	6540	7916	9400	11,570	115	121	125
	Bartholdi	403	470	564	705	805	151	157	160
	Scholl	1394	1584	1699	2049	2787	299	302	305

assembly sequences – when calculating the attractiveness of the individuals between each other.

Alternative subgraphs in ASALBP–1 can contain the same or different number of tasks. Therefore, any two individuals may have the same number of tasks or different number of tasks. Figure 3 depicts the distances – calculated with different tasks (shown in bold) in different assembly order – between any two individuals having the same number of tasks (Fig. 3a) and different number of tasks (Fig. 3b).

In accordance with, the distance between the two individuals in Fig. 3a is 4. If it has different number of tasks as in Fig. 3b, the distance is calculated over the individual having greater number of tasks. The missing number of digits in the individual having less number of tasks are indicated with zero at the end of the sequence. Thus, the distance between the individuals in Fig. 3b is found as four, too.

As the distance between individuals decreases, the light intensities that they use to impress other individuals increases, and therefore, the attractiveness between two individuals increases. The proposed HFAB algorithm calculates the attractiveness between any two individuals through Eq. 13 as original FA does.

$$\beta_{fg} = \beta_o * e^{-\gamma d^2} \tag{13}$$

where, β_o is the initial attraction when $d = 0$, γ is the light absorption coefficient and d is the distance between the individuals f and g. If the attractiveness of the individual f is greater than the individual g, then the individual g is going to move towards f. In this way, the proposed HFAB algorithm tries to improve the individuals as the original FA does.

3.3 Movement of individuals

After calculating the attractiveness of the individuals, the proposed HFAB algorithm is going to generate new individuals from the existing ones through the movement operation of the FA. The HFAB algorithm moves the individual f towards the individual g having a better attractiveness through the Eq. 14.

$$x_f = r\text{and}\left(2, d_{fg}\right) \tag{14}$$

The amount of movement of individual f – x_f – is a random integer chosen from the interval $[2, d_{fg}]$, where d_{fg} is the amount of distance between the individuals of f and g. In this step, the proposed HFAB algorithm applies x_f times a neighbourhood search algorithm to the individual f. Therefore, the HFAB algorithm generates x_f candidate individuals from the individual f and replace individual f with the best candidate if it is better than f. The proposed HFAB algorithm employs the swap and insertion heuristics (see Fig. 4) as the neighbourhood search algorithms. One of the heuristics are randomly selected while generating a candidate individual. Both the swap and insertion heuristics randomly select the tasks to be relocated and generates a candidate individual if and only if relocation of the selected tasks do not violate the precedence constraints.

3.4 Local search

After the movement step, FA phase of the proposed HFAB algorithm is completed and newly formed population is going to be caught by the BA to perform a local search phase. At this phase the proposed HFAB algorithm only employs the pulse rate and loudness condition of the original BA to generate new individuals and decide whether to accept them. In line with this purpose, the HFAB algorithm randomly selects a number from the interval [0, 1] for each individual of the population and compares it against the pulse rate parameter of the BA. If the randomly selected number is greater than the pulse rate for the related parent individual, then, the HFAB algorithm randomly selects one of the neighbourhood search algorithms (swap and insertion) and generate a new individual by applying it once on the parent individual. The HFAB algorithm replaces the parent individuals with the newly generated individual, if it has a better or equal fitness value in comparison to its parent and satisfies the loudness condition simultaneously. The loudness condition – A_f – is a parameter and it generally takes a random value from the interval [0.7, 1] for each newly generated individual f. To check the loudness condition for each newly generated individual f, the HFAB algorithm selects a random number from the interval [0, 1] and compares it against the loudness value of A_f. If the randomly selected number is less than the loudness value, then the loudness condition is satisfied.

3.5 Termination criteria

After the local phase, the HFAB algorithm checks whether the termination criteria is satisfied. If it is satisfied the algorithm ends itself and reports the BEST individual ever found, otherwise, FA catches the population formed by BA, and FA and BA iteratively and hierarchically runs within the concept of the proposed HFAB until the termination criteria is met. For the termination criteria, the HFAB algorithm uses both a pre-defined number of iterations and a pre-defined execution time limit.

4 Performance evaluation of the proposed HFAB algorithm

This section firstly compares the proposed HFAB algorithm against the standard FA and BA on a set of ASALBP–1 benchmark problems. Afterwards, a comparison against formerly developed 14 heuristics methods [18] are also provided. Following subsections respectively present main characteristics of the benchmark problems, and the computational results.

4.1 Benchmark problems

The performance of the proposed HFAB algorithm is evaluated on 12 benchmark sets [14] consisting of 133 problem instances in total and available at www.assembly line balancing.de. The benchmark sets included 13 small-sized problems, 75 medium-sized problems and 45 large-scale problems, and consisted of different subgraphs, tasks and cycle times. Main characteristics of the test problems are exhibited in Table 2.

4.2 Computational results

The proposed HFAB algorithm and the standard firefly and bat algorithms were coded in MATLAB, and run on a Core™ i3 CPU 2.27 GHz personal computer. The proposed HFAB algorithm has some control parameters required to be tune before execution. The values of the control parameters were determined through some preliminary experiments and Taguchi method [33], and are given in Table 3.

PS and *TC* are the parameter values should be set on a problem size basis and we defined an interval for the parameter *A*. Therefore, the values for these three parameters were set through preliminary experiments. For the remaining three parameters of *PR*, γ and β_o, we determined three levels for each as a result of preliminary experiments as $PR(1) = 0.1$, $PR(2) = 0.15$, $PR(3) = 0.2$, $\gamma(1) = 0.05$,

Table 3 Parameter values of the HFAB algorithm

Parameter	Definition	Value									
		5 Subgraphs			8 Subgraphs			11 Subgraphs			
		Problem Size			Problem Size			Problem Size			
		Small	Medium	Large	Small	Medium	Large	Small	Medium	Large	
PS	Population size	20	40	50	20	40	50	20	40	50	
A	Loudness value	[0.7, 1]	[0.7, 1]	[0.7, 1]	[0.7, 1]	[0.7, 1]	[0.7, 1]	[0.7, 1]	[0.7, 1]	[0.7, 1]	
PR	Pulse rate	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	
γ	Light absorption coefficient	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	
β_o	Initial attraction	1	1	1	1	1	1	1	1	1	
TC	Termination criteria	200 iterations	2.000 iterations	18.000 s	200 iterations	2.000 iterations	18.000 s	200 iterations	2.000 iterations	18.000 s	

Table 4 Comparison of the HFAB algorithm against FA and BA

Problem size	Algorithm		
	FA	BA	HFAB
Small	13	13	13
Medium	68	69	75
Large	45	45	45
Total	126	127	133

$\gamma(2) = 0.1$, $\gamma(3) = 0.15$, $\beta_o(1) = 1$, $\beta_o(2) = 2$ and $\beta_o(3) = 3$. To identify the appropriate values of these three parameters a Taguchi method was conducted. From Taguchi results it was observed that these 3 parameters have a meaningful effect at the 5% significance level over the objective value, and the parameters PR, γ and β_o were set to 0.1, 0.05 and 1, respectively.

After fine-tuning the values of the HFAB's parameters, Table 4 firstly compares the proposed HFAB algorithm against the standard FA and BA by reporting the total number of best solutions achieved for the 133 ASALBP–1 instances after 10 runs.

As can be clearly observed from Table 4, the proposed HFAB algorithm was able to achieve best results for all the 133 problems, while standard FA and BA were able to achieve 126 and 127 best results respectively. In other words, the proposed HFAB algorithm has a success level of 100%, while standard FA and BA have success levels of 94.74% and 95.49% respectively. From these findings, we can claim that the HFAB algorithm is an improvement for both the standard firefly and bat algorithms on solving the ASALBP–1 instances, especially for the medium-sized problems.

After putting forward, the satisfactory performance of the proposed HFAB algorithm in solving ASALBP–1 instances, now it is turn to come to compare the HFAB algorithm against he formerly developed solution procedures in the related literature. In line with this purpose, the HFAB algorithm is comparable against 14 heuristics methods [18] and GRASP [21], since the benchmark set was also been solved within the context of this two studies. However, unfortunately, between these two studies, results of the GRASP algorithm is not available for us. Therefore, we can compare the HFAB algorithm against the heuristic methods only. Table 5 compares the HFAB algorithm against the heuristic methods by considering the best results achieved. Note that, the bold values in Table 5 indicate the improved results achieved by the proposed HFAB algorithm.

As it is evident in Table 5, the proposed HFAB algorithm was able to hit the best known solution for 128 problems out of 133, which refers to a success level of 96.24%. Additionally, the HFAB algorithm has identified new best solutions for 8 problems, which refers to a 6.02% improvement. The identified new best results – with their tasks assignments

Table 5 Comparison of the HFAB algorithm against the heuristic methods

Problem name	Cycle time	Number of subgraphs	BKS	HFAB	Problem name	Cycle time	Number of subgraphs	BKS	HFAB
Gunther	7	5	14	14	Tonge	160	5	23	23
		8	14	14			8	23	23
		11	14	14			11	23	23
	44	5	12	12		176	5	21	21
		8	12	12			8	21	21
		11	12	12			11	21	21
	49	5	11	11		207	5	18	18
		8	11	11			8	18	18
		11	11	11			11	18	18
	61	5	9	9		251	5	15	15
		8	9	9			8	15	15
		11	9	9			11	15	15
	81	5	7	7		320	5	12	12
		8	7	7			8	12	12
		11	7	7			11	12	12
Kilbridge	57	5	10	10	Arcus2	5785	5	27	27
		8	10	10			8	27	27
		11	10	10			11	27	27
	79	5	7	7		6540	5	24	24
		8	7	7			8	24	24
		11	7	7			11	24	24
	92	5	6	6		7916	5	20	20
		8	6	6			8	20	20
		11	6	6			11	20	20
	138	5	4	4		9400	5	17	17
		8	4	4			8	17	17
		11	4	4			11	17	17
	184	5	3	3		11,570	5	14	14
		8	3	3			8	14	14
		11	3	3			11	14	14
Hahn	2004	5	8	8	Bartholdi	403	5	15	15
		8	8	8			8	15	15
		11	8	8			11	15	15
	2338	5	7	7		470	5	13	13
		8	7	7			8	13	13
		11	7	7			11	13	13
	2806	5	6	6		564	5	11	11
		8	6	6			8	10	11
		11	6	6			11	10	11
	3507	5	5	5		705	5	8	9
		8	5	5			8	8	8
		11	5	5			11	8	8
	4676	5	4	4		805	5	8	8
		8	4	4			8	7	8
		11	8	**4**			11	15	**8**

Table 5 (continued)

Problem name	Cycle time	Number of subgraphs	BKS	HFAB	Problem name	Cycle time	Number of subgraphs	BKS	HFAB
Warnecke	54	5	32	**31**	Scholl	1394	5	51	51
		8	32	**31**			8	51	51
		11	32	**31**			11	51	51
	62	5	28	**27**		1584	5	45	46
		8	28	**27**			8	45	45
		11	28	**27**			11	45	45
	74	5	22	22		1699	5	42	42
		8	22	22			8	42	42
		11	22	22			11	42	42
	92	5	18	18		2049	5	35	35
		8	18	18			8	35	35
		11	18	18			11	35	35
	111	5	15	15		2787	5	26	26
		8	15	15			8	26	26
		11	15	15			11	26	26
Bowman	20	5	5	5	Mitchell	14	5	8	8
Mansor	48	5	4	4		21		5	5
	62		3	3		35		3	3
	94		2	2		14	8	8	8
Buxey	30	8	12	12		21		5	5
	36		10	10		35		3	3
	54		7	7					

BKS Best Known Solution achieved by one of the heuristic methods

– are given in Appendix in detailed. As a summary, HFAB was able to solve all the problems successfully and therefore we can conclude that the HFAB is a powerful algorithm in solving ASALBP–1 instances.

5 Conclusions

The alternative subgraphs are applicable in the processes of toy manufacturing, commercial hand-held drills production, assembling car dashboards and disassembling complex products. This paper aimed at developing a novel hybrid metaheuristic algorithm to solve the ASALBP–1, which formally describes the alternative subgraphs for the assembly lines, as effectively as possible. Firstly, the firefly and bat algorithms were modified so as to be applicable for a combinatorial optimization problem, since both algorithms were originally designed to solve continuous optimization problems. After that, these two algorithms were combined within the hybrid metaheuristic framework to constitute an effective search algorithm for ASALBP–1. As it is obviously known from the literature on the metaheuristics and hybrid metaheuristics, an effective search algorithm must realize a balanced use of the diversification and intensification strategies. In line with this phenomenon, the proposed hybrid metaheuristic algorithm employed firefly and bat algorithms respectively to diversify and intensify its search. The proposed hybrid metaheuristic algorithm was firstly compared against the standard firefly and bat algorithms to observe whether it could improve the search capabilities of the firefly and bat algorithms in solving ASALBP–1 instances. After concluding the satisfactory performance of the HFAB algorithm against the firefly and bat algorithms, it was compared against some formerly developed heuristic methods. The proposed HFAB algorithm was able to hit the best known solutions for all the ASALBP–1 instances. Moreover, the HFAB algorithm identified new best solutions for 8 problems out 133.

To summarize, computational results indicate that the proposed HFAB algorithm is able to solve the ASALBP–1 instances with a satisfactory performance.

Future research directions may focus on designing matheuristics to tackle the ASALBP variants designing the proposed HFAB algorithm to solve assembly line balancing problem variants and other combinatorial optimization problems.

Appendix

See below Table 6.

Table 6 New Solutions

Sub assembly number	3		
Subgraph numbers	5–8-10		
Problem name- cycle time	Hann- 4676		
Task order	Station	Task time	Station Work-load
1	1	971	971
54	1	660	1631
55	1	660	2291
56	1	667	2958
11	1	82	3040
12	1	60	3100
13	1	112	3212
15	2	1556	1556
17	2	259	1815
14	2	420	2235
19	2	601	2836
22	2	70	2906
25	2	105	3011
20	2	80	3091
23	2	89	3180
26	2	330	3510
27	2	132	3642
28	2	69	3711
16	2	236	3947
21	2	80	4027
29	2	99	4126
24	2	89	4215
31	2	70	4285
34	2	70	4355
33	2	191	4546
30	2	70	4616
2	3	142	142
32	3	158	300
3	3	142	442
35	3	53	495
36	3	50	545
37	3	125	670
38	3	353	1023
39	3	70	1093
40	3	128	1221

Table 6 (continued)

Sub assembly number	3		
Subgraph numbers	5–8-10		
Problem name- cycle time	Hann- 4676		
Task order	Station	Task time	Station Work-load
41	3	65	1286
18	3	125	1411
57	3	1800	3211
58	3	300	3511
59	3	300	3811
60	3	300	4111
61	3	300	4411
62	4	900	900
63	4	800	1700
Sub assembly number	1		
Subgraph	4		
Problem name—cycle time	Warnecke—54		
Task order	Station	Task time	Station Work-load
5	1	35	35
28	1	12	47
1	2	10	10
3	2	41	51
22	3	37	37
13	3	12	49
6	4	17	17
16	4	33	50
14	5	52	52
18	6	7	7
12	6	34	41
15	6	12	53
9	7	14	14
11	7	33	47
33	8	34	34
19	8	15	49
2	9	53	53
17	10	44	44
32	11	47	47
27	12	16	16
20	12	13	29
30	12	22	51
21	13	29	29
8	13	23	52
23	14	43	43
26	14	9	52
25	15	24	24
24	15	23	47

Table 6 (continued)

Sub assembly number	1		
Subgraph	4		
Problem name—cycle time	Warnecke—54		
Task order	Station	Task time	Station Workload
29	16	26	26
34	16	23	49
31	17	51	51
36	18	52	52
44	19	37	37
37	19	12	49
7	20	34	34
41	20	15	49
40	21	7	7
39	21	44	51
42	22	13	13
35	22	12	25
43	22	29	54
45	23	43	43
46	24	23	23
49	24	16	39
50	24	12	51
47	25	24	24
51	25	26	50
38	26	33	33
52	26	12	45
48	26	9	54
53	27	52	52
54	28	12	12
55	28	33	45
56	29	44	44
57	29	7	51
58	30	15	15
4	30	36	51
10	31	52	52
Sub assembly number	1		
Subgraph number	3		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Workload
3	1	41	41
6	1	17	58
1	2	10	10
22	2	37	47
9	2	14	61
27	3	16	16
28	3	12	28

Table 6 (continued)

Sub assembly number	1		
Subgraph number	3		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Work-load
11	3	33	61
24	4	23	23
12	4	34	57
13	5	12	12
15	5	12	24
16	5	33	57
2	6	53	53
14	7	52	52
18	7	7	59
17	8	44	44
19	8	15	59
20	9	13	13
23	9	43	56
30	10	22	22
33	10	34	56
21	11	29	29
26	11	9	38
34	11	23	61
5	12	35	35
29	12	26	61
31	13	51	51
36	14	52	52
32	15	47	47
37	15	12	59
7	16	34	34
41	16	15	49
42	16	13	62
39	17	44	44
35	17	12	56
8	18	23	23
44	18	37	60
40	19	7	7
43	19	29	36
25	19	24	60
45	20	43	43
49	20	16	59
46	21	23	23
47	21	24	47
50	21	12	59
38	22	33	33
51	22	26	59
4	23	36	36
52	23	12	48
53	24	52	52

Table 6 (continued)

Sub assembly number	1		
Subgraph number	3		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Work-load
48	24	9	61
54	25	12	12
56	25	44	56
55	26	33	33
57	26	7	40
58	26	15	55
10	27	52	52
Sub assembly number	2		
Subgraph numbers	4–7		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Work-load
7	1	34	34
1	1	10	44
3	2	41	41
15	2	12	53
11	3	33	33
13	3	12	45
16	4	33	33
18	4	7	40
28	4	12	52
22	5	37	37
27	5	16	53
5	6	35	35
19	6	15	50
14	7	52	52
32	8	47	47
12	9	34	34
9	9	14	48
17	10	44	44
20	11	13	13
6	11	17	30
30	11	22	52
24	12	23	23
21	12	29	52
23	13	43	43
26	13	9	52
34	14	23	23
29	14	26	49
31	15	51	51
36	16	52	52
2	17	53	53

Table 6 (continued)

Sub assembly number	2		
Subgraph numbers	4–7		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Workload
37	18	12	12
33	18	34	46
41	19	15	15
8	19	23	38
35	19	12	50
44	20	37	37
42	20	13	50
39	21	44	44
40	21	7	51
43	22	29	29
25	22	24	53
45	23	43	43
48	23	9	52
46	24	23	23
47	24	24	47
51	25	26	26
49	25	16	42
50	25	12	54
38	26	33	33
52	26	12	45
53	27	52	52
54	28	12	12
55	28	33	45
56	29	44	44
57	29	7	51
4	30	36	36
58	30	15	51
10	31	52	52
Sub assembly numbers	2		
Subgraph numbers	4–6		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Workload
1	1	10	10
14	1	52	62
3	2	41	41
6	2	17	58
24	3	23	23
12	3	34	57
13	4	12	12
15	4	12	24
22	4	37	61

Table 6 (continued)

Sub assembly numbers	2		
Subgraph numbers	4–6		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Work-load
11	5	33	33
28	5	12	45
9	5	14	59
17	6	44	44
20	6	13	57
16	7	33	33
21	7	29	62
2	8	53	53
18	8	7	60
27	9	16	16
30	9	22	38
8	9	23	61
26	10	9	9
44	10	37	46
19	10	15	61
23	11	43	43
5	12	35	35
34	12	23	58
33	13	34	34
29	13	26	60
31	14	51	51
36	15	52	52
32	16	47	47
37	16	12	59
25	17	24	24
7	17	34	58
39	18	44	44
41	18	15	59
42	19	13	13
35	19	12	25
40	19	7	32
43	19	29	61
45	20	43	43
49	20	16	59
4	21	36	36
46	21	23	59
50	22	12	12
47	22	24	36
51	22	26	62
48	23	9	9
38	23	33	42
52	23	12	54
53	24	52	52
54	25	12	12

Table 6 (continued)

Sub assembly numbers	2		
Subgraph numbers	4–6		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Work-load
56	25	44	56
55	26	33	33
57	26	7	40
58	26	15	55
10	27	52	52
Sub assembly number	3		
Subgraph numbers	5–6–9		
Problem name—cycle time	Warnecke—54		
Task order	Station	Task time	Station Work-load
2	1	53	53
28	2	12	12
3	2	41	53
22	3	37	37
27	3	16	53
5	4	35	35
6	4	17	52
32	5	47	47
16	6	33	33
18	6	7	40
20	6	13	53
24	7	23	23
8	7	23	46
33	8	34	34
19	8	15	49
30	9	22	22
21	9	29	51
26	10	9	9
23	10	43	52
34	11	23	23
29	11	26	49
31	12	51	51
36	13	52	52
37	14	12	12
59	14	42	54
41	15	15	15
7	15	34	49
44	16	37	37
42	16	13	50
39	17	44	44
35	18	12	12
40	18	7	19

Table 6 (continued)

Sub assembly number	3		
Subgraph numbers	5–6–9		
Problem name—cycle time	Warnecke—54		
Task order	Station	Task time	Station Work-load
43	18	29	48
45	19	43	43
63	20	43	43
46	21	23	23
25	21	24	47
47	22	24	24
51	22	26	50
50	23	12	12
62	23	42	54
48	24	9	9
38	24	33	42
52	24	12	54
53	25	52	52
54	26	12	12
61	26	42	54
60	27	42	42
49	28	16	16
55	28	33	49
56	29	44	44
57	29	7	51
58	30	15	15
4	30	36	51
10	31	52	52

Sub assembly number	3		
Subgraph numbers	2–7–10		
Problem name—cycle time	Warnecke—62		
Task order	Station	Task time	Station Workload
28	1	12	12
6	1	17	29
16	1	33	62
18	2	7	7
2	2	53	60
1	3	10	10
3	3	41	51
22	4	37	37
24	4	23	60
9	5	14	14
19	5	15	29
11	5	33	62
14	6	52	52
23	7	43	43
15	7	12	55
12	8	34	34
13	8	12	46

Table 6 (continued)

Sub assembly number	3
Subgraph numbers	2–7–10
Problem name—cycle time	Warnecke—62

Task order	Station	Task time	Station Workload
27	8	16	62
17	9	44	44
20	9	13	57
21	10	29	29
26	10	9	38
30	10	22	60
33	11	34	34
29	11	26	60
5	12	35	35
34	12	23	58
35	13	12	12
32	13	47	59
31	14	51	51
36	15	52	52
7	16	34	34
37	16	12	46
41	16	15	61
42	17	13	13
39	17	44	57
8	18	23	23
44	18	37	60
40	19	7	7
25	19	24	31
43	19	29	60
45	20	43	43
49	20	16	59
38	21	33	33
46	21	23	56
4	22	36	36
47	22	24	60
10	23	52	52
48	23	9	61
64	24	50	50
65	25	50	50
66	26	50	50
50	26	12	62
67	27	43	43
58	27	15	58

Sub assembly number	3
Subgraph number	2–8–9
Problem name—cycle time	Bartholdi–805

Task number	Station	Task time	Station Workload	Task number	Station	Task time	Station Workload
2	1	30	30	68	5	14	142
3	1	7	37	113	5	11	153
4	1	47	84	98	5	34	187
1	1	16	100	91	5	115	302
6	1	8	108	120	5	31	333

Table 6 (continued)

Sub assembly number	3
Subgraph number	2–8–9
Problem name—cycle time	Bartholdi–805

Task number	Station	Task time	Station Workload	Task number	Station	Task time	Station Workload
8	1	37	145	95	5	20	353
9	1	32	177	132	5	36	389
10	1	29	206	72	5	25	414
5	1	29	235	134	5	20	434
7	1	39	274	142	5	148	582
14	1	15	289	116	5	31	613
16	1	53	342	100	5	39	652
15	1	53	395	71	5	118	770
17	1	8	403	55	5	7	777
19	1	24	427	76	5	5	782
18	1	24	451	133	5	23	805
20	1	8	459	105	6	58	58
24	1	13	472	117	6	32	90
23	1	14	486	121	6	32	122
22	1	8	494	92	6	35	157
21	1	7	501	128	6	8	165
26	1	25	526	147	6	78	243
32	1	10	536	148	6	78	321
33	1	14	550	135	6	46	367
34	1	41	591	101	6	30	397
35	1	42	633	129	6	11	408
28	1	25	658	58	6	52	460
25	1	10	668	86	6	21	481
27	1	11	679	130	6	27	508
29	1	11	690	42	6	16	524
31	1	25	715	103	6	13	537
36	1	47	762	43	6	32	569
61	1	3	765	122	6	26	595
62	1	8	773	102	6	26	621
37	1	7	780	127	6	55	676
152	2	403	403	44	6	66	742
45	2	80	483	118	6	26	768
38	2	80	563	123	6	19	787
39	2	7	570	96	7	31	31
40	2	41	611	126	7	48	79
74	2	40	651	11	7	17	96
59	2	14	665	49	7	47	143
75	2	101	766	136	7	64	207
46	2	7	773	77	7	28	235
47	3	41	41	157	7	105	340
56	3	28	69	12	7	11	351
154	3	105	174	13	7	32	383
153	3	104	278	30	7	29	412
155	3	403	681	138	7	15	427
64	3	33	714	93	7	26	453

Table 6 (continued)

Sub assembly number	3						
Subgraph number	2–8–9						
Problem name—cycle time	Bartholdi–805						
Task number	Station	Task time	Station Workload	Task number	Station	Task time	Station Workload
54	3	25	739	88	7	23	476
70	3	11	750	137	7	22	498
73	3	40	790	139	7	34	532
90	4	19	19	114	7	19	551
63	4	16	35	140	7	22	573
65	4	8	43	124	7	14	587
111	4	23	66	115	7	14	601
112	4	162	228	125	7	19	620
141	4	151	379	87	7	47	667
144	4	170	549	104	7	45	712
145	4	137	686	131	7	18	730
66	4	18	704	41	7	47	777
99	4	51	755	78	7	8	785
156	5	115	115	119	8	55	55
60	5	3	118	48	8	13	68
67	5	10	128	57	8	12	80
				94	8	46	126
				89	8	13	139
				97	8	19	158

Funding No funding.

Availability of data and material (data transparency) Available if it will be requested.

Code availability Available if it will be requested.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper**.**

Ethics approval Verbal informed consent was obtained from all subjects before the study.

Consent to participate Ümmühan Palamut**:** Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation, Validation, Writing- Reviewing and Editing. Şener Akpinar**:** Conceptualization, Methodology, Software, Data curation, Writing- Original draft preparation, Validation, Writing- Reviewing and Editing, Supervision.

Consent for publication The signed Consent ensures that the Publisher has the Author's permission to publish the relevant Contribution.

References

1. Scholl A (1999) Balancing and sequencing assembly lines, 2nd edn. Physica-Verlag HD, Heidelberg
2. Baybars I (1986) A survey of exact algorithms for the simple assembly line balancing problem. Manage Sci 32:909–932. https://doi.org/10.1287/mnsc.32.8.909
3. Becker C, Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. Eur J Oper Res 168(3):694–715. https://doi.org/10.1016/j.ejor.2004.07.023
4. Boysen N, Fliedner M, Scholl A (2007) A classification of assembly line balancing problems. Eur J Oper Res 183:674–693. https://doi.org/10.1016/j.ejor.2006.10.010
5. Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solutionapproaches. Int J Prod Econ 142(2):259–277. https://doi.org/10.1016/j.ijpe.2012.10.020
6. Pinto P, Dannenbring D, Khumawala B (1983) Assembly line balancing with processing alternatives: an application. Manage Sci 29:817–830. https://doi.org/10.1287/mnsc.29.7.817
7. Das S, Nagendra P (1997) Selection of routes in a flexible manufacturing facility. Int J Prod Econ 48:237–247. https://doi.org/10.1016/S0925-5273(96)00106-5
8. Bukchin J, Tzur M (2000) Designs of flexible assembly line minimize equipment cost. Institute Ind Eng Transact 32:585–598. https://doi.org/10.1080/07408170008967418
9. Senin N, Gropetti R, Wallace DR (2000) Concurrent assembly planning with genetic algorithms. Robot Comput Integrat Manuf 16:65–72. https://doi.org/10.1016/S0736-5845(99)00058-7

10. Capacho L, Pastor R (2005) ASALBP: The alternative subgraphs assembly line balancing problem. Technical Report: IOC−DT−P−2005−5. UPC. Barcelona, Spain
11. Capacho L, Pastor R (2008) ASALBP: The alternative subgraphs assembly line balancing problem. Int J Prod Res 46(13):3503–3516. https://doi.org/10.1080/00207540701197010
12. Yang XS (2008) Nature – inspired metaheuristic algorithm, 2nd edn. Luniver Press, England
13. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65–74). Springer, Berlin, Heidelberg
14. Capacho L, Pastor R (2006) The ASALB problem with processing alternatives involving different tasks: definition, formalization and resolution. In International Conference on Computational Science and Its Applications (pp. 554–563). Springer, Berlin, Heidelberg
15. Park K, Park S, Kim W (1997) A heuristic for an assembly line balancing problem with incompatibility, range, and partial precedence constraints. Comput Ind Eng 32(2):321–332. https://doi.org/10.1016/S0360-8352(96)00301-4
16. Capacho L, Pastor R, Guschinskaya O, Dolgui A (2006) Heuristic methods to solve the alternative subgraphs assembly line balancing problem. In 2006 IEEE international conference on automation science and engineering (pp. 501–506). IEEE
17. Capacho L, Guschinskaya O, Dolgui A, Pastor R (2006) Approximation methods to solve the alternative subgraphs assembly line balancing problem. Technical Report G2I-EMSE 2006–500–003, Ecole des Mines de Saint Etienne, France
18. Capacho L, Guschinskaya O, Dolgui A, Pastor R (2006) A comprehensive comparative analysis of heuristic methods for the alternative subgraphs assembly line balancing problem. Research Report: G2I-EMSE 2006–500–005, Ecole des Mines de Saint Etienne, France.
19. Capacho L, Pastor R, Dolgui A, Guschinskaya O (2009) An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. J Heurist 15(2):109–132. https://doi.org/10.1007/s10732-007-9063-x
20. Scholl A, Boysen N, Fliedner M (2009) Optimally solving the alternative subgraphs assembly line balancing problem. Ann Oper Res 172:243–258. https://doi.org/10.1007/s10479-009-0578-4
21. Capacho L, Pastor R (2011) A metaheuristic approach to solve the alternative subgraphs assembly line balancing problem. In Assembly Line-Theory and Practice (pp. 554–563). IntechOpen
22. Leiber D, Vuong AT, Reinhart G (2022) Alternative subgraphs assembly line balancing problem with resource selection and parallel stations. Eng Optim 54(11):1903–1918. https://doi.org/10.1080/0305215X.2021.1964493
23. Agarwal T, Kumar V (2022) A systematic review on bat algorithm: Theoretical foundation, variants, and applications. Arch Comput Methods Eng 29:2707–2736. https://doi.org/10.1007/s11831-021-09673-9
24. Kumar V, Kumar D (2021) A systematic review on firefly algorithm: past, present, and future. Arch Comput Methods Eng 28:3269–3291. https://doi.org/10.1007/s11831-020-09498-y
25. Oesterle J, Amodeo L, Yalaoui F (2019) A comparative study of multi-objective algorithms for the assembly line balancing and equipment selection problem under consideration of product design alternatives. J Intell Manuf 30:1021–1046. https://doi.org/10.1007/s10845-017-1298-2
26. Battaïa O, Dolgui A (2022) Hybridizations in line balancing problems: a comprehensive review on new trends and formulations. Int J Prod Econ 50:108673. https://doi.org/10.1016/j.ijpe.2022.108673
27. Arunarani AR, Manjula D, Sugumaran V (2017) FFBAT: A security and cost-aware workflow scheduling approach combining firefly and bat algorithms. Concurrency Comput Pract Exper 29(24):e4295. https://doi.org/10.1002/cpe.4295
28. Sureshkumar T, Lingaraj M, Anand B, Premkumar T (2018) Hybrid firefly bat algorithm (HFBA)–based network security policy enforcement for PSA. Int J Commun Syst 31(14):e3740. https://doi.org/10.1002/dac.3740
29. Chen G, Qian J, Zhang Z, Sun Z (2019) Multi-objective optimal power flow based on hybrid firefly-bat algorithm and constraints-prior object-fuzzy sorting strategy. IEEE Access 7:139726–139745. https://doi.org/10.1109/ACCESS.2019.2943480
30. Guo L, Wang G-G, Wang H, Wang D (2013) An effective hybrid firefly algorithm with harmony search for global numerical optimization. Scient World J Article ID 125625, 9 pages. https://doi.org/10.1155/2013/125625
31. Yang XS, He X (2013) Bat algorithm: literature review and applications. Int J Bio-inspir Comput 5(3):141–149. https://doi.org/10.1504/IJBIC.2013.055093
32. Helgeson WB, Birnie DP (1961) Assembly line balancing using the ranked positional weight technique. J Ind Eng 12(6):394–398
33. Taguchi G (1986) Introduction to Quality engineering: designing quality into products and processes, 1st ed., White Plains: Asian Productivity Organization

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.