Engineering and Technology Journal
Journal homepage: https://etj.uotechnology.edu.iq

Solving Mixed-Model Assembly Lines Using a Hybrid of Ant Colony Optimization and Greedy Algorithm

Huthaifa Al-Khazraji [a*], **Sohaib Khlil** [b], **Zina Alabacy** [c]

[a,c] Control and System Engineering Dept.,University of Technology-Iraq, Alsina'a street, P.O Box 10066 Baghdad, Iraq.
[b]Department of Applied Mechanics, University of Technology-Iraq, Alsina'a street, P.O. Box 10066 Baghdad, Iraq.

*Corresponding author Email: 60141@uotechnology.edu.iq

HIGHLIGHTS

- Mixed-model assembly line problem is a type of assembly line balancing problem at which two or more models of the same product are assembled sequentially at the same line.
- A hybrid of an Ant Colony Optimization and a Greedy Algorithm (Ant-Greedy) is presented for mixed-model assembly line balancing problem.
- The proposed approach is compared with the performance of the Merging Shortest and Longest Operation (MMSLO) method.

ARTICLE INFO

Handling editor: Muhsin J. Jweeg
Keywords:
Mixed-Model Assembly Line
Ant Colony Optimization
Greedy Algorithm
Automatic Changeover

ABSTRACT

The assembly line balancing problem deals with the assignment of tasks to work stations. Mixed-model assembly line problem is a type of assembly line balancing problem at which two or more models of the same product are assembled sequentially at the same line. To achieve optimality and efficiency of solving this problem, tasks at each work station have to be well balanced satisfying all constraints. This paper deals with the mixed-model assembly line balancing problem (MALBP) in which the objective is to minimize the cycle time for a given number of work stations. The problem is solved by using a hybrid of an ant colony optimization and a greedy algorithm (Ant-Greedy). MATLAB Software is used to perform the proposed method. Then, the proposed method is applied to a real case problem found in the literature for the assembly line of automatic changeover in the Electronic Industries Company in Iraq. The results of the proposed method are compared with the performance of the Merging Shortest and Longest Operation (MMSLO) method. The comparison shows that the Ant-Greedy optimization method is more efficient, where the efficiency increased from 93.53% for MMSLO method to 97.26% for the Ant-Greedy method.

1. Introduction

The line balancing (LP) problem is one of the considerable importance problems in many industrial applications. To assembly a product, the assembly process is divided into a number of tasks where a set of task among all the tasks are grouped into work stations. These work stations are connected together by a material handling system (i.e. conveyor). The sequence of work stations is designed same for every product based on the precedence diagram. A precedence diagram is a diagram illustrating the precedence relations between the tasks of a product. The line balancing problem concerns with the assignment of these tasks to the successive work stations efficiently in the line, without violating any precedence relations [1].

Assembly lines balancing (ALB) problem can be classified based on the variety of models into Single-Model Assembly Line Balance (SMALB) and Mixed-Model Assembly Line Balance (MMALB). In the SMALB, only one model of a product is assembled, whereas more than one model of a product is assembled in the MMALB [2]. A MMALB is more complicated to balance than a single-model line because of the difference in time between models. The advantages of using MMALB against SMALB are to get more flexible product design and reduction in capital expenditure [3]. In some cars industries, for example, most of the cars have a different number of models with different features. Therefore, car producers produce several models of the same car in the assembly line [4].

There are mainly two methods used to transform MMALB into SMALB: Combined Precedence Diagram (CPD) and Adjusted Task Times (ATT) [5]. The MMALB can be classified based on objectives into two different types [6]:

MMALB-I: minimizes the number of workstations, for given cycle time.

http://doi.org/10.30684/etj.v40i1.2153
Received 03 May 2021; Accepted 02 june 2021; Available online 25 January 2022

MMALB-II: minimizes the cycle time, for a given number of workstations.

The concept of the assembly line was started in 1901 when Mr. Ransom Olds register the production system of the Olds Motor car company by a patent as "assembly line". A significant innovation in the industrial assembly line was done in 1913 by Henry Ford's and the famous model-T [3]. The first mathematical model of the assembly line problem was established by Salveson [7]. Later, several extensions of the problem have been developed. In order to respond to the diversity of customer needs, Thomopoulos [8] studied the MMALB problem at which two or more models of the same product are assembled sequentially at the same line. The MMALB can be divided into two groups, named MMALB-I and MMALB-II, depending on the objective function. The MMALB-I class of problem minimizes the number of workstations, for given cycle time, whereas the MMALB-II class of problem minimizes the cycle time, for a given number of workstations. Several approaches have been presented in the literature to both types of the MMALB problems. These approaches can be classified into three categories: tradition approaches, heuristics and meta-heuristics, and hybrid approaches. A summary of different approaches that applies to solve MMALB is summarized in Table 1. The summary is classified according to the objectives, solution technique and line configuration of the study. For solving MMALB-I problem, Vilarinho and Simaria [9] proposed an ant colony optimization to parallel workstations line configuration. Hwang and Katayama [10] proposed genetic algorithms for a U-shaped assembly line system. Simulated annealing was proposed by Özcan et al. [11] of the parallel assembly line balancing. Yagmahan [12] developed an ant colony optimization for a straight line configuration.

A hybrid approaches for parallel line were presented by Akpınar and Bayhan [4] and Akpınar et al. [13]. On other hand, a MMALB-II problem with parallel workstations have been solved with different approaches such as mathematical programming and iterative genetic algorithm [14], genetic algorithm [15], and Linear programming [3]. Besides, Çil et al. [16] developed a heuristic algorithm based on beam search robotic MMALB-II problem. Later, Çil et al. [17] implemented bee algorithm (BA) and artificial bee colony (ABC) algorithm for MMALB-II problem that human workers and robots are collaborated in the assembly line. Finally, Zhang et al. [18] proposed an improved particle swarm optimization (PSO) with simulated annealing (SA) for a parallel model of MMALB with three objectives. The objectives are load balance between stations, dynamic balance in different working states and station's internal balance.

The objective of this study is to solve MMALB-II by using hybrid ant colony optimization and greedy algorithm (ACG). The new optimization is examined by a practical problem taken from the assembly line of automatic changeover in the Electronic Industries Company in Iraq.

2. Problem Description

In this section the definition of the MMALB is presented. The MMALB is to determine a possible assignment of tasks to assemble a product has different models to a sequence of work stations such that the precedence relations of each model are satisfied and objectives are optimized. A set of M models $(m = 1, 2, \dots, M)$ for similar product with set of N_m tasks related to each model $(i_m = 1, 2, \dots, N_m)$ are given. Each task (i_m) has an operation times (t_{i_m}) related to each model (m). Compared to SMALB, MMALB is more complicated since each model has its own precedence diagram. However, one way of transforming MMALB into SMALB can be done by combining all the precedence diagrams of all models into only one precedence diagram [12]. After combining all the precedence diagrams, the result combined model has only one precedence diagram with one number of N tasks $(i = 1, \dots, N)$ and an operation times (t_i). The operation times (t_i) in the joint precedence diagram are calculated as follows:

$$t_i = \sum_{m=1}^{M} q_m \times t_{i_m} \quad (1)$$

where q_m is the number of units of model m being assembled. The objective here in this paper is to minimize cycle time needed to assemble a product in a line given the number of stations (NS). The solution procedure now is the same as SMALB as follows:

Table 1: Summary of researchers on the mixed-model assembly line balance problem

Publications	Objectives	Solution Technique	Line configuration
Simaria and Vilarinho (2004)	MMALB-II	Mathematical programming, Genetic algorithm	Parallel
Vilarinho and Simaria (2006)	MMALB-I	Ant colony optimization	Parallel
Hwang and Katayama (2009)	MMALB-I	Genetic algorithms	U-shaped
Özcan et al (2010)	MMALB-I	Simulated annealing	Parallel
Yagmahan (2011)	MMALB-I	Ant colony optimization	Straight
Akpınar and Bayhan (2011)	MMALB-I	Hybrid genetic algorithm	Parallel
Akpınar et al (2013)	MMALB-I	Genetic algorithms, Ant colony optimization	Parallel
Raj et al (2016)	MMALB-II	Genetic algorithms	Parallel
Çil et al (2017)	MMALB-II	Beam search	Straight
Ashraf and Abbas (2017)	MMALB-II	Linear programming	Parallel
Zhang et al (2019)	Multiple Objectives	particle swarm optimization, simulated annealing	Parallel
Çil et al (2020)	MMALB-II	Bee algorithm, Artificial Bee Colony	Straight
Proposed work	MMALB-II	Ant colony optimization, greedy algorithm	Straight

The process time of each workstation (pw_j) is the time needed to finish all the tasks within a workstation (W_j), it is determined by:

$$pw_j = \sum_{i \in W_j} t_i \quad (2)$$

The cycle time (c) is the highest time among all the process time of each workstation, it is calculated by:

$$c = \text{Max}\{pw_j\}, \qquad j = 1, .., NS \quad (3)$$

The idle time (IT) is the difference time between processing time among workstations (unproductive time). It is used as an index to measure the performance of line and can be calculated as:

$$IT = (NS \times c) - \sum_{i=1}^{N} t_i \quad (4)$$

Another index is line efficiency, line efficiency (η) is compute by dividing the ratio between total processing time in all workstations to the result of the cycle time (c) multiply by the number of workstation (NS) as follows:

$$\eta = \frac{\sum_{i=1}^{N} t_i}{c \times NS} \quad (5)$$

For solving MMABL, these assumptions are considered:

1. The demand of each model is known.
2. The operation time of all tasks is fixed.
3. The changeover time between models is neglected.
4. The number of work station is given.

3. Methodology

The methodology to find the best solution of the MMALB problem can be summarized as follows:

1) A characteristic table of each model is constructed. In this table. a description of each task associated with the processing time of each task are illustrated.
2) A precedence diagram of each model is constructed. This diagram illustrates the relationships between tasks.
3) The MMABL problem is converted into the SMALB problem using joint precedence diagram as shown in Figure 1.
4) The theoretical cycle time (c_{th}) is determined. The c_{th} is obtained by dividing the total task time by the predefined number of workstations (NS). It is calculated as follows:

$$c_{th} = \frac{\sum_{i=1}^{n} t_i}{NS} \quad (6)$$

5) Each task is assigned to a workstation using ACG optimization approach. The optimization is explained in next section. The optimization is used to find the most appropriate set of tasks among all which are compatible with tasks that are already assigned. The process continues until the station is filled.
6) After calculating all the workstation process times (pw_j) as given in Eq. (2), the cycle time is determined as given in Eq. 3 (the highest pw_j among all workstations).
7) Finally, the idle time and line efficiency are computed based on Eq. (3) and Eq. (4).

4. ACG Optimization Approach

In this work, a hybrid of an ant colony optimization and a greedy algorithm (ACG) is proposed to solve MMALB-II problem. The structure of the proposed method is given in Figure 2. The problem is divided into two sub-problems. The first one is how to generate a different sequence of tasks without violating the precedence relations. This sub-problem is performed by using ant colony optimization. ACO algorithm is an algorithm inspired based on the behavior of real ant colonies. It is a population-based algorithm. As known that ants are being able to find the shortest path between their nest and a food source by chase pheromone trails exposed by other ants. For the trail that has more intense pheromone, the probability that an ant will follow it becomes higher [9]. However, in this work, ACO is used to perform the sequence of the task taking into account the precedence constraints. Here, the producer of applying ACO is slightly different form the usual use of this algorithm. The algorithm begins by generating n of ants equal to the number of tasks. As a result, each ant in the proposed algorithm represents a task. Instead of design route for each ant, here in this work, the goal is to sequence these ants based on the state transition rule given by Eq. (7) without violating the precedence relations.

$$p_i^t = \begin{cases} \frac{\tau_i}{\sum_{i \epsilon N^t} \tau_i} & if\ i \in N^t \\ 0 & if\ i \notin N^t \end{cases} \quad (7)$$

For each iteration t, the probability p of selecting the task i in the candidate sequence S is based on its pheromone trail τ_i. All the ant are initialized with the same amount of pheromone. The second sub-problem is how to group tasks for each sequence into workstation taking into account the objective to reduce the cycle time in order to improve the efficiency of the solution. This sub-problem is performed by using greedy algorithm. After receiving the candidate sequence S from ACO. The Greedy algorithm starts by assigning each task i to workstation W_j. For each workstation, the accumulated process time (pw_j) is computed. If the accumulated process time of the assigned workstation pw_j becomes more than c_{th}, the algorithm find shift time (ST). The ST is the deviation time between c_{th} and pw_j in two cases: the first case (ST_1) is the case when the accumulated process time of the workstation pw_j is less than theoretical cycle time which associated with task $i-1$. The second case (ST_2), is the case when the accumulated process time of the workstation pw_j is more than theoretical cycle time which associated with task i. If $ST_1 \leq ST_2$ then add just task $i-1$ to the workstation W_j and remove task i from W_j; otherwise add the task i to the workstation W_j. The optimization process is stopped if the entire task is assigned to workstation. For more explanation about this algorithm see [19].

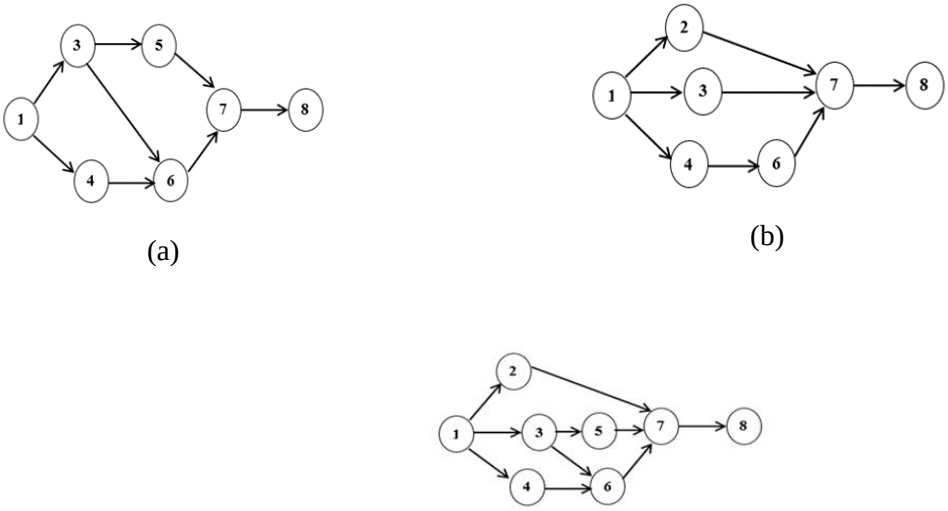

Figure 1: Precedence diagram of model (a), model (b) and joint model (c)

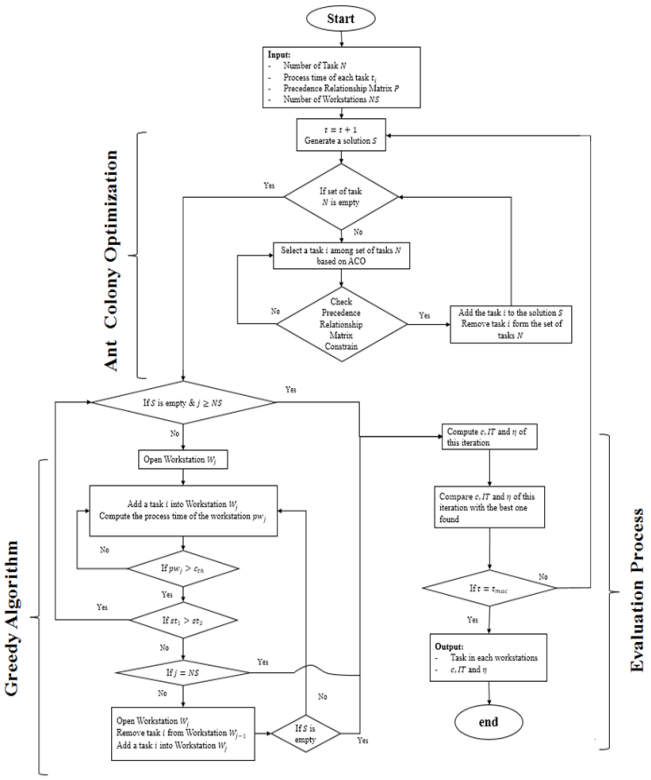

Figure 2: Structure of the Ant-Greedy optimization

5. Practical Application

The proposed algorithm is evaluated by using a real case study that was published by Al-Zubaidy et al. [5]. Their work was applied at the Electronic Industries Company (EIC) in Iraq for assembling varies models of changeover work automatically. Three different models of the changeover were assembled manually. The three models differ from each other as follows:

a. Model A which is an Automatic Changeover (80 A) with Timer.
b. Model B which is Automatic Changeover (30 A) without Timer.
c. Model C which is Automatic Changeover (30 A) with Timer.

The description, technological rout and task time of all models are shown in Table 2. Each model differs from the other in the task time for the same task and whether the task is required or not. As a result of that, each model has different assembly time. However, the total number of tasks for all models is 21 tasks. Model A has more parts than other models; therefore it required more time to assemble than other models. Figure 4 show the precedence diagram with the times of each task (minute) of this model. For model B, Figure 5 shows the precedence diagram of this model. Figure 6 shows the precedence diagram of this model C. The demand requested by the customer for each model has been observed as (2 of model A), (7 of model B) and (13 of model C). Using Eq. (1), Table 3 shows the combined model process time and Figure 7 shows the combined model diagram of all models.

Table 2: Total assembly tasks required for the Automatic Changeover assembly line [5]

Tas kNo.	Description of task	TA (min)	TB (min)	TC (min)
1	Soldering diodes on the circuit board	1.86	1.86	1.86
2	Soldering resistors on the circuit board	1.33	0.26	1.33
3	Soldering thermal resistors on the circuit board	0.33	0.33	0.33
4	Soldering transistors on the circuit board	0.73	-	0.73
5	Soldering wires on the circuit board	0.83	0.83	0.83
6	Soldering LED wires on the circuit board	2.26	1.7	1.41
7	Soldering electrical capacitors on the circuit board	0.46	0.35	0.58
8	Preparing and soldering LED to wires	-	-	0.83
9	Soldering ceramic capacitors on the circuit board	0.66	0.66	0.66
10	Soldering the relay (30A) on the circuit board	0.9	0.6	1.2
11	Soldering the relay (10A) on the circuit board	0.26	0.3	0.26
12	Connecting the neon lamps wires on the circuit board	1.33	1	1
13	Soldering the relay (120A) on the circuit board	16	-	-
14	Soldering pieces of wire to the circuit board	-	0.33	0.33
15	Making the circuit board robust for the relay (120A) area	4	-	-
16	Adjusting the circuit board on the base part of plastic box	3.33	0.25	0.25
17	Joined the terminal block (100A) to the plastic box	1.66	0.05	0.05
18	Adjusting the terminal block (60A) on the lower plastic box	-	0.05	0.05
19	Joined wires on the terminal block (100A)	1.33	1.16	1.16
20	Joined welded wires of relay (10A) on the terminal block (60A)	-	0.5	0.5
21	Joined the upper and lower plastic box	1	0.5	0.5

Table 3: Combined task time required for the Automatic Changeover assembly line

Task No.	t_{i_A}	t_{i_B}	t_{i_C}	$t_{i_A} \times 2$	$t_{i_B} \times 7$	$t_{i_C} \times 13$	t_i
1	1.86	1.86	1.86	3.72	13.02	14.18	40.92
2	1.33	0.26	1.33	2.66	1.82	17.29	21.77
3	0.33	0.33	0.33	0.66	2.31	4.29	7.26
4	0.73	-	0.73	1.46	-	9.49	10.95
5	0.83	0.83	0.83	1.66	5.81	10.79	18.26
6	2.26	1.7	1.41	4.52	11.9	18.33	34.75
7	0.46	0.35	0.58	0.92	2.45	7.54	10.91
8	-	-	0.83	-	-	10.79	10.79
9	0.66	0.66	0.66	1.32	4.62	8.58	14.52
10	0.9	0.6	1.2	1.8	4.2	15.6	21.6
11	0.26	0.3	0.26	0.52	2.1	3.38	6
12	1.33	1	1	2.66	7	13	22.66
13	16	-	-	32	-	-	32
14	-	0.33	0.33	-	2.31	4.29	6.6
15	4	-	-	8	-	-	8
16	3.33	0.25	0.25	6.66	1.75	3.25	11.66
17	1.66	0.05	0.05	3.32	0.35	0.65	4.32
18	-	0.05	0.05	-	0.35	0.65	1
19	1.33	1.16	1.16	2.66	8.12	15.08	25.86
20	-	0.5	0.5	-	3.5	6.5	10
21	1	0.5	0.5	2	3.5	6.5	12

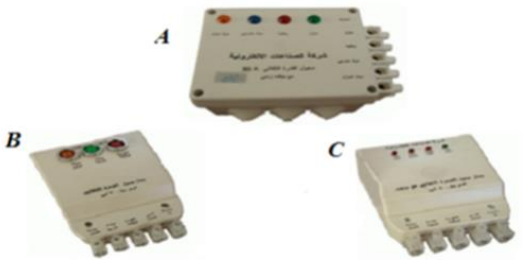

Figure 3: Models of the Automatic Changeover [5]

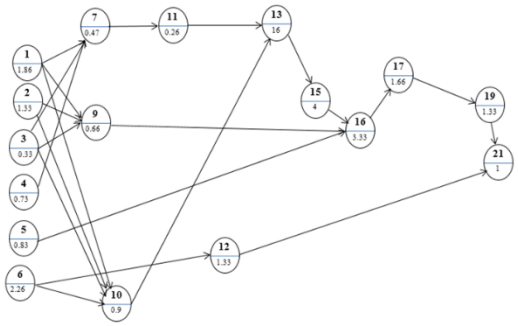

Figure 4: Precedence diagram of model

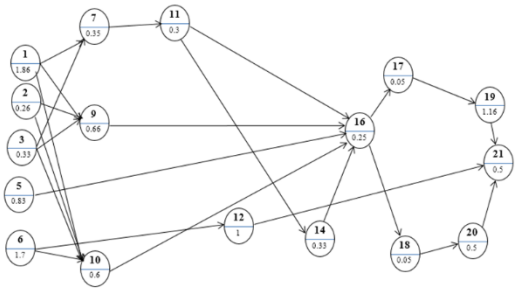

Figure 5: Precedence diagram of model B

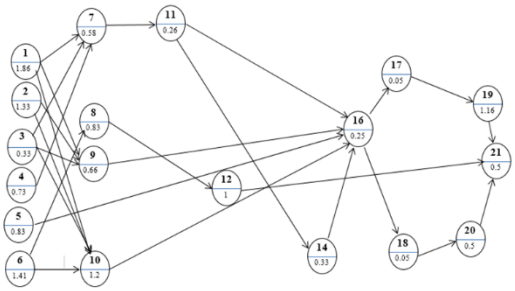

Figure 6: Precedence diagram of model C

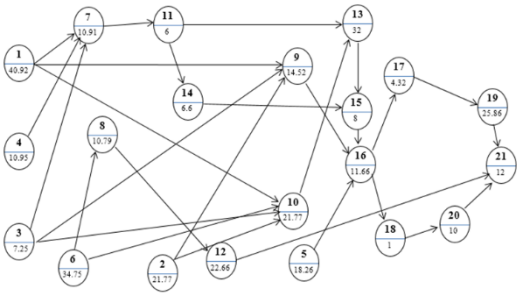

Figure 7: Precedence diagram of combined models

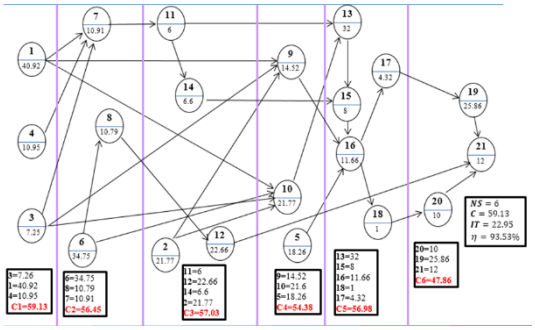

Figure 8: Distributing of tasks among the workstations in the line based on the MSLO method

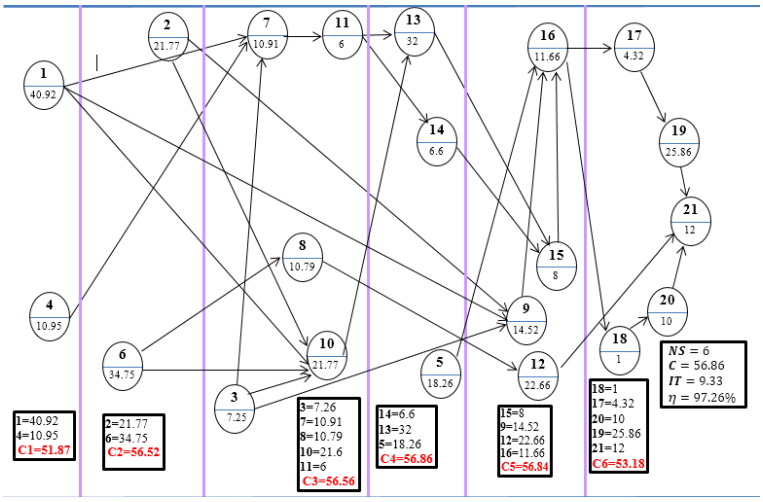

Figure 9: Distributing of tasks among the workstations in the line based on the Ant-Greedy optimization method

6. Results and Discussion

Zubaidy et al. [5] solve the mix-model of the Automatic Changeover at the Electronic Industries Company using a method named Merging Shortest and Longest Operation (MSLO). In this section, the results of using the proposed ACG optimization approach is compared with the results obtained from the MSLO method. Figure 8 shows the distributing of tasks among the workstations in the line based on the MSLO method.

On other hand, Figure 9 shows the distributing of tasks among the workstations in the line based on the ACG optimization. It can be noticed that the ACG algorithm achieves better performance in comparison with the MSLO method when the number of workstation same is equal to 6 ($NS = 6$) for both algorithms. Table 4 shows the performance comparison between the two methods. It can be seen that the cycle time is reduced to 56.86 min by using ACG in comparison with the cycle time of MSLO which is 59.13 min. As a result, the idle time is reduced to 9.33 min/cycle for the ACG in comparison with the idle time of MSLO which is 22.95 min/cycle. Besides the reduction in the idle time, the efficiency of the line is improved to become 97.26 % in comparison with the efficiency of the line by using MSLO which is 93.53%.

Table 4: Comparison between MSLO method and ACG optimization

Performance	MSLO [5]	ACG Optimization (proposed algorithm)
No. Workstations	6	6
Cycle time	59.13 min	56.86 min
Idle time	22.95 min/cycle	9.33 min/cycle
Efficiency	93.53%	97.26%

7. Conclusion

Designing a balanced assembly line is a critical step in the early stage when introducing new product lines. In this study, an approach based on a hybrid of an ant colony optimization and a greedy algorithm is developed. The developed approach is designed to obtain a good solution for the mixed-model assembly line when the objective is to minimize the cycle time, for a fixed number of workstations and improve the efficiency of the assembly line. The mixed-model assembly line problem is transferred into a single model assembly line problem using joint precedence diagram. The structure of the proposed method is divided into two sub-problems. The first sub-problem is performed by ant colony optimization to generate a different sequence of tasks without violating the precedence relations. The second sub-problem is performed by a greedy algorithm to group tasks into workstation taking into the account the objective is to reduce the cycle time in order to improve efficiency of the assemble line balance.

The new method is examined by a practical problem taken from the assembly line of automatic changeover in the Electronic Industries Company in Iraq. The outcomes of the new method are compared with the Merging Shortest and Longest Operation (MMSLO) method. The comparison shows the efficiency of the line increased from 93.53% for MMSLO method to 97.26% for the ACG method.

Acknowledgment

We would like to sincerely thank the reviewers for their constructive comments on our paper.

Author contribution

All authors contributed equally to this work.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author.

Conflicts of interest

The authors declare that there is no conflict of interest.

References

[1] J. I. Van Zante-de Fokkert and de T. G. Kok. The mixed and multi model line balancing problem: a comparison. European Journal of Operational Research, 100 (1997) 399-412.

[2] H. Güden and S. Meral. An adaptive simulated annealing algorithm-based approach for assembly line balancing and a real-life case study. The International Journal of Advanced Manufacturing Technology, 84 (2016) 1539-1559.

[3] S. R. Ashraf and S. N. Abbas. Mixed Model Assembly Line Balancing by Using of Sub-Assembly Parallel Shop. Proceedings of the First International Conference on Industrial Engineering and Management Applications, (2017) 93-97.

[4] S. Akpınar and G. M. Bayhan. A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. Engineering Applications of Artificial Intelligence, 24 (2011) 449-457.

[5] S. S. Al-Zubaidy, M. A. Mahmoud and I. D. Khalaf. Balancing Mixed-Model Assembly line in Electronic Industries Company. Engineering and Technology Journal, 34 (2016) 233-244.

[6] A. Scholl. Balancing and sequencing of assembly lines (No. 10881). Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL), 1999.

[7] M. E. Salveson. The Assembly Line Balancing Problem. Journal of Industrial Engineering, 6 (1955) 18–25.

[8] N. T. Thomopoulos. Line balancing-sequencing for mixed-model assembly. Management Science, 14 (1967) 59.

[9] P. M. Vilarinho and A. S. Simaria. ANTBAL: an ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. International journal of production research, 44 (2006) 291-303.

[10] R. Hwang and H. Katayama. A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. International Journal of Production Research, 47 (2009) 3797-3822.

[11] U. Özcan, H. Çerçioğlu, H. Gökçen and B. Toklu. Balancing and sequencing of parallel mixed-model assembly lines. International Journal of Production Research, 48 (2010) 5089-5113.

[12] B. Yagmahan. Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. Expert Systems with Applications, 38 (2011) 12453-12461.

[13] S. AkpıNar, G. M. Bayhan and A. Baykasoglu. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Applied Soft Computing, 13 (2013) 574-589.

[14] A. S. Simaria and P. M. Vilarinho. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. Computers & Industrial Engineering, 47 (2004) 391-407.

[15] A. V. Raj, J. Mathew, P. Jose and G. Sivan. Optimization of cycle time in an assembly line balancing problem. Procedia Technology, 25 (2016) 1146-1153.

[16] Z. A. Çil, S. Mete and K. Ağpak. Analysis of the type II robotic mixed-model assembly line balancing problem. Engineering Optimization, 49 (2017) 990-1009.

[17] Z. A. Çil, Z. Li, S. Mete, and E. Özceylan. Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration. Applied Soft Computing, 39 (2020) 106394.

[18] W. Zhang, L. Hou, Y.Gan, C. Xu, X. Bu, and H. Lin. Parallel Optimization of the Balancing and Sequencing for Mixed-model Assembly Lines. Manufacturing Technology, 19 (2019) 537-544.

[19] S. Khlil, H. Al-Khazraji, and Z. Alabacy, Solving Assembly Production Line Balancing Problem Using Greedy Heuristic Method. MS&E, 745 (2020) 012068.