Scientia Iranica E (2022) 29(4), 2084–2098

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
http://scientiairanica.sharif.edu

Optimization of two-sided assembly line balancing with resource constraints using modified particle swarm optimisation

M.R. Abdullah Make and M.F.F. Ab Rashid*

Department of Industrial Engineering, College of Engineering, Universiti Malaysia, Pahang, 26300 Kuantan, Malaysia.

Received 4 January 2019; received in revised form 29 July 2020; accepted 18 October 2020

KEYWORDS
Manufacturing systems;
Assembly line balancing;
Two-sided line;
Resource constraints;
Particle swarm optimization.

Abstract. Two-Sided Assembly Line Balancing (2S-ALB) is essential to the production of large-sized high-volume products, including automotive production, at assembly plants. The 2S-ALB problem involves different assembly resources such as worker skills, tools, and machines required for the assembly. This research modeled and optimized the 2S-ALB with resource constraints. In the end, besides favorable workload balance, the number of resources can be optimized. For optimization purpose, particle swarm optimization was modified to reduce dependence on a single best solution. This was conducted by replacing the best solution with the top three solutions in the reproduction process. Computational experiment results using 12 benchmark test problems indicated that the 2S-ALB with a resource-constrained model was able to reduce the number of resources for use in an assembly line. Furthermore, the proposed Modified Particle Swarm Optimization (MPSO) was capable of searching for minimum solutions to 11 out of 12 test problems. The good performance of MPSO was attributed to its ability to maintain particle diversity over iterations. The proposed 2S-ALB model and MPSO algorithm were validated later using an industrial case study. This research makes a two-fold contribution: (a) Proposition of a novel 2S-ALB with resource-constrained model and (b) a modified PSO algorithm with enhanced performance.

1. Introduction

Assembly line is a system that considers arrangement of workstations, workers, tools, or machines, and it successively outlines operations so as to reach completion. It is widely used in many manufacturing industries to cope with growing demands in manufacturing. The assembly line is set up for the most optimum design to meet production demands. The assembly line system was introduced around 1900 by Henry Ford for his automobile plants [1]. Since then, various evolutions and progresses in the assembly line were reported. Derived from the above idea, the balancing approach has been developed for the assembly line, known as Assembly Line Balancing (ALB). Balancing an assembly line can be difficult for most industries. It refers to not only assigning a task to a respective workstation but also enhancing production rate with the desired performance level [2]. Nowadays, ALB has become instrumental in coping with global competitiveness in the industry. It classically started in 1955 when Salveson firstly described the typical ALB problem that focused on an efficient and fast solution approach to solving the line balancing problem [3].

*. *Corresponding author. Tel.: +609-4246321; Fax: +609-4246222*
E-mail address: ffaisae@ump.edu.my (M.F.F. Ab Rashid)

doi: 10.24200/sci.2020.52610.2797

Great progresses occurring from time to time have extended classification of the ALB problem.

Later, various versions of ALB problems have been formulated to suit different assembly line problems [4]. One of the ALB branches is the assembly line that assembles large-sized and high-volume products like an automotive assembly line. The assembly process is conducted on both left and right sides of the product. This problem is known as Two-Sided Assembly Line Balancing (2S-ALB) and was first established by Bartholdi in 1993 [5]. Early investigations into 2S-ALB have inspired other researchers to further study and extend the above work to the next level. The 2S-ALB was built from a single-line production system, which is identically paired parallel to the first side of the assembly line. Figure 1 illustrates the 2S-ALB station features along the conveyor belt. Contrary to the one-sided line, the assembly process in 2S-ALB is conducted either from the left or right side, depending on various constraints. The 2S-ALB system is able to shorten and save space in the assembly lines, besides reducing the material handling of tools and fixtures.

Recently, the 2S-ALB problem has grown rapidly and different ALB versions have been adopted as variations of the 2S-ALB problem. The 2S-ALB variation began with the general 2S-ALB, as illustrated in Figure 1. The general 2S-ALB consists of two workstations facing each other along the assembly line. This version of the problem has its advantages, including shortening the assembly line, saving some spaces, reducing throughput time and material handling besides the cost of tools and fixtures. This general 2S-ALB has been well addressed in several research studies [3,6–9].

Besides studying the general 2S-ALB, researchers have combined 2S-ALB with mixed-Model Assembly Line Balancing (MALB). The MALB is particularly considered in levelling the workload in every workstation on the line, besides levelling the part usage. It literally functions in achieving a balanced workload at specific processing times for each assembly task while attempting to minimize the variation in different parts over time. The combination of 2S-ALB and MALB has led to the introduction of the implementation of different optimizations and line balancing solution approaches [10–12]. Another case of combination with

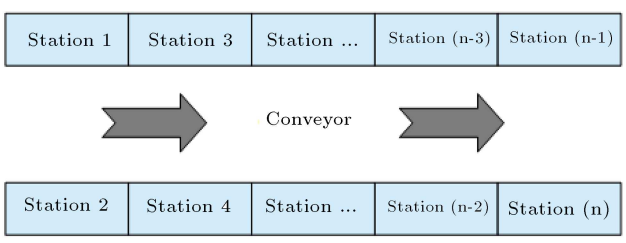

Figure 1. Two-sided assembly line.

the 2S-ALB is Parallel Assembly Line Balancing (P-ALB). The P-ALB is a combination of two or more lines placed parallel to each other, which has turned into sharing tools and fixtures to complete the entire job. The two-sided P-ALB, which is a combination of 2S-ALB and P-ALB, is to shorten the assembly line while steadily running during a breakdown [13–16]. This combination was discussed by Ozcan, Gokcen, and Toklu (2010) [17] to shed light on its many benefits:

(i) It can help produce similar products or different models of the same production on the adjacent lines;

(ii) It reduces the idle time and increases the efficiency of the assembly lines;

(iii) It is able to complete production with different cycle times for each of the lines;

(iv) It can improve visibility and communication skills between operators;

(v) It manages to reduce operator requirements.

Many studies have been conducted to work out the best optimum-seeking approach, that is implementing either heuristic or meta-heuristic method, for 2S-ALB. In an early study, Kim et al. (2000) used Genetic Algorithm (GA) as an optimization algorithm [18]. Then, in 2001, it was continued by Lee et al. employing the group assignment procedure [19]. The GA approach was also implemented by Delice et al. [20], Kucukkoc and Zhang [15], and Taha et al. [21] to optimize 2S-ALB. Meanwhile, Baykasoglu and Dereli adopted Ant Colony Optimization (ACO) to optimize the 2S-ALB [22]. They successfully applied the ACO algorithm for a domestic product, influencing other researchers to deal with other sectors apart from the large-sized automotive products. In addition, many other researchers have implemented the ACO because of its favorable performance, especially in dealing with combinatorial problems [15,23,24]. From earlier reviews, GA and ACO algorithms have successfully dominated other optimization methods in terms of performance and frequencies that make these algorithms more popular [13]. Besides, different algorithms have been implemented through several reported studies. For instance, Hu et al. [25] (2008) reported the implementation of the enumerative algorithm combined with the Hoffmann heuristic method.

In the meantime, Particle Swarm Optimization (PSO) algorithm was frequently implemented for 2S-ALB. The PSO assisted with Taguchi was implemented for 2S-ALB with multi-skilled worker assignment [26]. Researchers implemented ACO algorithm to optimize stochastic 2S-ALB instead of deterministic time in the majority of 2S-ALB works [27]. Meanwhile, in 2012, Chutima and Chimklai [11] proposed a PSO

with Negative Knowledge (PSONK) to optimize complex combination with the 2S-ALB problem. After implementing the PSONK by Delice et al. [28], they proposed a combined selection mechanism for the assembly task. Besides, different approaches were proposed to improve the PSO performance [28–30]. Although the advantages of PSO algorithm have been well reported, its application and improvement are still needed. Generally, PSO is known as a fast optimizer with a robust algorithm, which provides a high-quality solution. However, dedicating effort and focus to achieving a single best solution in PSO leads to premature convergence or local optimum. This phenomenon is described as a condition in which the convergence is stopped and is considered to express the solution as the best. This problem occurs when the algorithm attempts to determine the path of searching for a direction while still following the best earlier solution. Few parameter settings may cause a limitation in providing the best solution given that the PSO algorithm only requires a simple specification as a setting before generating a solution.

Despite many studies available on 2S-ALB, the majority of these works assumed that the assembly workstation had a similar capability to conduct the assembly process. In a real situation, there are various constraints that need to be considered during the assembly line design. For example, the workforce and machines have different skills and abilities in completing the assigned task. Proper utilization of resources depending on their skills and precedence has integrated the assembly line to be fully optimized. Besides, with proper use of the machine, one can solve the issue of inadequate space for the assembly line in allocating the required machines to the workstation [6].

In order to overcome the limitation, this paper considers the resources required to conduct a specific assembly task. By considering the assembly resource constraints, a number of resources can be optimized. For optimization purpose, the PSO is modified to reduce the dependence of the algorithm on a single best solution. The proposed modification is expected to improve the exploration ability of the algorithm. Section 2 of this paper presents the 2S-ALB with resource constraints. Section 3 presents the proposed Modified Particle Swarm Optimization (MPSO) algorithm. The computational experiment is set up and the results are discussed in Section 4. Finally, Section 5 summarizes and concludes the research work.

2. 2S-ALB with resource constraints

The 2S-ALB is a modified structure that is essentially formed from the one-sided ALB problem. The main objective of this problem is to enhance the production rate and increase the line efficiency. Flexibility to produce a large number of large-sized products in a two-sided assembly line configuration practically provides many beneficial advantages, including the ability to shorten the line length, save spaces on the lines, increase the line efficiency by reducing the number of workstations, and reduce the material handling cost of tools and fixture. Normally, on a two-sided assembly line, a pair of lines placed opposite to each other is represented. Figure 1 illustrates the two-sided assembly line possessing left and right sides of the lines in which the workstation is clamped together between the moving conveyors.

A comprehensive study intended for forming the notion of balancing 2S-ALB problem was presented [31]. Derived from a particular task relation called 'precedence relation graph' which is built using circles and arrows, the example of precedence relation graph with nine tasks is depicted in Figure 2. Each circle represents an assigned task, while the linked arrows represent each relation between the tasks. The associated data of each processing time and operational direction are specified on top of each circle (assign task). Three types of operational direction are considered: Left (L), Right (R), and Either (E). On left and right sides, the execution is outright and should be actualized for the following position. Meanwhile, for either direction side, the task could be executed on any side of the workstation, either on the left or right side.

Then, an assembly data is presented in the precedence matrix, as shown in Table 1. This matrix consists of one and zero values that represent the assembly relation information of the precedence graph. In Table 1, the relation of each task is transformed from the precedence relation graph, adopting 'i' and 'j' as the present and the next assigned tasks, respectively. The value of one in the precedence matrix indicates the predecessor link of 'i' task to the next task 'j'. This means that there is a precedence relationship to be examined. Meanwhile, the zero value implies no precedence relation between tasks i and j.

Besides the precedence matrix, a data matrix is also required to store the assembly information

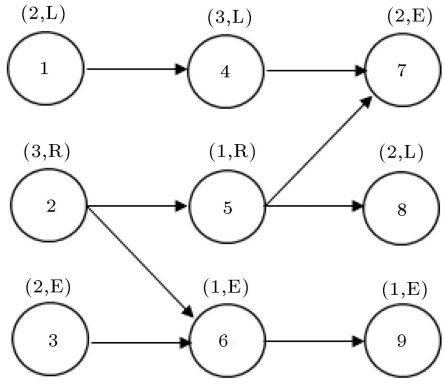

Figure 2. Precedence relation graph.

Table 1. Precedence matrix.

i/j	1	2	3	4	5	6	7	8	9
1	0	0	0	1	0	0	0	0	0
2	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

Table 2. Data matrix.

Task	Time	Side	Resources		
1	2	1	1	2	0
2	3	3	3	0	0
3	2	2	2	3	0
4	3	1	1	0	0
5	1	3	3	0	0
6	1	2	2	3	0
7	2	2	1	2	3
8	2	1	2	0	0
9	1	2	1	3	0

for the 2S-ALB with resource constraints. The data matrix (Table 2) expresses the assembly information such as processing time, assembly side, and resources details. For the side column, three different operational direction values indicate different sides. In this column, values '1', '2', and '3' are allocated to the left-side, either side, and value right-side operations, respectively. The resource details are coded in numbers in order to express different resources. It is important to note that the number of resources for one assembly task is not limited to three, as shown in Table 2. In cases where the number of resources is larger, the matrix can be expanded to fit the entire data.

2.1. Problem assumptions and notations

The general assumptions of the problem are as follows:

- Task times and resources used (machine, tools, and worker) are known and deterministic;
- Tasks have preferences regarding the operational directions (sides), i.e., left, either, or right sides;
- Every task can be done only upon the completion of all its immediate predecessors;
- The maximum operational cycle time is fixed and cannot be exceeded;
- Every task cannot be split between workstations and must be assigned to exactly one workstation;
- Tasks with positive zoning must be operated on the same workstation;
- Tasks with negative zoning could not be assigned to the same workstation;
- Parallel tasks and parallel stations are not allowed;
- The skill level of each worker is ignored to provide a similar working pace of assembly task;
- The working travel times are ignored and no inventory (work in progress) is allowed;
- Any machine and tool breakdowns are not considered and the assembly process is constantly performed.

The notations of this mathematical formulation are summarized below:

J	Number of mated workstations $j = 1, 2, ..., J$
I	Number of one-sided workstations $i = 1, 2, ..., I$
F	1, if there is any space available on the operating time, otherwise, 0
N	Number of resources utilization $n = 1, 2, ..., N$
X_{ms}	1, if the mated workstation j is utilized for both sides of the line, otherwise, 0
Y_s	1, if the mated workstation j is utilized for only one side of the line, otherwise, 0
m_t	Maximum processing time $t = 1, 2, ..., T$
r_t	Operational time of the task on the workstation j
p_v	Maximum gap value in space availability
q_v	Minimum gap value in space availability;
R_s	1, if resource is utilized in workstation j, otherwise, 0

2.2. Mathematical formulation and constraints

The mathematical model for 2S-ALB with resource constraints is presented below. In this problem, four optimization objectives are considered. The first optimization objective as in Eq. (1) is to minimize the mated workstation, f_1. The second optimization objective in Eq. (2) is to minimize the number of workstations, f_2. A mated workstation consists of a pair of left and right workstations on the assembly line. Meanwhile, the number of workstations calculates the total individual workstations. The third optimization objective is to minimize idle time, f_3, as presented in Eq. (3). Finally, the fourth optimization objective to minimize the number of resources, f_4, is presented in

Eq. (4). By using the number of resources as one of the optimization objectives, the number of resources can be minimized. This can be achieved by assigning the assembly task that uses a similar resource on one workstation.

$$f_1 = \sum_{j=1}^{J} X_{ms}, \tag{1}$$

$$f_2 = \sum_{j=1}^{J} 2JX_{ms} + \sum_{i=1}^{I} Y_s, \tag{2}$$

$$f_3 = \sum_{t=1}^{T} (m_t - r_t) + \sum_{t=1}^{T} F\,(p_v - q_v), \tag{3}$$

$$f_4 = \sum_{n=1}^{N} R_s, \tag{4}$$

$$\sum_{K} k\,(x_{ik_1} + x_{ik_2}) - \sum_{K} k\,(x_{jk_1} + x_{jk_2}) = 0$$

$$(i,j) \in ZP_{ij}, \tag{5}$$

$$\sum_{K} k\,(x_{ik_1} + x_{ik_2}) - \sum_{K} k\,(x_{jk_1} + x_{jk_2}) \neq 0$$

$$(i,j) \in ZN_{ij}, \tag{6}$$

$$\sum_{i=1}^{n} t_i x_{ijk} + s_{jk} \leq CT, \tag{7}$$

$$s_{jk} = \sum_{u=1}^{U} x_{ujk}\left(t^s_{u+1} - t^f_u\right) + \left(CT - t^f_u\right)$$

$$u \in Q_{jk}, \tag{8}$$

$$\sum_{k\in\{1,3,5,\ldots,m-1\}} x_{jk} = 1 \qquad \forall j \in L, \tag{9}$$

$$\sum_{k\in\{2,4,6,\ldots,m\}} x_{jk} = 1 \qquad \forall j \in R, \tag{10}$$

$$\sum_{k=1} x_{jk} = 1 \qquad \forall j \in E. \tag{11}$$

Besides the optimization objectives in Eqs. (1) to (4), several constraints are also considered to ensure the feasibility of the generated solution. Constraint (5) enables different tasks to be assigned to the same workstation. Meanwhile, Constraint (6) limits the assigned task on the same workstation as different prescribed equipment. Constraints (7) and (8) are related to controls and ensure the maximum operational cycle time not be exceeded. Constraints (9), (10), and (11) involve each assigned task assigned to only one workstation, which is either left or right.

In this work, the weighted sum approach is used to deal with the multi-objective problem. Therefore, the optimization objectives considered in this work need to be normalized because they have different ranges. For this purpose, f_i is normalized to the range of [0, 1] as follows:

$$\widehat{f_1} = \frac{f_i - f_{i_{\min}}}{f_{i_{\max}} - f_{i_{\min}}}. \tag{12}$$

The minimum and maximum optimization objectives are defined as follows:

$$f_{1_{\min}} = 0, \tag{13}$$

$$f_{1_{\max}} = f_{2_{\min}}, \tag{14}$$

$$f_{2_{\min}} = \frac{\sum_{i=1}^{n} t_i}{ct_{\max}}, \tag{15}$$

$$f_{2_{\max}} = \frac{\sum_{i=1}^{n} t_i}{\max(t_i)}, \tag{16}$$

$$f_{3_{\min}} = 0. \tag{17}$$

The fitness function for this problem is presented as follows:

$$f_{3_{\max}} = f_{2_{\max}} \cdot ct_{\max} - \sum_{i=1}^{n} t_i, \tag{18}$$

$$f_{4_{\min}} = r_{type} - 1, \tag{19}$$

$$f_{4_{\max}} = \sum r, \tag{20}$$

$$f = w_1\widehat{f_1} + w_2\widehat{f_2} + w_3\widehat{f_3} + w_4\widehat{f_4}. \tag{21}$$

w_1, w_2, w_3, and w_4 were set to 0.25.

3. Modified Particle Swarm Optimisation (PSO)

PSO is a meta-heuristic searching method that is inspired by the swarming behavior of flocking birds. This mechanism is particularly based on the migrating birds' population and their flying directions. Every single migrating bird is considered a particle, which usually adjusts its searching or flying direction according to previous flying experience. Each particle represents a potential solution with a certain position (current solution), velocity (magnitude and direction towards the optimal solution), and fitness value (performance measure of the specific problem). Compared to other evolutionary approaches including ACO and

GA methods, PSO is respectively known to have faster convergence towards the optimal solution [32].

The PSO algorithm begins with the initialization procedure, where each particle represents the population in a D-dimensional vector as the possible constructed solution, $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$, and velocity, $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. Then, each solution is evaluated in terms of the objective function. Since the PSO is coded using a real number, a topological sort procedure is applied to match with the combinatorial problem in 2S-ALB. For example, in Figure 2, let X_1 = (4.81, 7.90, 2.12, 6.91, 6.63, 4.09, 0.27, 3.54, 3.95). The topological sort begins with identifying the candidate task without precedence. In Figure 2, tasks 1, 2, and 3 are the candidate tasks. In this situation, x_{11}, x_{12}, and x_{13} are compared to determine the selected task. Since x_{12} is the highest, task 2 is selected and stored in a feasible solution, F_1 = [2]. The selected task is then removed from the precedence graph. This approach is repeated until all the tasks from the graph are selected. For this example, the decoded feasible solution is F_1=[251483697].

Next, the particle best solution (Pbest) and global best (Gbest) solution are updated. Pbest refers to the current best solution for a particular particle, while the Gbest is the overall best solution. The Pbest and Gbest solutions are used to update the velocity and position of the solution. The following formula is used to update velocity (Eq. (22)) and position (Eq. (23)):

$$V_i^{t+1} = wV_i^t + c_1 r_1 \left(Pbest_i^t - X_i^t\right) + c_2 r_2 \left(Gbest^t - X_i^t\right), \quad (22)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1}. \quad (23)$$

In Eq. (22), t denotes the iteration number, while w is the inertia weight for regulating the previous effect of historical velocities. On the other hand, c_1 and c_2 are the acceleration coefficients, while r_1 and r_2 are random numbers between [0, 1]. The Pbest, Gbest, and particle position are updated until the specific iteration number is reached.

Previously, many studies have proposed different approaches to reducing premature convergence in PSO. Premature convergence in soft computing occurs because of the lack of diversity in the solution during the iteration process. In PSO, this phenomenon is directly related to velocity and position-updating procedures. The solution position is influenced by Pbest and Gbest with some randomness by r_1 and r_2. The Pbest, however, only affects a specific particle, compared with Gbest, which affects all the particles to move towards it. In case where Gbest is not updated (no better solution found) in a few consecutive iterations, there is a possibility for the majority of the particles to reach the Gbest. This situation reduces the solution diversity.

To overcome this problem, this work proposed considering the top three best solutions instead of only the single solution in Gbest. For this purpose, the single solution in Gbest is replaced with the average of the three best solutions.

$$Gbest^t = \left(g_1^t + g_2^t + g_3^t\right)/3. \quad (24)$$

In Eq. (24), g_1^t, g_2^t, and g_3^t represent the solution particles in the first, second, and third ranks, respectively, for the tth iteration. In the modified PSO, the Gbest is replaced with the new Gbest in Eq. (24). The reason to consider the top three solutions for Gbest is to improve the solution diversity. In the proposed mechanism, the particle position follows the average position from the three best solutions. Furthermore, the possibility that all the three solutions remain out updated is lower than that for the single Gbest solution in the original PSO. This mechanism makes the search direction more diverse and reduces the chance of getting trapped in local optima.

To prove this concept, a simple test using Rastrigin function is conducted. For this function, the optimum point is (0, 0). In this test, only six particles are used. The first particle is set as (0, 0), while the remaining five particles are randomly generated using the same pseudorandom number generator for both PSO and MPSO. The purpose of setting the first particle as the optimum point is to observe the particle movement over the iteration. For this purpose, the iteration is set only to 10. The particle positions for the first, fifth, and tenth iterations are captured. All other parameters for PSO and MPSO are the same.

Figure 3(a) and (b) present the particle movement for PSO and MPSO. In Figure 3(a), all particles move directly towards the Gbest (i.e., point (0, 0)) during the fifth iteration. At iteration 10, the particles only search for the solution around the Gbest within a limited range. Meanwhile, in MPSO, the particles are capable of maintaining diversity at the fifth and tenth iterations (Figure 3(b)). Although the searching range over the iteration becomes smaller, the particles in MPSO do not directly move towards the best solution. Therefore, it is expected that the MPSO will have a better exploration ability. The MPSO procedure is presented in Figure 4.

3.1. Coefficient tuning

MPSO algorithm consists of three coefficients that determine the algorithm performance. They are inertia (w), cognitive (c_1), and social (c_2) coefficients which are found in Eq. (22). The inertia coefficient determines how much the current velocity affects the position. Meanwhile, the cognitive and social coefficients control the exploration and exploitation of the candidate

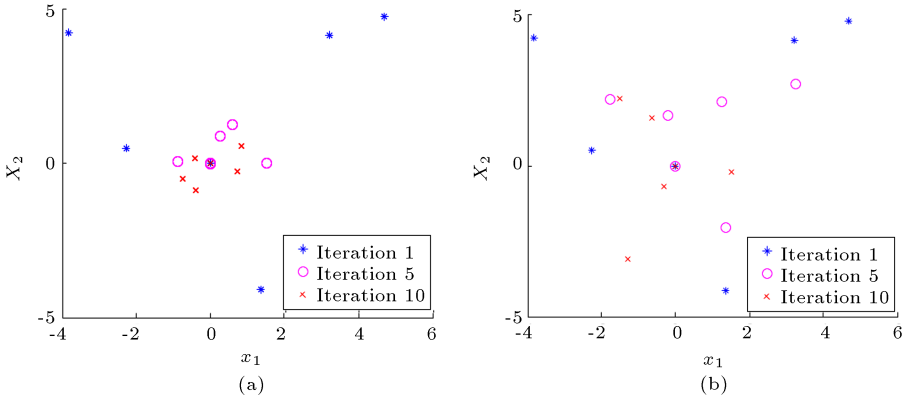

Figure 3. Particle movement for (a) PSO and (b) MPSO.

Initialize MPSO parameters: Population size (*npop*), coefficients (w, c_1, c_2), iteration counter ($iter = 0$), and maximum iteration ($iter_{max}$)

Initialize random velocity V_i and position X_i for $i = 1,2,\ldots, npop$

while $iter \leq iter_{max}$

$iter = iter + 1$

Decode X_i into a feasible assembly sequence, F_i

Evaluate the fitness function for the i^{th} solution, f_i

Update personal solution, $Pbest_i$

Update top three global solutions, g_1, g_2, and g_3

Update $Gbest^t = \left(g_1^t + g_2^t + g_3^t\right)/3$

Update velocity

$$V_i^{t+1} = wV_i^t + c_1 r_1 \left(Pbest_i^t - X_i^t\right) + c_2 r_2 \left(Gbest^t - X_i^t\right)$$

Update position

$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

End

Figure 4. Procedure of modified PSO

Table 3. Coefficient level for Taguchi design.

Coefficient	Low	Medium	High
w	0.8	1	1.2
c_1	1	1.4	1.8
c_2	1	1.4	1.8

solution in a search space, respectively. In order to identify the best coefficient value for MPSO to optimize 2S-ALB with resource constraint, an experiment using Taguchi design was conducted. For this experiment, the coefficients were set to three levels, as in Table 3. For this purpose, a Taguchi design with L9 orthogonal array was used.

To assess the coefficient performance, three sample problems were chosen among different problem size categories [18,33]. The selected problems were optimized using MPSO with different coefficient values. For each experiment setting, 20 repetitions were made and the fitness mean was calculated as output parameter. Based on the experiments conducted, the mean fitness for each experiment is presented in Table 4.

Taguchi analysis using the signal-to-noise ratio of "smaller is better" was used to analyze the output. Figure 5 shows the main effect plot for the signal-to-noise ratio. Based on the main effect plot, coefficient c_1 gives the highest effect, followed by c_2 and w. According to the figure, the MPSO performance was better when using lower inertia weight, w. Lower w allows the solution to be more diverse and open to changes. Meanwhile, for c_1 and c_2, the medium level was preferable in both coefficients. This indicates that the exploration and exploitation levels must be balanced to achieve a good-quality solution. Based on

Table 4. L9 Taguchi orthogonal array.

Experiment no.	w	c_1	c_2	Mean fitness
1	0.8	1	1	0.3729
2	0.8	1.4	1.4	0.2853
3	0.8	1.8	1.8	0.3351
4	1	1	1.4	0.3123
5	1	1.4	1.8	0.3249
6	1	1.8	1	0.4124
7	1.2	1	1.8	0.4692
8	1.2	1.4	1	0.3243
9	1.2	1.8	1.4	0.3255

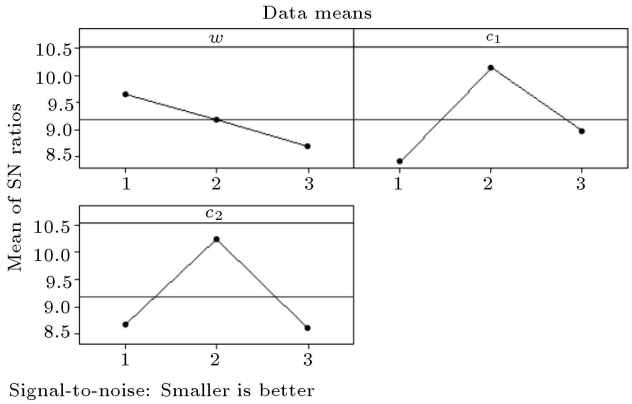

Figure 5. Main effect plot for signal-to-noise ratios.

the main effect plots, the optimum coefficients levels for MPSO are $w = 0.8$, $c_1 = 1.4$, and $c_2 = 1.4$.

4. Results and discussion

4.1. Computational experiment

A computational experiment is conducted to measure the performance of the Modified PSO (MPSO) to optimize 2S-ALB with resource constraints. For this purpose, 12 benchmark test problems are selected according to small, medium, and large sizes. The test problems are adopted from different sources [3,5,7,18,19,33,34]. Based on a range of problem sizes used in the literature, the small-sized problem is an assembly problem with less than 20 tasks. Meanwhile, the large-sized problem is a problem with more than 80 tasks. The assembly problem with a range of 20 to 80 tasks is considered to be medium in size. The detail of the test problems is presented in Table 5. Due to the lack of large-sized test problems, problems T83 and T111 are adopted from a simple ALB problem and the assembly directions (i.e., left, right, or either) are randomly generated. These benchmark problems, however, did not consider the resources required to conduct an assembly task. Therefore, the assembly

Table 5. Test problem category and sources.

Size	Problem	Number of tasks	Data source
Small	T4	4	[7]
	T9	9	[18]
	T12	12	[18]
	T16	16	[19]
Medium	T24	24	[18]
	T47	47	[34]
	T65	65	[19]
	T70	70	[3]
Large	T83	83	[33]
	T111	111	[33]
	T148	148	[5]
	T205	205	[19]

resources are also randomly generated for each of the assembly tasks.

MPSO is then compared with GA, ACO, and PSO in terms of performance. These algorithms are chosen because of their popularity in optimizing 2S-ALB problem. According to the earlier survey on the ALB problem, 70% of the problem was optimized using GA, ACO, and PSO algorithms [35]. A recent survey on 2S-ALB reveals that the GA and ACO were popular algorithms to optimize 2S-ALB according to the frequencies [13]. For computational purpose, the population size for all algorithms is 30 and the maximum iteration is 500. The optimization run is repeated for 20 times with different pseudorandom numbers for each of the cases.

The optimization results for the 2S-ALB with precedence constraints are presented in Table 6 to Table 8 based on the problem size. For the result of the small-sized problem in Table 6, all the algorithms are able to generate the same fitness and objective function value for T4 and T9 problems. On the other hand, for the T12 problem, MPSO exhibits the best fitness of all other algorithms. For this problem, MPSO is able to search for a solution with a smaller number of resources while maintaining other optimization objectives. In the T16 problem, all the algorithms are able to converge to the best solution, but ACO is of better performance in terms of consistency. For this problem, ACO is able to reach an optimum solution for every optimization run.

The results of the medium-sized problem in Table 7 indicate that the MPSO and ACO lead in terms of algorithm performance. The MPSO reaches minimum fitness and minimum average fitness in three

Table 6. Small-sized problem comparison.

Test problem	Algorithm	Minimum fitness	Maximum fitness	Average fitness	Standard deviation	f_1	f_2	f_3	f_4
T4	ACO	0.5505	0.5505	0.5505	0.0000	1	2	7	4
	GA	0.5505	0.5505	0.5505	0.0000	1	2	7	4
	PSO	0.5505	0.5505	0.5505	0.0000	1	2	7	4
	MPSO	0.5505	0.5505	0.5505	0.0000	1	2	7	4
T9	ACO	0.3094	0.3094	0.3094	0.0000	2	4	3	8
	GA	0.3094	0.3094	0.3094	0.0000	2	4	3	8
	PSO	0.3094	0.3094	0.3094	0.0000	2	4	3	8
	MPSO	0.3094	0.3094	0.3094	0.0000	2	4	3	8
T12	ACO	0.2531	**0.2531**	0.2531	**0.0000**	2	4	3	11
	GA	0.2531	**0.2531**	0.2531	**0.0000**	2	4	3	11
	PSO	0.2531	0.4380	0.3051	0.0733	2	4	3	11
	MPSO	**0.2455**	**0.2531**	**0.2470**	0.0031	2	4	3	10
T16	ACO	0.2151	**0.2151**	**0.2151**	**0.0000**	2	4	6	12
	GA	0.2151	0.4710	0.2506	0.0873	2	4	6	12
	PSO	0.2151	0.5076	0.4099	0.1065	2	4	6	12
	MPSO	0.2151	0.4068	0.2343	0.0590	2	4	6	12

Table 7. Medium-sized problem comparison.

Test problem	Algorithm	Minimum fitness	Maximum fitness	Average fitness	Standard deviation	f_1	f_2	f_3	f_4
T24	ACO	0.1899	**0.1930**	**0.1920**	**0.0011**	2	4	4	26
	GA	0.1899	0.1970	0.1927	0.0025	2	4	4	26
	PSO	0.1899	0.2144	0.2023	0.0096	2	4	4	26
	MPSO	**0.1899**	0.2073	0.1925	0.0053	2	4	4	26
T47	ACO	0.2697	0.3186	0.2989	0.0216	5	9	11702	88
	GA	0.3031	0.3270	0.3164	0.0079	5	10	13415	90
	PSO	0.2973	0.3231	0.3059	0.0077	5	10	10945	95
	MPSO	**0.1731**	**0.1776**	**0.1753**	**0.0020**	**4**	**8**	**2237**	73
T65	ACO	0.2586	0.2718	0.2674	**0.0041**	6	12	565	118
	GA	0.2612	0.3857	0.3332	0.0566	6	12	649	**113**
	PSO	0.2524	0.3822	0.2725	0.0388	6	12	469	**113**
	MPSO	**0.2491**	**0.2603**	**0.2569**	0.0052	**6**	**12**	**385**	116
T70	ACO	**0.4230**	0.4432	0.4339	0.0052	6	10	**273**	**97**
	GA	0.5492	0.6939	0.6205	0.0577	6	11	646	106
	PSO	0.4277	0.4556	0.4379	0.0096	6	10	303	99
	MPSO	0.4260	**0.4393**	**0.4316**	**0.0048**	6	10	293	98

Table 8. Large-sized problem comparison.

Test problem	Algorithm	Minimum fitness	Maximum fitness	Average fitness	Standard deviation	f_1	f_2	f_3	f_4
T83	ACO	0.4420	**0.4497**	0.4472	0.0032	6	11	39716	109
	GA	0.4886	0.4917	0.4906	**0.0014**	6	12	47593	118
	PSO	0.4329	0.4951	0.4598	0.0309	6	11	37109	107
	MPSO	**0.4324**	0.4524	**0.4400**	0.0079	**6**	**11**	**36944**	**107**
T111	ACO	0.3931	0.4206	0.4115	**0.0109**	7	13	67520	144
	GA	0.3212	0.4126	0.3737	0.0474	6	12	50709	136
	PSO	0.3177	0.4249	0.3952	0.0439	6	12	48681	135
	MPSO	**0.2989**	**0.3276**	**0.3200**	0.0120	**6**	**12**	**37365**	**135**
T148	ACO	0.2648	0.3682	**0.3070**	0.0553	5	10	565	119
	GA	0.2609	0.4228	0.3580	0.0865	5	10	515	118
	PSO	0.3623	0.4134	0.4003	**0.0214**	6	11	938	123
	MPSO	**0.2533**	**0.3608**	0.3092	0.0460	**5**	**10**	**405**	**122**
T205	ACO	0.2343	0.2399	0.2362	0.0023	5	10	4375	127
	GA	0.2363	0.3532	0.3236	0.0492	5	10	4575	126
	PSO	0.2343	0.3514	0.2990	0.0594	5	10	4375	127
	MPSO	**0.2308**	**0.2366**	**0.2348**	**0.0023**	**5**	**10**	**4055**	**126**

Table 9. Frequency of the rank for different algorithms.

	Algorithm	Rank 1	Rank 2	Rank 3	Rank 4	Average rank
Minimum fitness	ACO	5	3	3	1	2.00
	GA	4	2	1	5	2.58
	PSO	4	5	2	1	2.00
	MPSO	11	1	0	0	1.08
Average fitness	ACO	5	6	0	1	1.75
	GA	2	2	3	5	2.92
	PSO	2	0	6	4	3.00
	MPSO	9	3	0	0	1.25

cases. In the meantime, the ACO found minimum fitness in two cases while achieving minimum average in only one case. In T24, all the algorithms are able to search for minimum fitness; however, the ACO has better consistency. In T47 and T65 problems, the MPSO dominates the best minimum and average fitness among the algorithms. Meanwhile in T70, the ACO is able to search for better minimum fitness, while the proposed MPSO has better average fitness and standard deviation.

Table 8 presents the optimization result for the large-sized problem. For this class of the problem, MPSO is consistently able to search for better minimum fitness than the compared algorithms. In terms of average fitness, the MPSO has a better average in three cases, while the ACO has a better average rate in the remaining one case. The MPSO consistently finds minimum mated workstation, number of workstations, and idle time in all cases of the large-sized problem.

Next, a standard competition ranking method was used to analyze the results. In this approach, the algorithm with the best result was assigned Rank 1, while the worst one was ranked fourth. In case where the performance is tied, a similar rank is given and the next position is left empty. Table 9 presents the frequency of the rank for every algorithm in terms of minimum and average fitness.

Based on Table 9, the proposed MPSO is only ranked first and second for both minimum and average fitness, respectively. For minimum fitness, the MPSO

is able to search for the best solution in 91.6% of the problems. At the same time, the MPSO obtains better average fitness in 75% of the problems, while the remaining 25% is in the second place. The MPSO is characterized by better average rank for minimum and average fitness. In both categories, the MPSO scores 1.08 and 1.25 in average rank, respectively.

The nearest challenger to MPSO is the ACO algorithm. The ACO obtains the average rank 2.00 for minimum fitness while gets 1.75 for average fitness. Meanwhile, the PSO algorithm has the same average rank as ACO for minimum fitness, yet in the last position for average fitness. It is exhibited that the PSO converges to different angles in the search space for different optimization runs. For this reason, the PSO comes out with a different solution at different runs, which makes the fitness too diverse. For different angles, this behavior exhibits exclusive advantages because the algorithm explores different sides of the search space. However, it requires a high number of repetitions for the optimization run.

Figures 6 and 7 present the average rank by problem size for minimum and average fitness. In general, these figures show that for ACO, GA, and PSO, the performance of the algorithm becomes worse when the problem size increases. This trend is related to the size of the search space. When the problem size increases, the number of possible solutions excessively increases because of permutation combination. This makes the searching process harder, thus requiring an efficient algorithm. In contrast, the MPSO is able to maintain performance throughout different problem sizes.

Figures 8, 9, and 10 present the mean convergence for small-, medium-, and large-sized problems, respectively. For the small-sized problem, the MMFO convergence is almost stagnant at iteration 180. Meanwhile, in the medium-sized problem, the MMFO convergence is roughly stable at iteration 300. Even then, a few small improvements remain to take place until the end. For the large-sized problem, the convergence can still be observed to occur until the end of the run.

In the small-sized problem where the search space is relatively small, the MMFO algorithm manages to converge faster. This can be observed from the steep slope for the first 75 iterations in Figure 8. On the other hand, the early MMFO convergence in the medium-sized problem is intermixed between steep and short flat slopes. Meanwhile, the longer flat slope can be

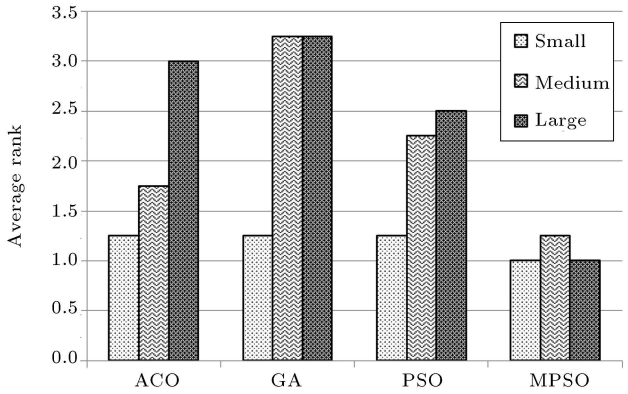

Figure 6. Minimum fitness by problem size.

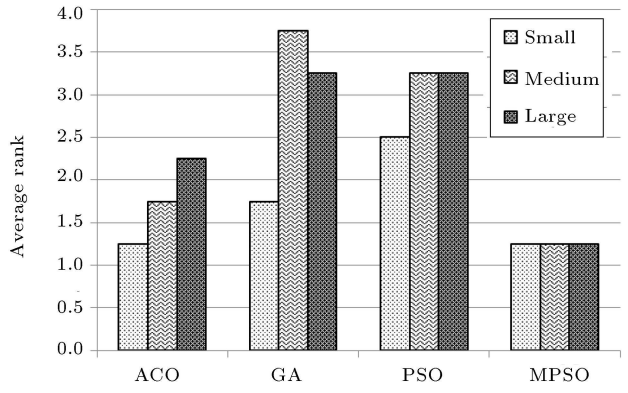

Figure 7. Average fitness by problem size.

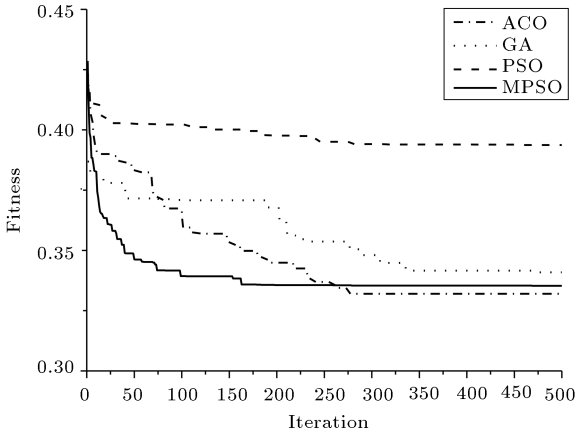

Figure 8. Convergence plot of small size problem.

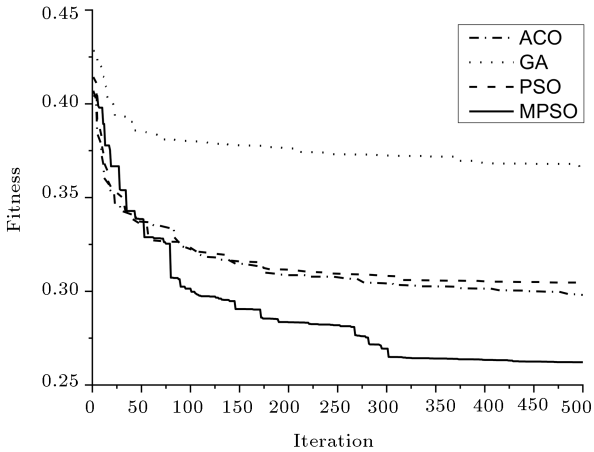

Figure 9. Convergence plot of medium size problem.

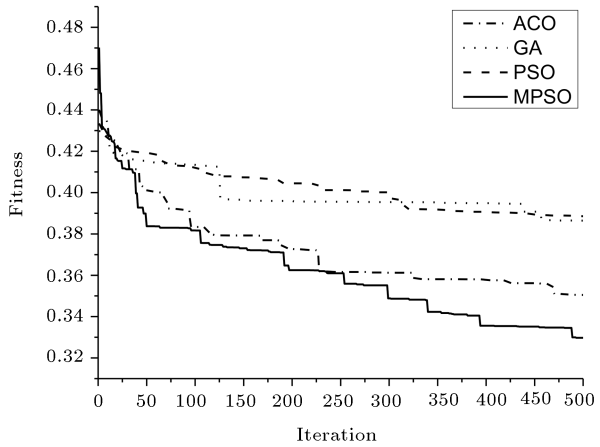

Figure 10. Convergence plot of large size problem.

observed in the case of the large-sized problem with periodical steep slopes. The patterns of convergence in small-, medium-, and large-sized problems are affected by the size of search space. When the problem size increases, the number of possible solutions increases. Furthermore, in the small-sized problems, tiny changes in the assembly sequence allocate a greater effect to the fitness value than the larger-sized problems because of the ratio between the changes and problem size.

4.2. Case study validation

A case study was conducted to validate the proposed model and algorithm to optimize 2S-ALB with resource constraints. The case study was conducted at an automotive assembler and focused on underbody assembly, which consisted of 34 assembly tasks. The assembly process on the studied line was conducted manually and it mainly involved the spot welding process. The existing assembly data are presented in Table 10. Currently, the production line is targeted to assemble 25 units of rear axle per day. Considering nine working hours per day, the desired cycle time should not exceed 22 minutes.

This problem has been modeled using the proposed 2S-ALB model and then, optimized using the MPSO algorithm. Since the company is expected to produce 25 units per day, the desired cycle time of 22 minutes is used for the optimization. Table 11 shows assignment of assembly tasks for existing and optimized layouts. Based on the existing layout, the actual cycle time is 25 minutes obtained at stations 2R and 5R. Meanwhile, for the optimized layout, the actual achieved cycle time is 21 minutes, which is found at stations 2L, 2R, and 3L. The optimized layout still utilizes 5-mated workstations and 10 workstations as in the existing layout, but it comes out with better cycle time, idle time, and total number of the resources used. According to the optimized layout, resource numbers and total idle time experienced 14.7% and 75% reduction rates, respectively.

Table 10. Assembly data for the underbody assembly.

Task	Precedence	Time (minute)	Side	Resource
1	–	9	Left	M1
2	1	3	Left	M2
3	–	5	Either	M1
4	2	7	Left	M3
5	–	8	Either	M1
6	–	6	Right	M2
7	5	3	Either	M1, M3
8	4	12	Left	M1, M2
9	3	4	Either	M1, M4
10	8	2	Left	M3
11	7	7	Either	M1
12	6	2	Right	M2
13	12	3	Right	M4
14	11	12	Either	M3
15	10	16	Left	M3
16	9	5	Either	M4
17	14	2	Either	M3
18	17	5	Right	M3, M4
19	13	2	Right	M4, M5
20	15, 16	2	Left	M5
21	20	2	Left	M6
22	20	3	Either	M7
23	21	7	Left	M5, M7
24	22	4	Either	M7, M8
25	18, 19	9	Either	M5
26	25	4	Right	M6, M10
27	23	6	Left	M7
28	24	4	Either	M8
29	27, 28	6	Left	M7
30	26	2	Either	M7, M8
31	30	11	Either	M8, M9
32	26	4	Right	M6, M7
33	32	5	Right	M9, M10
34	31	3	Either	M9, M11

Figure 11 shows the sensitivity of the obtained solution from MPSO optimization. In this test, the cycle time for 2S-ALB was simulated 5000 times through randomly varying assembly tasks from 5 to 10% using Gaussian distribution. The nonconformance percentage represents the cases that simulated cycle times exceeding the desired cycle time (i.e. 22 minutes). Based on the figure, to achieve nonconformance of less

Table 11. Assembly task assignment for existing and optimized layouts.

Layout	Station	Task	Time (minute)	Resource	Cycle time (minutes)	Total idle (minutes)
Existing layout	1L	1, 2, 3, 4	24	M1, M2, M3	25	64
	1R	5, 6, 7	17	M1, M2, M3		
	2L	8, 9, 10	18	M1, M2, M3, M4		
	2R	11, 12, 13, 14	25	M1, M2, M3, M4		
	3L	15, 16	21	M3, M4		
	3R	17, 18, 19	9	M3, M4, M5		
	4L	20, 21, 22, 23, 24	18	M5, M6, M7, M8		
	4R	25, 26	13	M5, M6, M10		
	5L	27, 28, 29	16	M7, M8		
	5R	30, 31, 32, 33, 34	25	M6, M7, M8, M9, M10, M11		
Optimized layout	1L	1, 2, 5	20	M1, M2	21	16
	1R	3, 6, 7, 12	20	M1, M2, M3		
	2L	4, 8, 10	21	M1, M2, M3		
	2R	11, 14, 17	21	M1, M3		
	3L	15, 20	21	M3, M5		
	3R	9, 13, 16, 18, 19	19	M1, M3, M4, M5		
	4L	21, 22, 24, 28, 30	17	M6, M7, M8		
	4R	25, 26, 32	17	M5, M6, M7, M10		
	5L	23, 27, 29	19	M5, M7		
	5R	31, 33, 34	19	M8, M9, M10, M11		

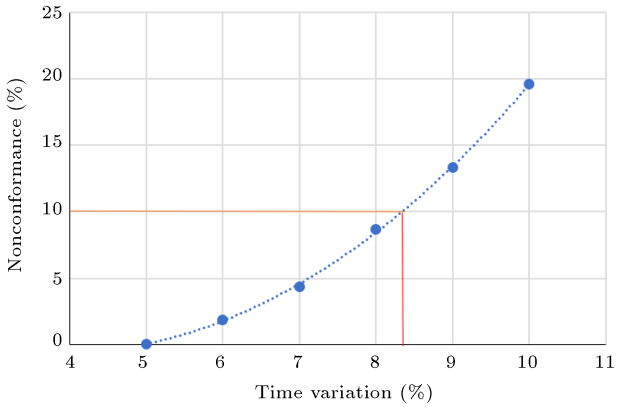

Figure 11. Sensitivity of optimised layout.

than 10%, the maximum assembly time variation is 8.34%.

The case study results indicate that the proposed 2S-ALB with the resource-constrained model can be applied to real-life problems. The result also proves that the proposed MPSO is capable of suggesting a better production layout with a shorter cycle time, idle time, and also total number of resources. In addition, the solution provided by MPSO enjoys good flexibility in terms of assembly time variation.

5. Conclusion and future work

This paper presented a Two-Sided Assembly Line Balancing (2S-ALB) with resource constraints. In contrast to the majority of existing works that assume all workstations have similar capabilities, this research considered the assembly resources including tools, machines, and workers to be minimized during the line balancing. For optimization purposes, Modified Particle Swarm Optimization (MPSO) was introduced by considering the top three solutions as the global best (Gbest) instead of one best solution in PSO algorithm. This change was made to maintain the solution diversity over the iterations.

A computational experiment was conducted by using 12 benchmark test problems in small, medium, and large sizes. The optimization results of MPSO were compared with those from popular algorithms for 2S-ALB, including Genetic Algorithm (GA), Ant Colony Optimization (ACO) and PSO algorithms. The computational experiment results indicate that the proposed MPSO has the capacity to search for the best solution in 11 out of 12 test problems. Unlike the compared algorithms, the MPSO is capable of maintaining performance even when the problem size

increases. Besides, the results also indicate that the proposed model for 2S-ALB with resource constraints can reduce the number of resources in an assembly line. It is essential that the assembly line be made in an efficient way. This result was proven through a case study where the optimized solution by MPSO could reduce the number of resources up to 14.7% compared to the existing layout. At the same time, the optimized case study problem also managed to reduce cycle time and idle time.

Modification to the Gbest has made the MPSO be more dynamic in terms of search direction. This change has two-fold advantages. The first advantage is that the proposed MPSO is of better exploration, which increases the chance to obtain an optimum solution. Meanwhile, the second advantage is that the possibility for the algorithm to get trapped in local optima could be reduced. This study, however, has a drawback in terms of multi-objective handling. Since this study applied the weighted sum approach to the multi-objective problem, the result highly depended on the weight used for each optimization objective. Currently, a similar weight was assigned to all optimization objectives. A future study should determine a suitable weight for different optimization objectives. Finally, the pareto optimality concept for multi-objective handling is suggested to have a better understanding of the optimum solution.

Acknowledgement

The authors would like to acknowledge the Ministry of Higher Education, Malaysia and Universiti Malaysia, Pahang for funding this research under FRGS grant RDU1901108 (FRGS/1/2019/TK03/UMP/02/3).

References

1. Alavidoost, M.H., Tarimoradi, M., and Zarandi, M.H.F. "Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems", *Applied Soft Computing*, **34**, pp. 655–677 (2015).
2. Saif, U., Guan, Z., Wang, B., et al. "Pareto lexicographic α-robust approach and its application in robust multi objective assembly line balancing problem", *Frontiers of Mechanical Engineering*, **9**(3), pp. 257–264 (2014).
3. Tuncel, G. and Aydin, D. "Two-sided assembly line balancing using teaching-learning based optimization algorithm", *Computers and Industrial Engineering*, **74**(1), pp. 291–299 (2014).
4. Saif, U., Guan, Z., Wang, B., et al. "A survey on assembly lines and its types", *Frontiers of Mechanical Engineering*, **9**(2), pp. 95–105 (2014).
5. Bartholdi, J.J. "Balancing two-sided assembly lines: A case study", *International Journal of Production Research*, **31**(10), pp. 2447–2461 (1993).
6. Purnomo, H.D., Wee, H., Rau, H., et al. "Two-sided assembly lines balancing with assignment restrictions", *Mathematical and Computer Modelling*, **57**(1–2), pp. 189–199 (2013).
7. Chutima, P. and Naruemitwong, W. "A Pareto biogeography-based optimisation for multi-objective two-sided assembly line sequencing problems with a learning effect", *Computers and Industrial Engineering*, **69**(1), pp. 89–104 (2014).
8. Khorasanian, D., Hejazi, S.R., and Moslehi, G. "Two-sided assembly line balancing considering the relationships between tasks", *Computers and Industrial Engineering*, **66**(4), pp. 1096–1105 (2013).
9. Duan, X., Wu, B., Hu, Y., et al. "An improved artificial bee colony algorithm with MaxTF heuristic rule for two-sided assembly line balancing problem", *Frontiers of Mechanical Engineering*, **14**, pp. 241–253 (2019).
10. Yuan, B., Zhang, C., Shao, X., et al. "An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines", *Computers & Operations Research*, **53**, pp. 32–41 (2015).
11. Chutima, P. and Chimklai, P. "Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge", *Computers and Industrial Engineering*, **62**(1), pp. 39–55 (2012).
12. Simaria, A.S. and Vilarinho, P.M. "2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines", *Computers & Industrial Engineering*, **56**(2), pp. 489–506 (2009).
13. Abdullah Make, M.R., Ab Rashid, M.F.F., and Razali, M.M. "A review of two-sided assembly line balancing problem", *The International Journal of Advanced Manufacturing Technology*, **89**(5–8), pp. 1743–1763 (2017).
14. Tapkan, P., Özbakir, L., and Baykasoğlu, A. "Bee algorithms for parallel two-sided assembly line balancing problem with walking times", *Applied Soft Computing Journal*, **39**, pp. 275–291 (2016).
15. Kucukkoc, I. and Zhang, D.Z. "Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters", *Computers and Industrial Engineering*, **84**, pp. 56–69 (2015).
16. Kucukkoc, I. and Zhang, D.Z. "A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem", *Production Planning and Control*, **26**(11), pp. 874–894 (2015).
17. Özcan, U., Gökcen, H., and Toklu, B. "Balancing parallel two-sided assembly lines", *International Journal of Production Research*, **48**(16), pp. 4767–4784 (2010).
18. Kim, Y.K., Kim, Y., and Kim, Y.J. "Two-sided assembly line balancing: A genetic algorithm approach", *Production Planning & Control*, **11**(1), pp. 44–53 (2000).

19. Lee, T.O., Kim, Y., and Kim, Y.K., "Two-sided assembly line balancing to maximize work relatedness and slackness", *Computers and Industrial Engineering*, **40**(3), pp. 273–292 (2001).
20. Delice, Y., Kızılkaya Aydoğan, E., and Özcan, U. "Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach", *International Journal of Production Research*, **54**(11), pp. 3429–3451 (2016).
21. Taha, R.B., El-Kharbotly, A.K., Sadek, Y.M., and Afia, N.H. "A genetic algorithm for solving two-sided assembly line balancing problems", *Ain Shams Engineering Journal*, **2**(3–4), pp. 227–240 (2011).
22. Baykasoglu, A. and Dereli, T. "Two-sided assembly line balancing using an ant-colony-based heuristic", *International Journal of Advanced Manufacturing Technology*, **36**(5–6), pp. 582–588 (2008).
23. Kucukkoc, I. and Zhang, D.Z. "Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach", *Computers and Industrial Engineering*, **97**, pp. 58–72 (2016).
24. Zhang, Z., Hu, J., and Cheng, W. "An ant colony algorithm for two-sided assembly line balancing problem type-II", *Advances in Intelligent Systems and Computing*, **213**, pp. 369–378 (2014).
25. Hu, X., Wu, E., and Jin, Y. "A station-oriented enumerative algorithm for two-sided assembly line balancing", *European Journal of Operational Research*, **186**(1), pp. 435–440 (2008).
26. Fattahi, P., Samouei, P., and Zandieh, M. "Simultaneous multi-skilled worker assignment and mixed-model two-sided assembly line balancing", *International Journal of Engineering*, **29**(2), pp. 211–221 (2016).
27. Chiang, W., Urban, T.L., and Luo, C. "Balancing stochastic two-sided assembly lines", *International Journal of Production Research*, **54**(20), pp. 6232–6250 (2016).
28. Delice, Y., Aydoğan, E.K., Özcan, U., et al. "Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm", *4OR*, **15**(1), pp. 37–66 (2017).
29. Li, Z., Janardhanan, M.N., Tang, Q., et al. "Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem", *Advances in Mechanical Engineering*, **8**(9), pp. 1–14 (2016).
30. Tang, Q., Li, Z., Zhang, L., et al. "A hybrid particle swarm optimization algorithm for large-sized two-sided assembly line balancing problem", *ICIC Express Letters*, **8**(7), pp. 1981–1986 (2014).
31. Make, M.R.A., Rashid, M.F.F., and Razali, M.M. "Modelling of two-sided assembly line balancing problem with resource constraints", in *IOP Conference Series: Materials Science and Engineering*, **160**(1), pp. 1–9 (2016).
32. Adnan, M.A. and Razzaque, M.A. "A comparative study of particle swarm optimization and cuckoo search techniques through problem-specific distance function", In *International Conference of Information and Communication Technology, ICoICT 2013*, pp. 88–92 (2013).
33. Scholl, A. *Benchmark Data Sets by Scholl*, Assembly Line Balancing Data Dets & Research Topics (1993). http://assembly-line-balancing.mansci.de/salbp/benchmark-data-sets-1993/.
34. Rubiano-Ovalle, O. and Arroyo-Almanza, A. "Solving a two-sided assembly line balancing problem using memetic algorithms", *Ingenieria y Universidad*, **13**(2), pp. 267–280 (2009).
35. Rashid, M.F.F., Hutabarat, W., and Tiwari, A. "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches", *The International Journal of Advanced Manufacturing Technology*, **59**(1–4), pp. 335–349 (2012).

Biographies

Muhammad Razif Abdullah Make is a part-time MSc graduate researcher at College of Engineering, Universiti Malaysia, Pahang. He is currently a Mechanical Engineer at a plantation company in Malaysia.

Mohd Fadzil Faisae Ab Rashid is an Associate Professor and Researcher at the Department of Industrial Engineering, College of Engineering, Universiti Malaysia, Pahang. His research interest is in manufacturing system optimization. He is also a Chartered Engineer under the Institution of Mechanical Engineers.