Optimisation of Two-sided Assembly Line Balancing with Resource Constraints Using Modified Particle Swarm Optimisation

Muhammad Razif ABDULLAH MAKE[1] and Mohd Fadzil Faisae AB RASHID[1,*]

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

*Corresponding Author's Email: ffaisae@ump.edu.my

Phone: +609-4246321   Fax: +609-4246222

Abstract

Two-sided Assembly Line Balancing (2S-ALB) is important in assembly plants that produce large-sized high-volume products, such as in automotive production. The 2S-ALB problem involves different assembly resources such as worker skills, tools, and machines required for the assembly. This research modelled and optimised the 2S-ALB with resource constraints. In the end, besides having good workload balance, the number of resources can also be optimised. For optimisation purpose, Particle Swarm Optimisation was modified to reduce the dependencies on a single best solution. This was conducted by replacing the best solution with top three solutions in the reproduction process. Computational experiment result using 12 benchmark test problems indicated that the 2S-ALB with resource constraints model was able to reduce the number of resources in an assembly line. Furthermore, the proposed modified Particle Swarm Optimisation (MPSO) was capable of searching for minimum solutions in 11 out of 12 test problems. The good performance of MPSO was attributed to its ability to maintain the particle diversity over the iteration. The proposed 2S-ALB model and MPSO algorithm were later validated using industrial case study. This research has a twofold contribution; novel 2S-ALB with resource constraints model and also modified PSO algorithm with enhanced performance.

Keywords: Manufacturing systems, Assembly line balancing, Two-sided line, resource constraints, Particle Swarm Optimisation

1. Introduction

Assembly line is a system that considers the arrangement of workstation, workers, tools or machines, which successively outlines the operations for being completed. It has been widely used in many manufacturing industries to cope with the increasing 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 have been reported in regard to the assembly line. Commencing from that 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 not only refers to the assigning of a task to a respective workstation but also towards enhancing production rates with the desired performance level [2]. Nowadays, ALB has become more important to cope 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 for solving the line balancing problem [3]. The great progress developed from time to time has extended the 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 work on 2S-ALB has inspired other researchers to study and extend this 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 one-sided line, the assembly process in 2S-ALB could be 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 are adopted as variations of the 2S-ALB problem. The 2S-ALB variation started 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 is well addressed in several research studies [3], [6]–[9].

Besides studying the general 2S-ALB, researchers also combined the 2S-ALB with the mixed-model assembly line balancing (MALB). The MALB is particularly considered to level the workload in every workstation on the line, besides levelling the part usage. It literally functions to achieve a balanced workload at specific processing times for each assembly task, while attempting to minimise the variation used by the different parts over time. This combination of 2S-ALB with MALB has broadly introduced the implementation of different optimisations and line balancing solution approaches [10]–[12]. Another combination with the 2S-ALB is the parallel assembly line balancing (P-ALB). The P-ALB is the combination of two or more lines placed parallel to each other, which becomes an idea of sharing tools and fixtures to complete the entire job. The two-sided P-ALB, which is the combination of 2S-ALB and P-ALB, is to shorten the assembly line while steadily running during a breakdown [13]–[16]. This combined problem was discussed by Ozcan, Gokcen, and Toklu (2010) [17] to provide much more benefits: (i) it can help produce similar products or different models of the same production of the adjacent lines, (ii) it can reduce the idle time and increase the efficiency of the assembly lines, (iii) it is able to complete production with a different cycle time for each of the lines, (iv) it can improve visibility and communication skills between operators, and (v) it is also able to reduce operator requirements.

Many studies have been conducted to work out the best optimum seeking approach, implementing either heuristic or meta-heuristic method for 2S-ALB. In an early study, Kim et al. in 2000 used Genetic Algorithm (GA) as the optimisation algorithm [18]. Then in 2001, it has been continued by Lee et al. employing the group assignment procedure [19]. The GA approach was also implemented by Yılmaz Delice, Kızılkaya Aydoğan, & Özcan (2016), Kucukkoc & Zhang (2015a), and Taha, El-Kharbotly, Sadek, & Afia (2011) to optimise 2S-ALB. Meanwhile, Baykasoglu and Dereli adopted Ant Colony Optimisation (ACO) to optimise the 2S-ALB [22]. They have successfully applied the ACO algorithm for a domestic product, which influences other researchers to deal with other sectors apart from the large-sized automotive products. In addition, many other researchers also implemented the ACO because of its good performance especially in combinatorial problems [15], [23], [24]. From earlier reviews, GA and ACO algorithms have successfully dominated other optimisation methods in terms of performance and also frequencies that make these algorithms more popular [13]. Besides that, different algorithms were also implemented through several reported studies. For instance, Hu et al. (2008) have been reported to implement the enumerative algorithm combined with the Hoffmann heuristic method [25].

In the meantime, Particle Swarm Optimisation (PSO) algorithm was also frequently implemented for 2S-ALB. The PSO assisted with Taguchi has been implemented for 2S-ALB with multi-skilled worker assignment [26]. Researchers also implemented ACO algorithm to optimise stochastic 2S-ALB, instead of deterministic time in the majority of 2S-ALB works [27]. Meanwhile in 2012, Chutima and Chimklai proposed a PSO with Negative Knowledge (PSONK) for the optimisation of complex combination with the 2S-ALB problem [11]. Y Delice, Kızılkaya Aydoğan, Özcan, and İlkay (2017) also later implemented the PSONK but proposed a combined selection mechanism for the assembly task. Besides that, different approaches have been proposed to improve the PSO performances [29]–[31]. Although the advantages of PSO algorithm have been well reported, its application and improvement are still needed. Generally, PSO is known as a fast optimiser with a robust algorithm, which provides a high-quality solution. However, the high focus towards a single best solution in PSO can lead to a premature convergence or local optima. This phenomenon is described where the convergence is stopped earlier and is considered to express the solution as the best. This problem occurs when the algorithm tries to figure out the path of searching direction while still following the best earlier solution. The limitation in providing the best solution may occur due to fewer parameter settings, as the PSO algorithm only requires a simple specification as a setting before generating a solution.

Despite the many studies on 2S-ALB conducted, the majority of these works assumed that the assembly workstation has 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. The proper utilisation of resources depending on their skills and precedence has integrated the assembly line to be fully optimised. Besides, with the appropriate use of the machine, it is also able to solve the inadequate space problem for the assembly line in allocating the required machines on the workstation [6].

In order to overcome the limitation, this paper considered the resources required to conduct a specific assembly task. By considering the assembly resource constraints, the number of resources could be optimised. For optimisation 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 algorithm exploration ability. Section 2 of this paper presents the 2S-ALB with resource constraints problem. Section 3 presents the proposed Modified Particle Swarm Optimisation (MPSO) algorithm. The computational experiment is set up and the results are discussed in Section 4. Finally, Section 5 summarises and concludes the research work.

2. 2S-ALB with Resource Constraints Problem

The 2S-ALB is a modified structure that is essentially created from the one-sided ALB problem. The main goal of this problem is to enhance the production rate and increase the line efficiency. Flexibility to produce a high volume of large-sized product in 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 in a two-sided assembly line, a pair of lines placed opposite to each other will be 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 conveyor.

A comprehensive study in making the idea of balancing 2S-ALB problem has been presented [32]. Commencing from a particular task relation called 'precedence relation graph' that is built with circle and arrows, the example of precedence relation graph with nine tasks is depicted in **Figure 2**. Each circle represents the assigned task, while the arrows linked represent each relation between the task. The associated data of each processing time and operational direction are also specified on top of each circle (assign task). Three types of operational direction will be considered: left (L), right (R) and either (E). For left and right side, the execution is outright and should be actualised for the following position. Meanwhile, for either side direction, the task could be executed on any side of the workstation, either on the left side or the right side.

Then, an assembly data is presented in 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*' as the present task and '*j*' as the next assigned task. 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 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, value '1' is for the left side operation, value '2' is for either side operation, while value '3' is for the right-side operation. The resource details are also coded in numbers 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 the case where the number of resources is larger, the matrix can be expanded to fit all the data.

2.1 Problem Assumptions and Notations

The general assumptions of the problem are as follows:

- Task times and resources used (machine, tools, worker) are known and deterministic.
- Tasks have preferences regarding the operational direction (side), i.e., left side, either side or right side.
- Every task can be operated only after all its immediate predecessors are completed.
- The maximum operational cycle time is fixed and could not be exceeded.
- Every task cannot be split between workstations and must be assigned to exactly one workstation.
- The tasks with positive zoning must be operated in the same workstation.
- The 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 breakdowns of machines and tools are not considered, and the assembly process is constantly performed.

The notations used in this mathematical formulation are summarised as follows.

J	: number of mated-workstation $j = 1, 2, ..., J$
I	: number of one-sided workstation $i = 1, 2, ..., I$
F	: 1, if there is any space available on the operating time, otherwise, 0
N	: number of resource utilisation $n = 1, 2, ..., N$
X_{ms}	: 1, if mated-workstation *j* is utilised for both side of the line, otherwise, 0
Y_s	: 1, if mated-workstation *j* is utilised 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 utilised 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 optimisation objectives are considered. The first optimisation objective as in Equation (1) is to minimise the mated workstation, *f₁*. The second optimisation objective in Equation (2) is to minimise the number of the workstation, *f₂*. A mated workstation consists of a pair of left and right workstation on the assembly line. Meanwhile, the number of workstations calculates the total individual workstation. The third optimisation objective is to minimise idle time, *f₃* as presented in Equation (3). Finally, the fourth optimisation objective to minimise the number of resources, *f₄* presented in Equation (4). By using the number of resources as one of the optimisation objectives, the number of resources can be minimised. This can be achieved by assigning the assembly task that uses a similar resource in one workstation.

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

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

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

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

$$\sum_{K} k\left(x_{ik_1} + x_{ik_2}\right) - \sum_{K} k\left(x_{jk_1} + x_{jk_2}\right) = 0 \qquad (i, j) \in ZP_{ij} \tag{5}$$

$$\sum_{K} k\left(x_{ik_1} + x_{ik_2}\right) - \sum_{K} k\left(x_{jk_1} + x_{jk_2}\right) \neq 0 \qquad (i, j) \in ZN_{ij} \tag{6}$$

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

$$s_{jk} = \sum_{u=1}^{U} x_{ujk}\left(t_{u+1}^{s} - t_u^{f}\right) + \left(CT - t_u^{f}\right) \qquad u \in Q_{jk} \tag{8}$$

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

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

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

Besides the optimisation objectives in Equation (1) to (4), several constraints are also being considered to ensure the feasibility of 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) are engaged to each assigned task to only one workstation which is either left or right.

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

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

The minimum and maximum optimisation objectives are defined as follows:

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

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

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

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

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

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

$$f_{4_{\min}} = r_{type} - 1 \qquad (19)$$

$$f_{4_{\max}} = \sum r \qquad (20)$$

The fitness function for this problem is presented as follows: The w_1, w_2, w_3 and w_4 were set at 0.25.

$$f = w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4 \qquad (21)$$

3. Modified Particle Swarm Optimisation

PSO is a meta-heuristic searching method that is inspired from the swarming behaviour 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 the 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 such as ACO and GA methods, PSO is respectively known to have a faster convergence towards the optimal solution [33].

The PSO algorithm begins with the initialisation procedure, where each particle represents the population in a *D*-dimensional vector as the constructed possible solution, $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$ and velocity, $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$. Then, each solution is evaluated according to 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 the example in **Figure 2**, let the 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**, Task 1, 2 and 3 are the candidate tasks. In this situation, the 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 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 = [2 5 1 4 8 3 6 9 7].

Next, the particle best solution (Pbest) and global best (Gbest) 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 (22) and position (23):

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

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{23}$$

In Equation (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, a lot of studies 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 the Pbest and Gbest with some randomness by r_1 and r_2. The Pbest, however, only influences a specific particle, compared with Gbest that affects all the particles to move towards it. In the 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 will reduce the solution diversity.

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

$$Gbest^t = \left(g_1^t + g_2^t + g_3^t \right) / 3 \tag{24}$$

In Equation (24), g_1^t, g_2^t, g_3^t refer to the solution particle 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 Equation (24). The reason to consider the top three solutions for Gbest is to improve the solution diversity. In the proposed mechanism, the particle position will follow the average position from the three best solutions. Furthermore, the possibility for all three solutions not being updated is small compared with 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 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 position for the first, fifth and tenth iterations are captured. All other parameters for PSO and MPSO are the same.

Figure 3(a) and **3(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. During iteration 10, the particles only search the solution around the Gbest within a limited range. Meanwhile in MPSO, the particles are capable of maintaining the diversity in 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 better exploration ability. The procedure of MPSO is presented as follows:

Procedure of Modified PSO

```
Initialise MPSO parameters: Population size ($npop$), coefficients ($w$, $c_1$, $c_2$), iteration counter
    ($iter = 0$) and maximum iteration ($iter_{max}$)
Initialise random velocity, $V_i$ and position, $X_i$ for $i = 1,2,\ldots, npop$
While $iter \leq iter_{max}$
    $iter = iter + 1$
    Decode the $X_i$ into feasible assembly sequence, $F_i$
    Evaluate the fitness function for $i$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
        $V_i^{t+1} = wV_i^t + c_1r_1\left(Pbest_i^t - X_i^t\right) + c_2r_2\left(Gbest^t - X_i^t\right)$
    Update position
        $X_i^{t+1} = X_i^t + V_i^{t+1}$
End
```

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 that found in Equation (22). The inertia coefficient determines how much the current velocity influence the position. Meanwhile, the cognitive and social coefficients control the exploration and exploitation of the candidate solution in search space, respectively. In order to identify the best coefficient value for MPSO to optimise 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 from different problem size category [18], [34]. The selected problems were optimised using MPSO with different coefficient values. For each experiment setting, 20 repetitions were made and the mean of fitness were calculated as output parameter. Based on experiment conducted, the mean fitness for each experiment is presented in **Table 4.**

Taguchi analysis using "smaller is better" Signal-to-Noise ratio was used to analyse the output. **Figure 4** shows the main effect plot signal-to-noise ratio. Based on the main effect plot, c_1 coefficient gives the highest effect, then followed by c_2 and *w*. According to the figure, the MPSO performance was better when using lower inertia weight, w. A 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 indicated that the exploration and exploitation level must be balanced to achieve a good quality solution. Based on main effect plots, the optimum coefficients level 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 optimise 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], [34], [35]. Based on the range of problem size used in the literature, the small-sized problem is an assembly problem with less than 20 tasks. Meanwhile, the large-sized problem is the problem with more than 80 tasks. The assembly problem in between 20 to 80 tasks is considered as medium size. The detail of the test problems is presented in **Table 5**. Due to the lack of large-sized test problems, problem 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 resources are also randomly generated for each of the assembly tasks.

The performance of MPSO is then compared against GA, ACO and PSO. These algorithms are chosen because of their popularity in optimising 2S-ALB problem. According to the earlier survey on the ALB problem, 70% of the problem was optimised using GA, ACO and PSO algorithms [36]. The recent survey on 2S-ALB also reveals that the GA and ACO were the popular algorithms to optimise 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 optimisation run is repeated for 20 times with different pseudorandom for each of the cases.

The optimisation results for the 2S-ALB with precedence constraints are presented in **Table 6** until **Table 8** based on the problem size. For the result of small-sized problem in **Table 6**, all 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 shows the best fitness compared with other algorithms. For this problem, MPSO is able to search for a solution with a smaller number of resources while maintaining other optimisation objectives. In the T16 problem, all algorithms are able to converge to the best solution, but ACO has better performance in terms of consistency. For this problem, ACO is able to reach the optimum solution for every optimisation run.

The results of 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 cases. In the meantime, the ACO found the minimum fitness in two cases, while the minimum average in only one case. In T24, all algorithms are able to search for minimum fitness, but again the ACO has better consistency. In T47 and T65 problems, the MPSO dominates the best minimum and average fitness compared with other algorithms. Meanwhile in T70, the ACO is able to search for better minimum fitness, but the proposed MPSO has better average fitness and standard deviation.

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

Next, a standard competition ranking method was used to analyse the results. In this approach, the algorithm with the best result was assigned as Rank 1, while the worst being Rank 4. In the case where the performance is tied, a similar rank will be given, and the next position will be 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 in Rank 1 and Rank 2 for both minimum and average fitness. 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 also has a better average rank for minimum and average fitness. In both categories, the MPSO obtained 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 1.75 for average fitness. Meanwhile, the PSO algorithm also has the same average rank as ACO for minimum fitness, but in the last position for average fitness. It shows that the PSO converges to the different angles in the search space for the different optimisation runs. For this reason, the PSO comes out with a different solution for different runs that makes the fitness too diverse. For different angles, this behaviour has its own advantage because the algorithm will explore different sides of the search space. However, it requires a high number of repetitions for the optimisation run.

Figure 5 and **Figure 6** 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 the performance throughout the different problem sizes.

Figures 7, **8** and **9** 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 still occur 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 also relatively small, the MMFO algorithm manages to converge faster. This can be observed from the steep slope for the first 75 iterations in **Figure 7**. On the other hand, the early MMFO convergence in medium-sized problem is intermixed between steep and short flat slopes. Meanwhile, the longer flat slope can be observed in 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 also increases. Furthermore, in the small-sized problems, tiny changes in the assembly sequence give more effect on the fitness value compared with the larger-sized problems because of the ratio between the changes and problem size.

4.2 Case Study Validation

A case study has been conducted to validate the proposed model and algorithm to optimise 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 in the studied line was conducted manually and mainly involved spot welding process. The existing assembly data is 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 modelled using the proposed 2S-ALB model and then optimised 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 optimisation. **Table 11** shows assembly tasks assignment for existing and optimised layout. Based on the existing layout, the actual cycle time is 25 minutes, obtained at stations 2R and 5R. Meanwhile for the optimised layout, the actual cycle time achieved is 21 minutes, which is found at stations 2L, 2R and 3L. The optimised layout still utilised 5-mated workstations and 10 workstations as in the existing layout, but came out with better cycle time, idle time and total number of resources used. According to the optimised layout, there were 14.7 % and 75 % reduction of resource numbers and total idle time, respectively.

Figure 10 shows the sensitivity of the obtained solution from MPSO optimisation. In this test, cycle time for 2S-ALB was simulated 5000 times by randomly varying assembly tasks time between 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 than 10 %, the maximum assembly time variation is 8.34 %.

The case study results indicated that the proposed 2S-ALB with resource constraints model can be implemented for real-life problems. The result also proved that the proposed MPSO is capable of suggesting better production layout with less cycle time, idle time and also total number of resources. In addition, the solution provided by MPSO has good flexibility in terms of assembly time variation.

5. Conclusion & Future Work

This paper presented a 2S-ALB with resource constraints. In contrast to the majority of existing works that assume all workstations have similar capabilities, this research considers the assembly resources including tools, machines, and workers to be minimised during the line balancing. For optimisation purposes, 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 from small, medium and large sizes. The optimisation results of MPSO were compared with results from popular algorithms for 2S-ALB, including GA, ACO and PSO algorithms. The computational experiment results indicated that the proposed MPSO has the capacity to search for the best solution in 11 out of 12 test problems. Unlike the comparison algorithms, the MPSO is capable of maintaining performance even when the problem size increases. Besides that, the results also indicated that the proposed model for 2S-ALB with resource constraints can reduce the number of resources in an assembly line. This is important to set up the assembly line in an efficient way. This result has been proven via case study, where the optimised solution by MPSO is the able to reduce number of resources up to 14.7 % compared with the existing layout. At the same time, the optimised case study problem also managed to reduce cycle time and idle time.

The modification on the Gbest has made the MPSO become more dynamic in terms of search direction. This change has two-fold advantages. The first advantage is that the proposed MPSO has better exploration, which increases the chances 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 work however, has a drawback in terms of multi-objective handling. Since this work implemented the weighted sum approach for the multi-objective problem, the result highly depends on the weight used for each optimisation objective. Currently, a similar weight is assigned to all optimisation objectives. In the future, a study to determine a suitable weight for different optimisation objectives should be proposed. Finally, the Pareto optimality concept for multi-objective handling is suggested to have a better view on 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*, vol. 34, pp. 655–677, (2015).
2. Saif, U., Guan, Z., Wang, B., and Mirza, J., “Pareto lexicographic α-robust approach and its application in robust multi objective assembly line balancing problem,” *Frontiers of Mechanical Engineering*, vol. 9, no. 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*, vol. 74, no. 1, pp. 291–299, (2014).

4. Saif, U., Guan, Z., Wang, B., Mirza, J., and Huang, S., "A survey on assembly lines and its types," *Frontiers of Mechanical Engineering*, vol. 9, no. 2, pp. 95–105, (2014).

5. Bartholdi, J. J., "Balancing two-sided assembly lines: A case study," *International Journal of Production Research*, vol. 31, no. 10, pp. 2447–2461, (1993).

6. Purnomo, H. D., Wee, H., Rau, H., Dwi, H., Wee, H., and Rau, H., "Two-sided assembly lines balancing with assignment restrictions," *Mathematical and Computer Modelling*, vol. 57, no. 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*, vol. 69, no. 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*, vol. 66, no. 4, pp. 1096–1105, (2013).

9. Duan, X., Wu, B., Hu, Y., Liu, J., and Xiong, J., "An improved artificial bee colony algorithm with MaxTF heuristic rule for two-sided assembly line balancing problem," *Frontiers of Mechanical Engineering*, vol. In Press, (2018).

10. Yuan, B., Zhang, C., Shao, X., and Jiang, Z., "An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines," *Computers & Operations Research*, vol. 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*, vol. 62, no. 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*, vol. 56, no. 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*, vol. 89, no. 5–8, pp. 1743–1763, (2017).

14. Tapkan, P., Özbakir, L., and Baykasoılu, A., "Bee algorithms for parallel two-sided assembly line balancing problem with walking times," *Applied Soft Computing Journal*, vol. 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*, vol. 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*, vol. 26, no. 11, pp. 874–894, (2015).

17. Özcan, U., Gökcen, H., and Toklu, B., "Balancing parallel two-sided assembly lines," *International Journal of Production Research*, vol. 48, no. 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*, vol. 11, no. 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*, vol. 40, no. 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*, vol. 54, no. 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*, vol. 2, no. 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*, vol. 36, no. 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*, vol. 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*, vol. 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*, vol. 186, no. 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*, vol.

29, no. 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*, vol. 54, no. 20, pp. 6232–6250, (2016).

28. Delice, Y., Kızılkaya Aydoğan, E., Özcan, U., and İlkay, M. S., "A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing," *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 23–36, (2017).

29. Delice, Y., Aydoğan, E. K., Özcan, U., and İlkay, M. S., "Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm," *4OR*, vol. 15, no. 1, pp. 37–66, (2017).

30. Li, Z., Janardhanan, M. N., Tang, Q., and Nielsen, P., "Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem," *Advances in Mechanical Engineering*, vol. 8, no. 9, pp. 1–14, (2016).

31. Tang, Q., Li, Z., Zhang, L., and Floudas, C. A., "A hybrid particle swarm optimization algorithm for large-sized two-sided assembly line balancing problem," *ICIC Express Letters*, vol. 8, no. 7, pp. 1981–1986, (2014).

32. 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*, 2016, vol. 160, no. 1.

33. Adnan, M. A. and Razzaque, M. A., "A comparative study of Particle Swarm Optimization and Cuckoo Search techniques through problem-specific distance function," in *2013 International Conference of Information and Communication Technology, ICoICT 2013*, 2013, pp. 88–92.

34. 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/.

35. Rubiano-Ovalle, O. and Arroyo-Almanza, A., "Solving a two-sided assembly line balancing problem using memetic algorithms," *Ingenieria y Universidad*, vol. 13, no. 2, pp. 267–280, (2009).

36. 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*, vol. 59, no. 1–4, pp. 335–349, (2012).

LIST OF FIGURES CAPTION

LIST OF TABLES CAPTION

LIST OF FIGURES

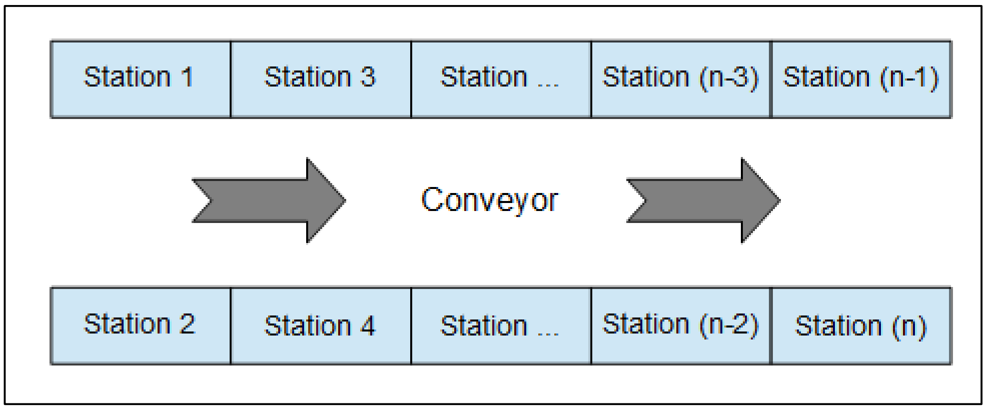

Figure 1. Two-sided assembly line

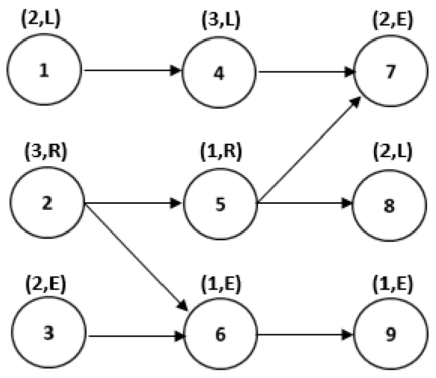

Figure 2. Precedence relation graph

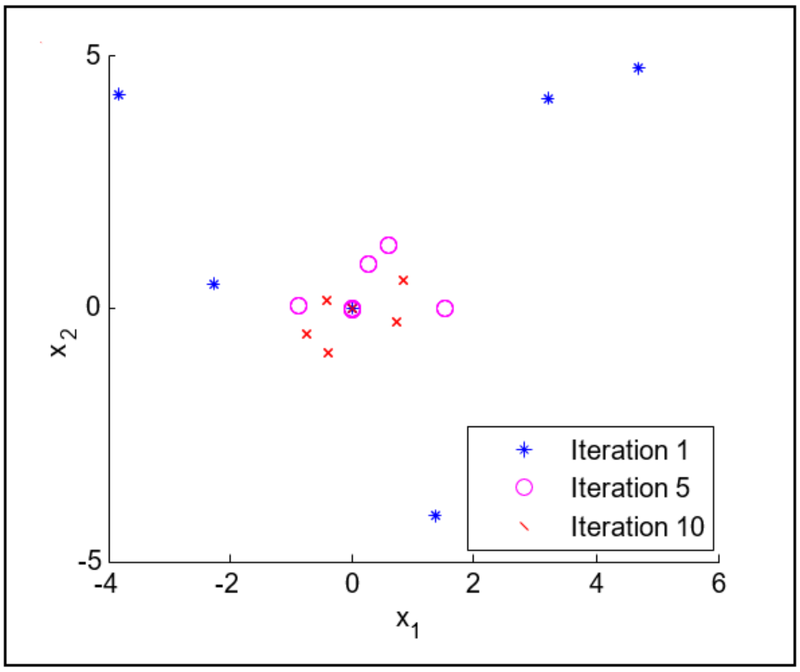

Figure 3(a). Particle movement for PSO

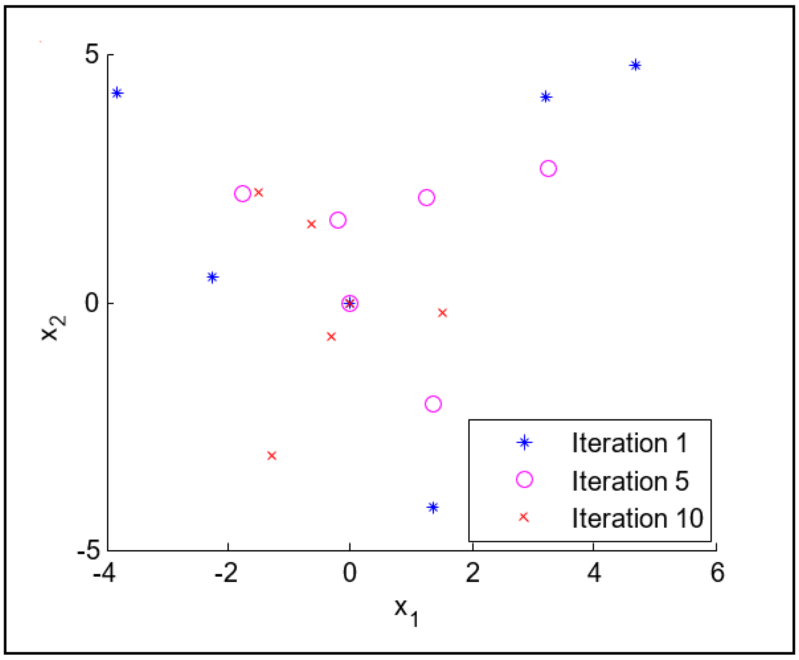

Figure 3(b). Particle movement for MPSO

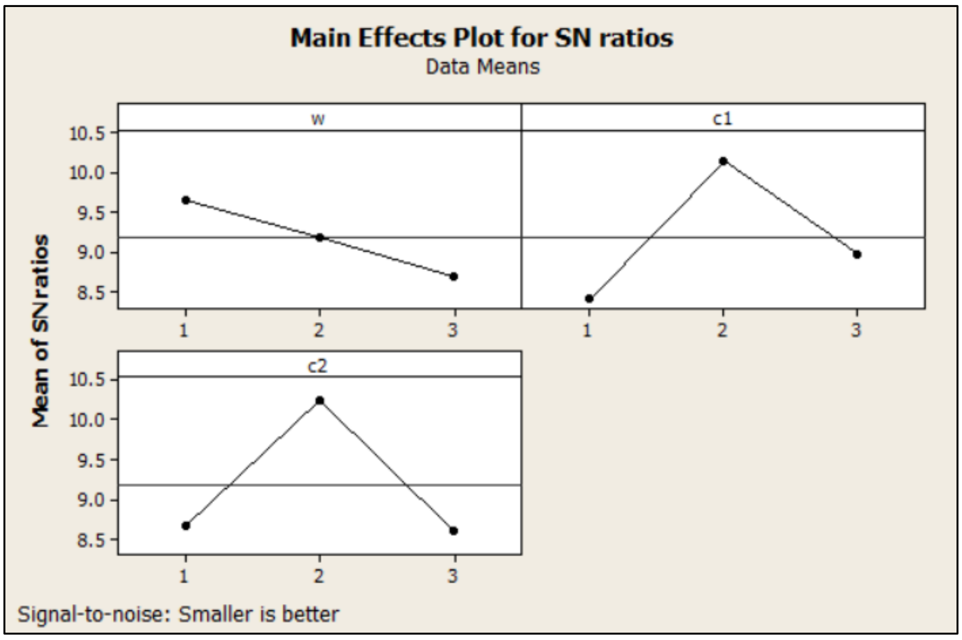

Figure 4. Main effect plot for Signal-to-Noise ratios

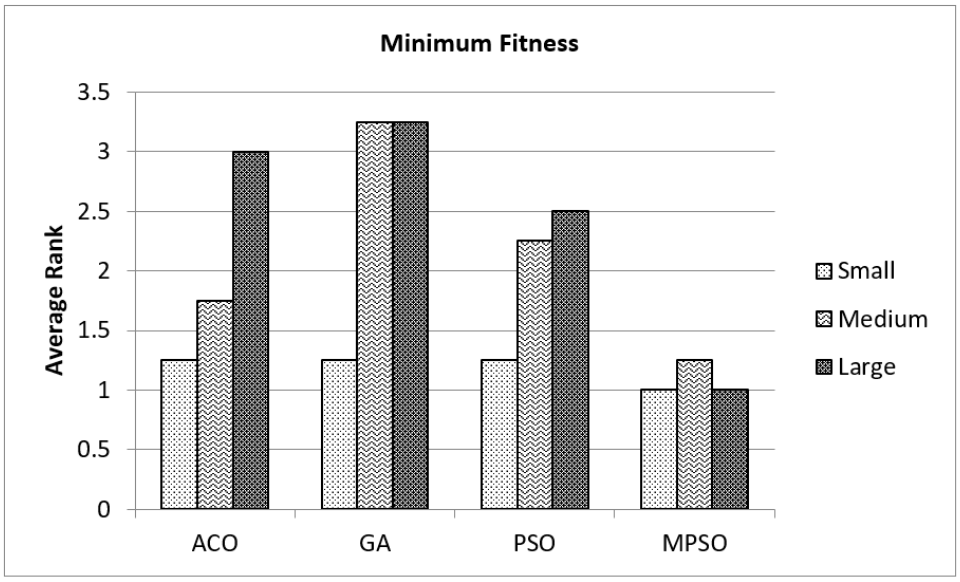

Figure 5. Minimum fitness by problem size

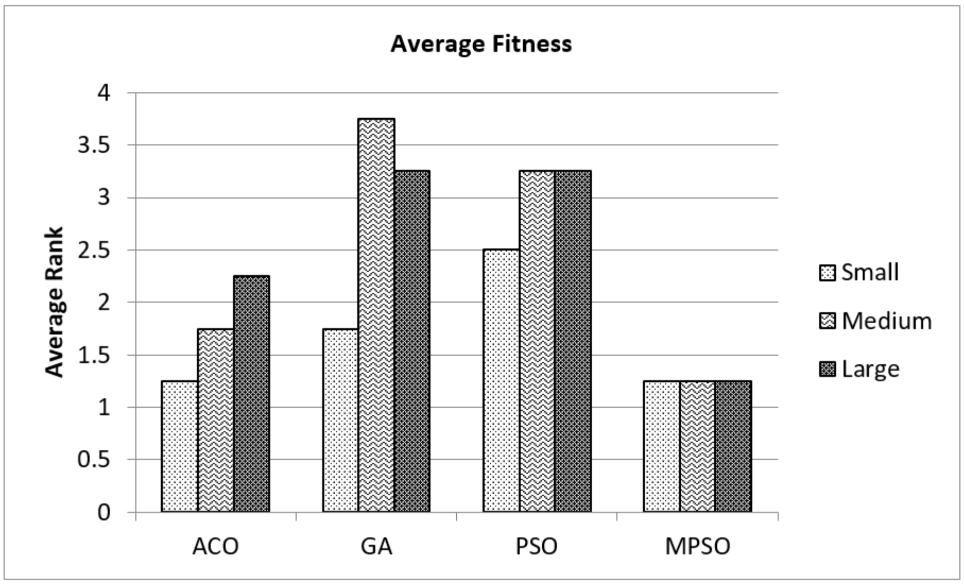

Figure 6. Average fitness by problem size

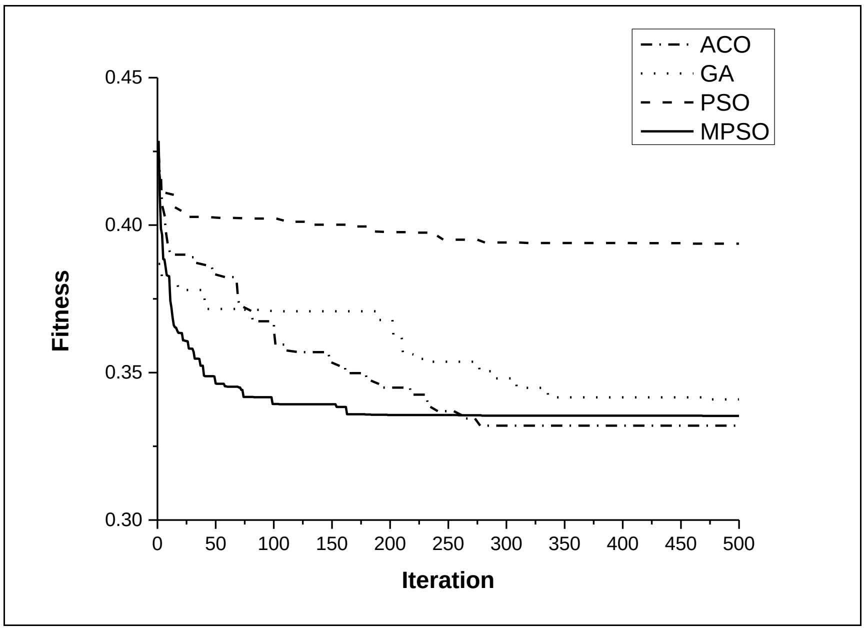

Figure 7. Convergence plot of small size problem

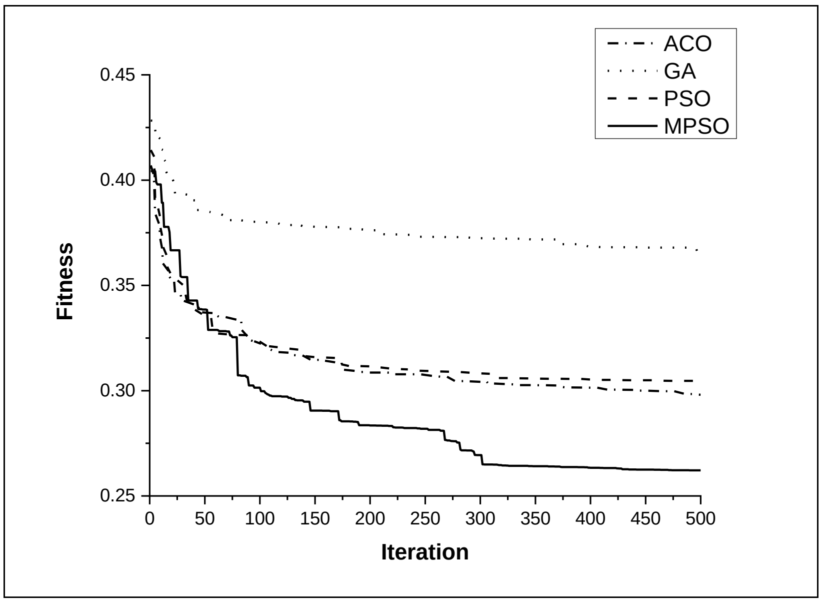

Figure 8. Convergence plot of medium size problem

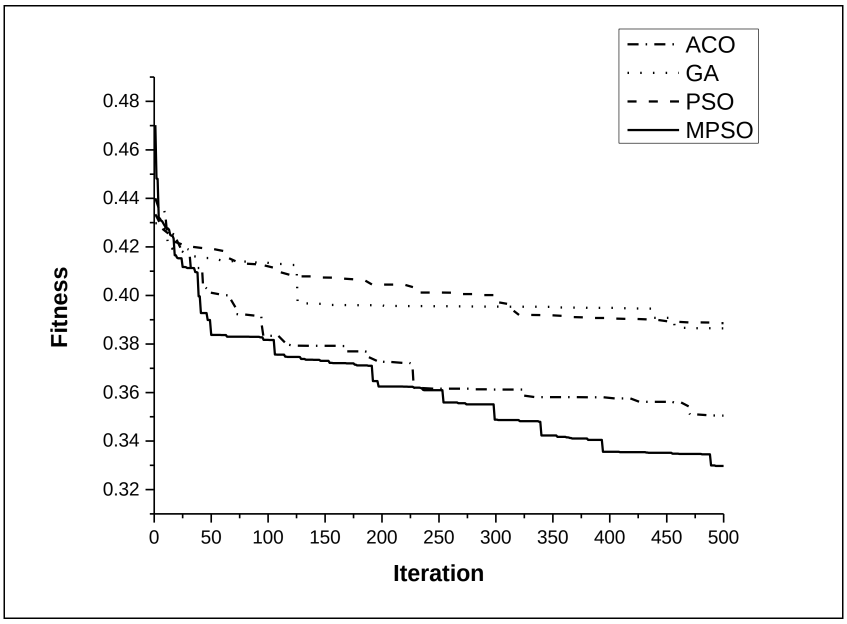

Figure 9. Convergence plot of large size problem

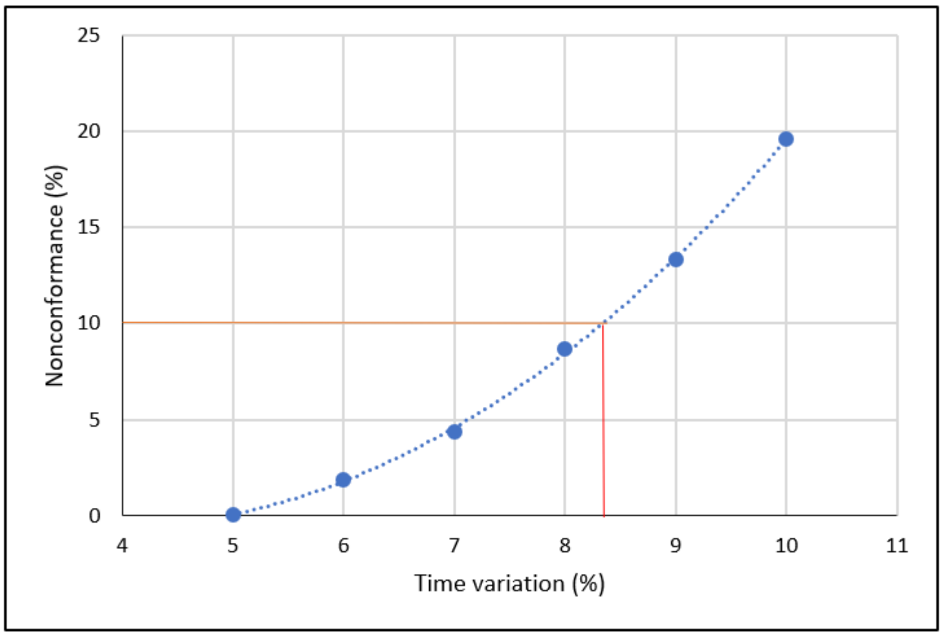

Figure 10. Sensitivity of optimised layout

LIST OF TABLES

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

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

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

Table 5. Test problem category and sources

Size	Problem	Number of task	Data source
Small	T4	4	[7]
	T9	9	[18]
	T12	12	[18]
	T16	16	[19]
Medium	T24	24	[18]
	T47	47	[35]
	T65	65	[19]
	T70	70	[3]
Large	T83	83	[34]
	T111	111	[34]
	T148	148	[5]
	T205	205	[19]

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**	***f₁***	***f₂***	***f₃***	***f₄***
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

Table 10. Assembly data for 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

Table 11. Assembly task assignment for existing and optimised layouts

Layout	**Station**	**Task**	**Time (minute)**	**Resource**	**Cycle time (minute)**	**Total idle (minute)**
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		

Biographies

Muhammad Razif Abdullah Make is a part time MSc graduate researcher in 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 in 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.