JOURNAL OF MODERN MANUFACTURING SYSTEMS AND TECHNOLOGY (JMMST)
e-ISSN: 2636-9575
VOL. 4, ISSUE 2, 73 – 83
DOI: https://doi.org/10.15282/jmmst.v4i2.4787

ORIGINAL ARTICLE

# Assessment of metaheuristic algorithms to optimize a mixed-model assembly line balancing problem with resource constraints

M.M. Razali[1], M.F.F. Ab. Rashid[1,*] and M.R.A. Make[1]

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

**ABSTRACT** – Mixed- model assembly line balancing problem (MMALBP) is an NP-hard problem which requires an effective algorithm for solution. In this study, an assessment of metaheuristic algorithms to optimize MMALBP was conducted by using four popular metaheuristics , namely particle swarm optimization (PSO), simulated annealing (SA), ant colony optimization (ACO), and genetic algorithm (GA). Three categories of test problem (small, medium, and large) were used, ranging from 8 to 100 tasks. For computational experiment, MATLAB software was used to investigate the metaheuristic algorithm performances to optimize the designated objective functions. Results revealed that the ACO algorithm performed better in terms of finding the best fitness functions when dealing with many tasks.Averagely, it produced better solution quality than PSO by 5.82%, GA by 9.80%, and SA by 7.66%. However, PSO was more superior in terms of processing time as compared to ACO by 29.25%, GA by 40.54%, and SA by 73.23%.Therefore, future research directions, such as by using the actual manufacturing assembly line data to test the algorithm performances, are likely to happen.

**ARTICLE HISTORY**
Revised: 7th October 2020
Accepted: 8th October 2020

**KEYWORDS**
*Mixed- model assembly*
*Line balancing*
*Metaheuristic optimization*

## INTRODUCTION

Assembly line balancing problem (ALBP) is a matter of decisions that arise when designing or redesigning the assembly line. It involves finding the optimum assignment of tasks. In recent decades, ALBP has become one of the major interesting research subjects due to its importance in current manufactures. The subject is important because manufacturers are able to increase efficiency, productivity, as well as gain profit and reduce operational cost by applying assembly line balancing [1]. Mixed- model assembly line balancing problem (MMALBP) is categorized under ALBP. It differs from other ALBP classification problems because it deals with an assembly line that is capable of assembling more than one product model at the same time [2].

In MMALBP context, various other factors are considered, for example, number of models which will be assembled and total demands throughout the planning horizon. To propose an effective solution for this problem, a few methods were used by previous researchers. For instance, a mathematical approach, namely mixed integer linear programming and mathematical programming techniques, were utilized to optimize MMALBP [3]. However, problems associated with the use of optimization in large scale problems frequently reach local optimum, especially when faced with NP-hard problems. The NP-hard problem contains a massive number of variables as well as non-linear objective functions, which make it complicated for the conventional method to deliver a decent solution [4].

To counter this problem, some alternative solutions were proposed. Meta-heuristic algorithms were presented to find a near-optimal solution for MMALBP. Metaheuristic algorithms is a stochastic optimizer programming that is capable of solving multi-objective optimization problems. The algorithms can manage multi-objective problems with a set of possible solutions simultaneously [5]. The algorithms can find the near-optimal solution in a single run as compared to traditional techniques, which need to be executed in a series of separate runs.

The metaheuristic algorithms imitate the metaphor of natural biological evolution and social behavior of species [6]. Examples for that metaphorare ants searching for the shortest route that will lead to a food source , and how a flock of birds work together to get to their destination during migration. However, the first reported metaheuristic algorithm in previous literature was the genetic algorithm inspired from Darwin's principle, which was based on natural evolution in the 1970s [7]. Similar to the metaphor previously mentioned, simulated annealing optimization technique is to imitate the physical process of annealing, which is also known as heat treatment process, whereby a metal is heated to a specific temperature and then allowed to cool gradually [8].

In the interest to mimic the behavior of this species effectively, various researchers developed a computerized system that was capable of finding solutions for complex optimization problems. It was encouraged by the aforementioned natural biological evolution and social behavior of species, such as the ant colony optimization and particle swarm optimization. Then, much attention was given to the metaheuristic algorithms performance measurement to generally verify the applicability of a particular algorithm to that particular ALBP. The objective functions were used as the evaluation criteria

*CORRESPONDING AUTHOR | M.F.F. Ab Rashid | ffaisae@ump.edu.my

to measure the studied metaheuristic algorithm performance . The most common objective functions being studied in existing literature are minimize cycle time [9], minimize number of workstation [10], minimize total idle cost [11], and balance workload [12].

Despite these prior efforts by researchers, some limitations on the existing works. These include the assumption that all workstations have the same capability in terms of resources, such as tool, manpower, and machine . Furthermore, lack of consideration for existing resources on the manufacturing line, including recent studies only considered specific resource constraints in their research.

This paper investigates the mixed-model assembly line balancing problem (MMALBP) with resource constraints. Then, a computational study to compare the performance of four metaheuristic algorithms in terms of fitness value, processing time, and quality of solutions were conducted. According to a recent study, the most popular metaheuristic algorithm to optimize MMALBP was genetic algorithm (GA) [13]. This was followed by simulated annealing (SA), ant colony optimization (ACO), and particle swarm optimization (PSO). In addition, these algorithms were proven to optimize other variants of line balancing problems. Therefore, for assessment purpose, these four metaheuristic algorithms will be used to optimize MMALBP. Three objective functions were chosen, which were minimize total cycle time, minimize product rate variation, and minimize number of resources used on the assembly line. The selection of these three objective functions were based on MMALBP literature, whereby it is the most studied by previous research.

## PROBLEM MODELING

To evaluate the performance of selected metaheuristic algorithms, a problem modeling was constructed with the objective functions to minimize cycle time, minimize product rate variation (PRV) and resources used on the assembly line [9]. Mixed-model assembly line consists of more than one product which will be assembled on the same line. Each model has its own precedence relation diagram that shows an arrangement of tasks which needed to be completed to produce the final finished product. A joint precedence diagram was formed from a combination of two or more product models. A simple illustration on how the join precedence was formed is presented in Figure 1.

Each model has six tasks which need to be completed but may be different in terms of task arrangement. For example, in Model 1, Task 2 and Task 3 must be finished first before it can proceed with Task 5. Meanwhile, in Model 3, only Task 2 is needed to be completed before moving to Task 5. Then, Figure 2 shows how the joint precedence diagram is formed based on these three models. It shows the proposed sequence of the task that must be followed to assemble the products from start tofinish. All known solutions for MMALBP rely on the joint precedence diagram, which is crucial in solving such problems [14].

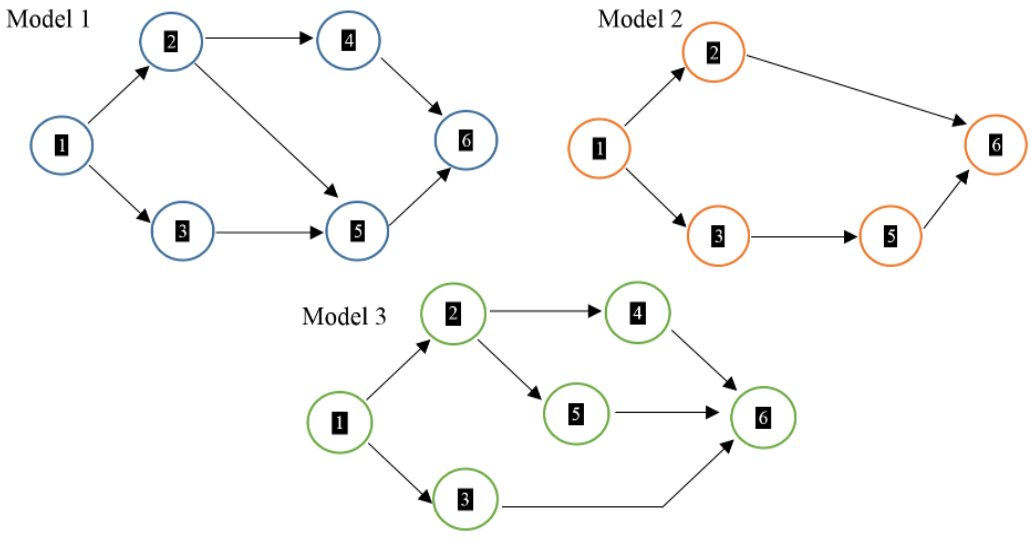


**Figure 1.** Precedence diagram for Model 1, Model 2, and Model 3.

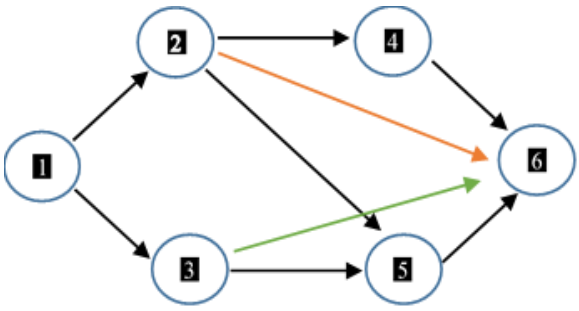


**Figure 2.** Joint precedence diagram.

The parameters and indices of the model will be as follows:

| Notation | Definition |
|---|---|
| $S$ | number of workstations(fixed) $s = 1, 2\ldots, S$ |
| $J$ | number of product models to be assembled $j = 1, 2\ldots, J$ |
| $Ne$ | number of task $e = 1, 2\ldots, Ne$ |
| $pre_i$ | predecessor for task $i$ based on precedence diagram |
| $t_i$ | execution time for task $i$ |
| $D_T$ | total quantity of units or total demand |
| $d_j$ | demand for product $j$, $j = 1, 2, \ldots, a$ |
| $X_{i,k}$ | total quantity of product/produced over stages 1 to $k$, $k = 1, 2, \ldots, Dt$ |
| $maxR$ | maximum resources $r = 1, 2,\ldots, maxR$ |
| $C_T$ | cycle time |
| $T_{ej}$ | shift task model time |
| $T_e$ | shift task time |
| $t_e$ | task time |
| $N_j$ | demand schedule for each model |
| $U$ | production rates variation of production sequence |
| Decision variables | |
| $U_{ej}$ | 1 if task, $e$ is used on model $j$ ; 0,otherwise |
| $X_{es}$ | 1 if task, $e$ is assigned to workstation $s$; 0, otherwise |
| $Y_{rs}$ | 1 if resource, $r$ is used in workstation $s$; 0,otheriwse |

## Objective functions and constraints

In this paper, three objective functions were used as the evaluation criteria. First was tominimize the cycle time. Second was to minimize product rate variation (PRV) while the third was to minimize resources used on the assembly line. These selected objective functions and their related constraints were formulated as below [9]:

$$f1 = min \sum_{e=1}^{N_e} \sum_{j=1}^{J} C_T \tag{1}$$

$$f2 = min \sum_{s=1}^{S} \sum_{r=1}^{max_R} Y_{rs} \tag{2}$$

$$f3 = \min \sum_{k=1}^{D_T} \sum_{j=1}^{J} \left( x_{j,k} - k \times \frac{d_j}{D_T} \right)^2 \tag{3}$$

Objective function *f1*, in Equation (1) is to minimize cycle time. Meanwhile, *f2* in Equation (2) aims to minimize resources used on assembly line and *f3* in Equation (3) is to minimize product rate variation (PRV) based on the demand of planning horizon. These three objective functions were bound by the following restrictions:

$$\sum_{s=1}^{S} X_{as} - \sum_{s=1}^{S} X_{bs} \leq 0, for\ \forall (a,b) \in pre_i \tag{4}$$

$$\sum_{i \in wk} t_i(X_{es}) \leq C, \qquad s = 1, \dots, S \tag{5}$$

$$\sum_{s=1}^{S} X_{es} = 1, \qquad e = 1, \dots, n \tag{6}$$

Constraint (4) is to assure that the precedence constraints among the tasks is followed, which is to guarantee that no successor task is appointed to an earlier station. Meanwhile, Inequality (5) is to ensure that total task times assigned to each station does not surpass the designated maximum cycle time. Restriction of each task can only be assigned to one workstation, which is created by using Constraint (6). The maximum cycle time mentioned in this paper was stated as reference cycle time, $Ref_{CT}$ ,and can be expressed as:

$$Ref_{CT} = \frac{\sum shift\ task\ time, T_e}{no.\ of\ workstation, s}, \quad s = 1, \dots, S \tag{7}$$

A multi-objective optimization was involved as a result of multiple objective functions being considered in this paper.Therefore, a weighted sum approach was employed to give better control on the final output based on preferment. The approach is expressed as follows:

$$\sum_{i=1}^{M} w_i f_i(x) \quad ; w_1 f_1(x) + w_2 f_2(x) + \cdots + \omega_n f_n(x) \tag{8}$$

For the general purpose, all three objective functions need to be normalized in Equation (8) to provide a constant proportion for each objective function. This can be accomplished by distributing the fitness value with the maximum value for every objective function measured. After applying the weighted sum approach, the normalized fitness functions is represented in Equation (9). For this purpose, $w_1$, $w_2$, and $w_3$ were set to 0.33 each to give equal weightage for all objective functions. Furthermore, there was no prior knowledge on the objective function preference, especially for resource utilization.

$$F(X) = w_1 f_1'(x) + w_2 f_2'(x) + w_3 f_3'(x) \tag{9}$$

## OPTIMIZATION ALGORITHM

In general, metaheuristic algorithms share the same approach in their application for a given problem. Firstly , the problem needs some representation in accordance with each method. Then, metaheuristic search algorithms are used iteratively to reach a solution that is near-optimal. Based on literature review that governs mixed-model assembly line balancing problems, GA, PSO, ACO, and SA were the most studied algorithms by previous researchers [13]. This can be ranked by GA as the most used algorithm in solving MMALBP, followed by SA, ACO, and PSO in second, third, and fourth, respectively. This is among the possible reasons why these four metaheuristic algorithms were chosen for comparison [15]. The following subsections present a brief description governing these four metaheuristic algorithms framework.

### Ant colony optimization

ACO algorithms evolve not only in their designated genetics, but also in their social behavior. Looking back into the history of ACO, it led to Marco Dorigo [16] who first developed this algorithm,which was taken from the metaphor on how ants are able to search their source of food and nest by using the shortest route. The real framework on ACO algorithm is by using pheromone trails, which are scientifically deposited by ants when they navigate to find sources of food. These pheromone trails are used as some sort of communication medium between the ants.

At the point when ants leave their homes to look for sources of food, they arbitrarily turn around an obstacle, and on the primary store of pheromone will be the same for the left and right directions. However, when the ants in the shorter direction discover food, they will carry it together and start to return, following their pheromone trails, and still spare more pheromone. As indicated in this figure, an ant will most likely choose the shortest route when returning to its home with food as this path has the most deposited pheromones. The pseudocode for ACO is as follows:

**Begin**
**Initialize** pheromone values and parameters
**While** stopping condition not satisfied, **Do**
Create all ants solution
Perform local search
Evaluate trail
Update pheromone value
**End While**
**End**

### Particle swarm optimization (PSO)

Based on the original literature, PSO was originally invented in the mid-1990s by Kennedy and Eberhart [17]. PSO was inspired by the behavior of a flock of birds on their journey to find sources of food. Their social behavior helps them to adapt to the current environment as well as avoid predators by using an approach called 'information sharing'; hence, created the evolutionary advantage. The pseudocode for PSO is presented as follows. In PSO, the updating procedure relies on Equation (10) and Equation (11). In Equation (10), $\omega$, $c_1$, and $c_2$ are the inertia, cognitive, and social coefficients, respectively.

**Begin**
**Initialize** velocity, position and parameters
**While** stopping condition not satisfied, **Do**
Evaluate position
Update *Pbest* and *Gbest*
Update velocity
Update position
**End While**
**End**

### Genetic algorithms (GA)

GA was recorded as the first evolutionary algorithm presented by John Holland in the 1970s. Inspired from the 'survival of the fittest' principle, it was developed in a way for over a number of generations, through which the populations evolved. Naturally, an individual who possesses the highest survival rate is likely to have a larger number of offsprings. Therefore, in each succeeding generation, the genes from the fittest individuals will increase in number. In this manner, the species becomes more and more well-adapted to their current environment as they evolve [18]. Below is the pseudocode for GA.

**Begin**
**Initialize** population of solution
**While** stopping condition not satisfied, **Do**
Evaluate solution
Selection
Crossover
Mutation
**End While**
**End**

### Simulated annealing (SA)

SA algorithm is a meta-heuristic search technique which was first invented by Kirkpatrick Gelatt, and Vecchi in 1983. It served a purpose for solving NP-hard optimization problems, specifically to enhance the objective functions value. In fact, the 'annealing' term comes from the concept of annealing process used in metallurgical industry. Annealing is a process of slow cooling cast-off to metals to get a low energy-state crystallization and produce a better aligned finished metal product. The optimization procedure of SA searches for a near-optimum solution which impersonate the slow cooling procedure in the physical annealing process [19]. The pseudocode for SA is shown as follow.

**Begin**
**Initialize** random initial temperature
**While** stopping condition not satisfied, **Do**
Evaluate solution
Update stored solution
Adjust temperature
**End While**
**End**

## RESULTS AND DISCUSSIONS

To evaluate the performance of ACO, GA, PSO, and SA algorithms to an extend limit, a benchmark dataset in MMALBP must be tested. The test problems used in this paper were taken from the website http://www.assembly-line-balancing.de under the categories of mixed-model assembly line balancing problem. In addition, the test problems were widely used to test the algorithms in searching for quality solution to MMALBP.

The dataset contained small-sized, medium-sized and large-sized test problems that ranged from 8 to 100 tasks. Specifically, small-sized problem contains from 8 to 20 tasks, medium-sized problem ranges from 25 to 50 tasks, while large-sized problem has from 60 to 100 tasks. All these four algorithms were developed based on -their own features and targeted to optimize the selected objective functions.

Therefore, these four metaheuristic algorithms were tested by using the chosen test problems through MATLAB simulation. For experimental purpose, the number of population for all algorithms was set to 30, while the maximum iteration was 100. Environment of the computational experiment included: Intel(R) Core (TM) i7 2.40GHz, 8 GB memory, Windows 8.1. Considering that all four algorithms might be influenced by random characteristics, each optimization process was run for 20 times under the same parameter and experimental environment. The number of repetition runs used in this work was considered acceptable since the earlier works had applied between 5 to 30 repetition runs in their studies [20]–[22]. Mean from the test results of each algorithm were listed. The result from this comparison of performance is presented as follows:

**Table 1.** Computational result for small-sized problem.

| Problem (no. of task) | Algorithm | PSO | ACO | SA | GA |
|---|---|---|---|---|---|
| Bowman (8) | Min Fitness | 0.9443 | **0.8166** | 0.9443 | 0.8166 |
| | Max Fitness | 0.9443 | **0.8166** | 0.9443 | 0.8193 |
| | Mean Fitness | 0.9443 | **0.8166** | 0.9443 | 0.8169 |
| | Std Deviation | 0 | 0 | 0 | 0.0007 |
| | Mean CPU time | **82.9557** | 88.0041 | 98.2792 | 89.314 |
| Mansoor (11) | Min Fitness | 0.7051 | **0.5773** | 0.7051 | 0.5773 |
| | Max Fitness | 0.7051 | **0.5773** | 0.7051 | 0.5787 |
| | Mean Fitness | 0.7051 | **0.5773** | 0.7051 | 0.5775 |
| | Std Deviation | 0 | 0 | 0 | 0.0004 |
| | Mean CPU time | **49.387** | 107.9176 | 161.7723 | 143.5887 |
| Mitchell (20) | Min Fitness | **0.7026** | 0.9499 | 0.9443 | 0.8166 |
| | Max Fitness | **0.9077** | 0.9513 | 0.9443 | 0.8179 |
| | Mean Fitness | **0.812** | 0.95 | 0.9443 | 0.8166 |
| | Std Deviation | 0.0727 | 0.0004 | **0** | 0.0003 |
| | Mean CPU time | **52.3941** | 131.1931 | 161.1315 | 140.1623 |

**Table 2.** Computational result for medium-sized problem.

| Problem (no. of task) | Algorithm | PSO | ACO | SA | GA |
|---|---|---|---|---|---|
| Buxey (29) | Min Fitness | 0.7138 | **0.6395** | 0.6538 | 0.8658 |
| | Max Fitness | 0.9634 | **0.7517** | 0.8493 | 0.9268 |
| | Mean Fitness | 0.8401 | **0.6969** | 0.7906 | 0.88 |
| | Std Deviation | 0.0454 | **0.0272** | 0.0354 | 0.0275 |
| | Mean CPU time | **101.4522** | 101.7651 | 263.718 | 243.4568 |
| Gunther (35) | Min Fitness | 0.5888 | 0.7368 | **0.5842** | 0.8318 |
| | Max Fitness | 0.9742 | **0.8427** | 0.8743 | 0.9671 |
| | Mean Fitness | 0.7973 | 0.7792 | **0.6199** | 0.9242 |
| | Std Deviation | 0.1101 | **0.0351** | 0.0673 | 0.0599 |
| | Mean CPU time | 137.1709 | **136.9874** | 257.1917 | 153.3631 |
| Kilbridge (45) | Min Fitness | **0.5673** | 0.6888 | 0.6499 | 0.6182 |
| | Max Fitness | **0.7318** | 0.8304 | 0.7834 | 0.7723 |
| | Mean Fitness | 0.6359 | 0.76 | 0.7162 | 0.6106 |
| | Std Deviation | 0.0464 | **0.0204** | 0.0382 | 0.0697 |
| | Mean CPU time | **276.8158** | 337.7423 | 658.4571 | 405.0386 |

**Table 3.** Computational result for large-sized problem.

| Problem (no. of task) | Algorithm | PSO | ACO | SA | GA |
|---|---|---|---|---|---|
| Wee-Mag (75) | Min Fitness | 0.6947 | **0.6475** | 0.6742 | 0.6505 |
| | Max Fitness | 0.8411 | 0.7859 | 0.8247 | **0.7595** |
| | Mean Fitness | 0.769 | **0.7513** | 0.763 | 0.7777 |
| | Std Deviation | 0.0397 | **0.0329** | 0.0393 | 0.0458 |
| | Mean CPU time | 1038.831 | **1024.288** | 1090.052 | 1365.338 |
| Arc (83) | Min Fitness | 0.5274 | **0.3807** | 0.5279 | 0.7264 |
| | Max Fitness | 0.6475 | **0.4835** | 0.6503 | 0.9647 |
| | Mean Fitness | 0.5903 | **0.4412** | 0.5832 | 0.7953 |
| | Std Deviation | 0.0316 | **0.0286** | 0.029 | 0.0614 |
| | Mean CPU time | **1632.653** | 2631.114 | 3044.799 | 6313.902 |
| Mukherjee (94) | Min Fitness | 0.5527 | **0.4858** | 0.4974 | 0.5772 |
| | Max Fitness | 0.7298 | **0.6743** | 0.7358 | 0.8384 |
| | Mean Fitness | 0.6525 | **0.6107** | 0.6161 | 0.7012 |
| | Std Deviation | 0.0517 | **0.0510** | 0.071 | 0.0874 |
| | Mean CPU time | **2148.149** | 2389.164 | 2897.187 | 3650.265 |

Based on the results presented in Table 1, Table 2, and Table 3, the values in bold were the best value in each measured parameter. In the perspective of problem category which included small, medium, and large size test problem, each metaheuristics resulted in different solution qualities, which will be discussed next.

Referring to small-sized problems in Table 1, the result varied among ACO, PSO, SA, and GA. The results were measured based on minimum fitness, maximum fitness, mean fitness, standard deviation, and CPU time. For instance, in Bowman's problem, ACO and GA gave the best value for minimum fitness. Meanwhile, for maximum fitness and mean fitness ACO performed better as compared to the other three metaheuristics. On the other hand, for PSO, mean CPU time was the most superior in this problem. Likewise, the same result analysis was produced for Mansoor's problem. However, different results were found for Mitchell's problem with 20 tasks. PSO was at its best performance in this problem when it performed better for minimum fitness, maximum fitness, mean fitness and mean CPU time than the other three

metaheuristics. The only parameter that was lack of was standard deviation, which gave an early view that even though it performed better, the solution quality might vary for each single run.

Moving from small-sized, the medium-sized problem analysis diplayed in Table 2 gave a more diversified result. For example, better solution quality for Buxey's problem pointed to the ACO algorithm, but mean CPU time came off second best to PSO. Meanwhile, for Gunther's problem, SA algorithm outperformed its competitors by yielding better minimum fitness and mean fitness, but lack of ACO in terms of maximum fitness, standard deviation, and mean CPU time. Next in Kilbridge's problem, the PSO algorithm offered better minimum fitness, maximum fitness, and mean CPU time, whereas ACO presented better standard deviation, with GA producing the best mean fitness.

Result tabulation in Table 3 represents large-sized problem which contain the highest number of tasks in this study, ranging from 65 to 100 tasks. Wee-Mag's problem showed that the ACO algorithm achieved better result for each parameter, except for maximum fitness, which came off second best to GA. Based on Arc's problem with 83 tasks, the ACO algorithm performed better in terms of minimum fitness, maximum fitness, mean fitness, and standard deviation than the other three metaheuristic algorithms. The only shortcomings was in terms of its mean CPU time, which trailed behind the PSO algorithm. Lastly, all four algorithms were tested by using Mukherjee's problem that contained 100 tasks. Once again, the ACO algorithm resulted with a better solution quality and only lacked in terms of mean CPU time to PSO. In addition, large-sized problem gave a better view on which metaheuristic algorithms finished the optimization with a constant solution quality over multiple runs. The best standard deviation value in large-sized problem was clearly dominated by the ACO algorithm; hence, presented an early view in terms of its solution reliability as compared to PSO, GA, and SA.

A pattern could be identified in this experiment, in which PSO and ACO performed better when dealing with small-sized and medium-sized problems, covering a range from 8 to 50 tasks. However, when tested with large-sized problem, the PSO algorithm performance was decreased. On the other hand, the ACO algorithm proved its reliabilty when solving large-sized problems as compared to the other three algorithms. In this experiment, the phenomenon occured in PSO algorithm. However, it could be related to the mechanisms of PSO itself. Since large-sized problem were used, it brought along a larger solution space which needed to be explored. The designated inertia, ω in the PSO framework occurred and affected the solution quality over generations. Generally, ω is equal to 1, then at the later period of the several generations, there was lack in searching ability of the particle for a better solution quality, which produced a poorer result for PSO [23]. To calculate the percentage difference of performances by the four metaheuristic algorithms in every category, average value for the best fitness solution is tabulated in **Error! Reference source not found.**. Averagely, ACO produced better solution quality than PSO by 5.82%, GA by 9.80%, and SA by 7.66%. However, PSO was more superior in terms of processing time than ACO by 29.25%, GA by 40.54%, and SA by 73.23%.

**Table 4.** Mean of solution quality by each metaheuristic algorithms for every category.

| | Problem Category | ACO | PSO | GA | SA |
|---|---|---|---|---|---|
| Average solution quality | Small | 0.7813 | 0.8204 | **0.7703** | 0.8645 |
| | Medium | **0.7453** | 0.7577 | 0.8049 | 0.7789 |
| | Large | **0.6010** | 0.6705 | 0.7581 | 0.6541 |
| Average CPU time (sec) | Small | 327.114 | **184.736** | 373.065 | 421.183 |
| | Medium | 776.493 | **705.438** | 801.858 | 1662.367 |
| | Large | 6044.56 | **4819.633** | 7329.505 | 9032.038 |

As mentioned earlier, each optimization for each metaheuristic algorithm was repeated 20 times and each single run returned the best fitness. Therefore, to plot the convergence graph, mean value of solution output for each algorithm was used. Figure 3, Figure 4, and Figure 5 present the mean convergence of four metaheuristic algorithms for each category.

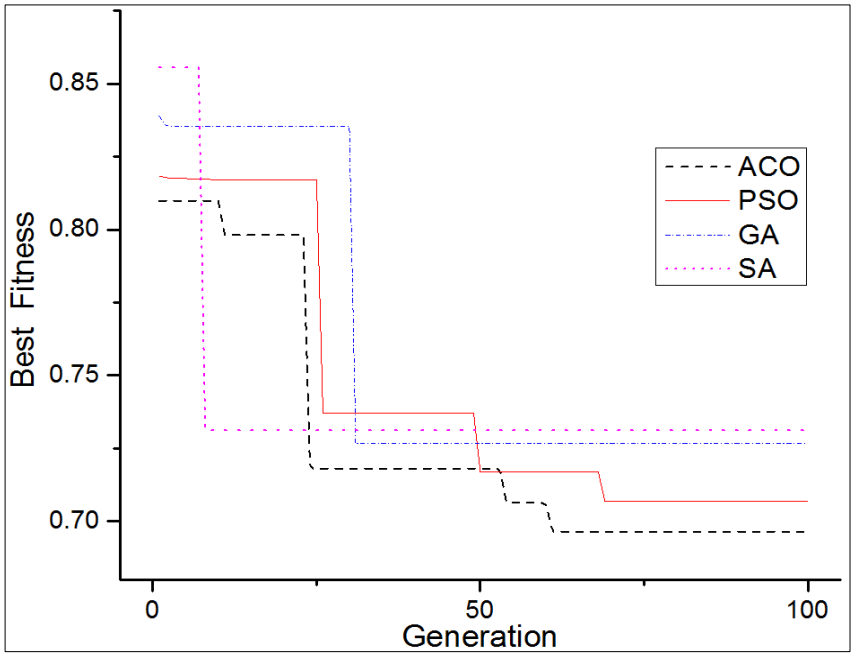


**Figure 3.** Convergence plot for small-sized problem

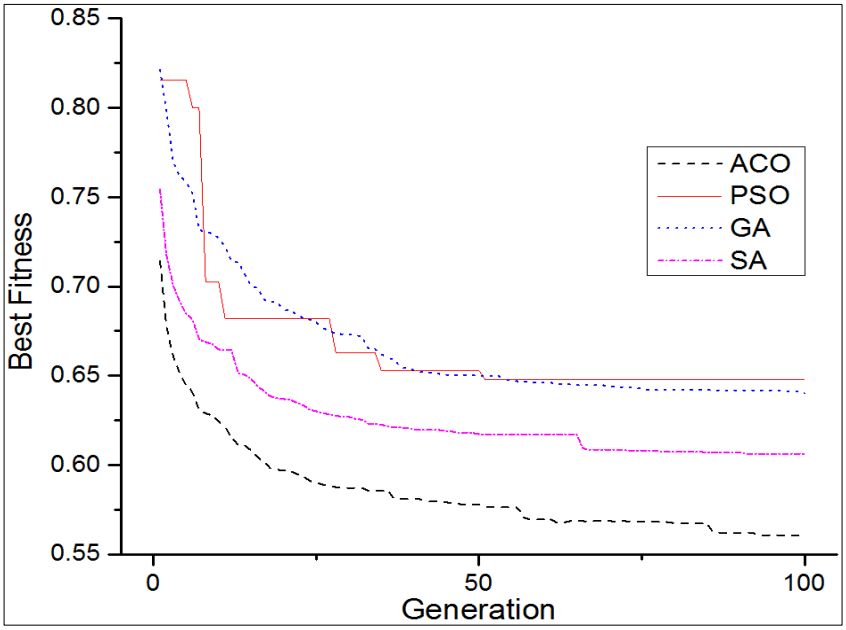


**Figure 4.** Convergence plot for medium-sized problem

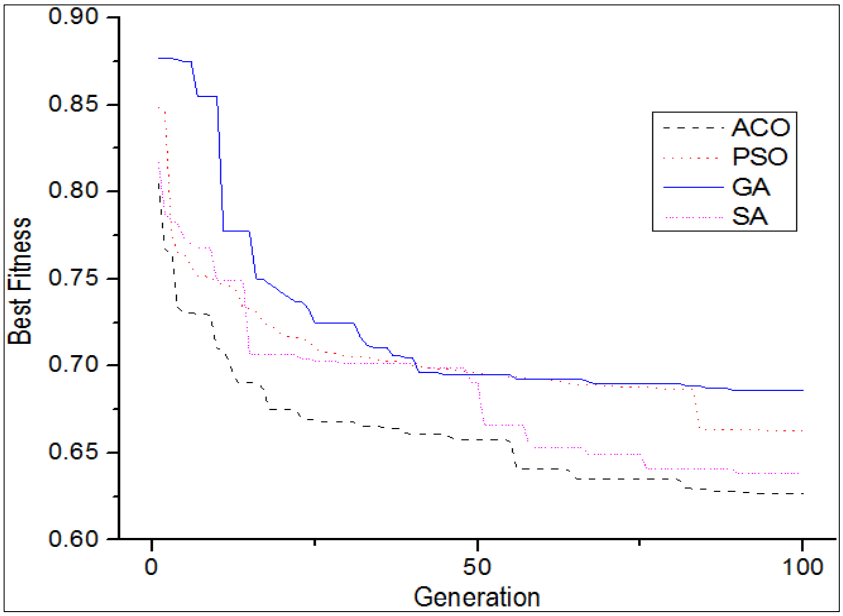


**Figure 5.** Convergence plot for large-sized problem

Based on the graph in Figure 3, all algorithms showed rapid convergences, even though ACO presented better performance. The rapid convergence in this class of problem was due to the small search space in small-sized problems. In contrast, with the increasing number of tasks from small-sized problem to medium and large-sized problem, the search space became larger. Therefore, the convergence occured more frequently since the choices of solutions were larger.

## CONCLUSION

The performance represented by fitness function and processing time of the four metaheuristic algorithms (ACO, PSO, GA, and SA) in MMALBP with resource constraint was presented. Comparison between these algorithms were made and selected objective functions were evaluated, which were to minimize cycle time, minimize product rate variation, and minimize resources used on the assembly line. One of the significant findings to emerge from this study suggested that the ACO algorithm performed better in terms of finding the best fitness functions when dealing with many tasks.

However, the present study has some limitations, which included the parameter settings for algorithms. In this study, the parameter tuning was not considered. A proper parameter setting for the algorithms could possibly return a better solution quality. Further investigation can also be implemented by considering the actual assembly line such as in manufacturing industry. Based on the listed limitations, a further experimentation to measure the performance of these four metaheuristic algorithms is strongly recommended. Besides, it would be interesting to assess the effects of increasing the number of test problems to the solution quality produced, as well as the results when applying to the actual problems in an actual assembly line.

## ACKNOWLEDGMENT

This research was supported by a research fund from the Ministry of Higher Education under RDU1901108 (FRGS/1/2019/TK03/UMP/02/3). In addition, the authors would like to thank Universiti Malaysia Pahang for providing the facilities.

## REFERENCES

[1] M. Rabbani, M. Moghaddam, and N. Manavizadeh, "Balancing of mixed-model two-sided assembly lines with multiple U-shaped layout," *Int. J. Adv. Manuf. Technol.*, vol. 59, no. 9–12, pp. 1191–1210, Apr. 2012, doi: 10.1007/s00170-011-3545-6.

[2] Z. Yuguang, A. Bo, and Z. Yong, "A PSO algorithm for multi-objective hull assembly line balancing using the stratified optimization strategy," *Comput. Ind. Eng.*, vol. 98, pp. 53–62, 2016, doi: 10.1016/j.cie.2016.05.026.

[3] I. Kucukkoc and D. Z. Zhang, "Mathematical model and agent based solution approach for the simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines," *Int. J. Prod. Econ.*, vol. 158, pp. 314–333, 2014, doi: https://doi.org/10.1016/j.ijpe.2014.08.010.

[4] O. Battaïa *et al.*, "Workforce minimization for a mixed-model assembly line in the automotive industry," *Int. J. Prod. Econ.*, vol. 170, pp. 489–500, Jun. 2015, doi: 10.1016/j.ijpe.2015.05.038.

[5] X. Zhao, C.-Y. Hsu, P.-C. Chang, and L. Li, "A genetic algorithm for the multi-objective optimization of mixed-model assembly line based on the mental workload," *Eng. Appl. Artif. Intell.*, vol. 47, pp. 140–146, Jan. 2016, doi: 10.1016/J.ENGAPPAI.2015.03.005.

[6] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Adv. Eng. Informatics*, vol. 19, no. 1, pp. 43–53, Jan. 2005, doi: 10.1016/j.aei.2005.01.004.

[7] S. Akpınar and G. M. Bayhan, "A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints," *Eng. Appl. Artif. Intell.*, vol. 24, no. 3, pp. 449–457, 2011, doi: 10.1016/j.engappai.2010.08.006.

[8] N. Manavizadeh, N. Hosseini, M. Rabbani, and F. Jolai, "Computers & Industrial Engineering A Simulated Annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach," *Comput. Ind. Eng.*, vol. 64, no. 2, pp. 669–685, 2013, doi: 10.1016/j.cie.2012.11.010.

[9] M. M. Razali, M. Fadzil, F. Ab, and M. Razif, "Mathematical Modelling of Mixed-Model Assembly Line Balancing Problem with Resources Constraints," vol. 012002, doi: 10.1088/1757-899X/160/1/012002.

[10] B. Yagmahan, "Mixed-model assembly line balancing using a multi-objective ant colony optimization approach," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 12453–12461, Sep. 2011, doi: 10.1016/j.eswa.2011.04.026.

[11] N. Manavizadeh, L. Tavakoli, M. Rabbani, and F. Jolai, "A multi-objective mixed-model assembly line sequencing problem in order to minimize total costs in a Make-To-Order environment, considering order priority," *J. Manuf. Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013, doi: 10.1016/j.jmsy.2012.09.001.

[12] Y. Kara, U. Ozcan, and A. Peker, "Balancing and sequencing mixed-model just-in-time U-lines with multiple objectives," *Appl. Math. Comput.*, vol. 184, no. 2, pp. 566–588, Jan. 2007, doi: 10.1016/j.amc.2006.05.185.

[13] M. M. Razali, N. H. Kamarudin, M. F. F. Ab. Rashid, and A. N. Mohd Rose, "Recent trend in mixed-model assembly line balancing optimization using soft computing approaches," *Eng. Comput.*, vol. 36, no. 2, pp. 622–645, Jan. 2019, doi: 10.1108/EC-05-2018-0205.

[14] A. a. Mamun, A. a. Khaled, S. M. Ali, and M. M. Chowdhury, "A heuristic approach for balancing mixed-model assembly line of type I using genetic algorithm," *Int. J. Prod. Res.*, vol. 50, no. 18, pp. 5106–5116, 2012, doi: 10.1080/00207543.2011.643830.

[15] R. Hassan and B. Cohanim, "A comparison of particle swarm optimization and the genetic algorithm," *1st AIAA Multidiscip. Des. Optim. Spec. Conf.*, pp. 1–13, 2005, doi: 10.2514/6.2005-1897.

[16] M. Dorigo and G. DiCaro, "The Ant Colony Optimization meta-heuristic," pp. 11–32, 1999.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*, no. 1. MIT CogNet, 1992.

[19] A. Hamzadayi and G. Yildiz, "A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines," *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 1070–1084, Dec. 2013, doi: 10.1016/j.cie.2013.08.008.

[20] Q. Bai, "Analysis of Particle Swarm Optimization Algorithm," *Comput. Inf. Sci.*, vol. 3, no. 1, pp. 180–184, 2010, doi: 10.5539/cis.v3n1P180.