Article:

Ab Rashid, M.F.F., Tiwari, A. and Hutabarat, W. orcid.org/0000-0001-7393-7695 (2019) Integrated optimization of mixed-model assembly sequence planning and line balancing using Multi-objective Discrete Particle Swarm Optimization. AI EDAM. ISSN 0890-0604

https://doi.org/10.1017/S0890060419000131

Integrated Optimisation of Mixed-Model Assembly Sequence Planning and Line Balancing using Multi-Objective Discrete Particle Swarm Optimisation

Mohd Fadzil Faisae Ab Rashid

Corresponding Author

Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Malaysia

Email: ffaisae@ump.edu.my

Telephone: +609-4246321

Ashutosh Tiwari

Manufacturing and Materials Department, Cranfield University, Bedford, MK43 0AL, United Kingdom.

Windo Hutabarat

Manufacturing and Materials Department, Cranfield University, Bedford, MK43 0AL, United Kingdom.

Short title: Integrated Optimisation of Mixed-Model ASP and ALB using MODPSO

Number of pages: 31

Number of tables: 10

Number of figures: 4

Abstract

Recently, interest in integrated Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) began to pick up because of its numerous benefits, such as the larger search space that leads to better solution quality, reduced error rate in planning and expedited product time-to-market. However, existing research is limited to simple assembly problem that only runs one homogenous product. This paper therefore model and optimise the integrated mixed-model ASP and ALB using Multi-objective Discrete Particle Swarm Optimisation (MODPSO) concurrently. This is a new variant of the integrated assembly problem. The integrated mixed-model ASP and ALB is modelled using task based joint precedence graph. In order to test the performance of MODPSO to optimise integrated mixed-model ASP and ALB, an experiment using a set of 51 test problems with different difficulty levels was conducted. Besides that, MODPSO coefficient tuning was also conducted to identify the best setting so as to optimise the problem. The results from this experiment indicated that the MODPSO algorithm presents significant improvement in term of solution quality towards Pareto optimal and demonstrates ability to explore the extreme solutions in the mixed-model assembly optimisation search space. The originality of this research is on the new variant of integrated ASP and ALB problem. This paper is the first published research to model and optimise the integrated ASP and ALB research for mixed-model assembly problem.

Keywords

Manufacturing systems, Assembly sequence planning, Line balancing, concurrent optimization, Particle swarm optimisation

1. Introduction

Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) are classified as important activities in assembly optimisation although it occurs in different stages (Marian, 2003). Recently, there are efforts to integrate and optimise both activities concurrently because of benefits of reduced planning error and reduced costing in manufacturing (Tseng and Tang, 2006). The use of integrated scheme in engineering provide huge benefits (Penciuc et al., 2016). A recent study that compared the sequential and integrated optimisation approaches for ASP and ALB concluded the integrated approach is preferable for better solution quality because of larger search space (Ab. Rashid, Tiwari and Hutabarat, 2017). Additionally, the integrated optimisation can also speed up time-to-market for a product (Lu and Yang, 2016).

Assembly line problems are categorised into simple and generalised assembly line balancing problem (Becker and Scholl, 2006). The simple assembly line balancing problem (SALBP) only considers the production of one homogeneous product on serial line layout, while the generalised assembly line balancing problem (GALBP) includes all of the problems that are not SALBP, such as mixed-model, parallel, U-shaped and two-sided lines with stochastic dependent processing times (Tasan and Tunali, 2008; Jusop and Ab Rashid, 2015).

There are works on optimisation of integrated ASP and ALB problem focusing on SALBP. Chen proposed a hybrid Genetic Algorithm to optimise integrated ASP and ALB, where GA is combined with heuristic search (Chen, Lu and Yu, 2002). Tseng and Tang studied combining ASP together with ALB based on assembly "connectors" (i.e. the connector basis) by using Genetic Algorithm. However, when using this approach, whenever the number of connectors is increased, a few of the parameters that govern GA performance need to be reset (Tseng and Tang, 2006). Another work by Tseng et al. on integrated ASP and ALB was done in 2008. This work adopted Hybrid Evolutionary Multi-objective Algorithms (HEMOA) that was also based on GA (Tseng et al., 2008). In recent work of integrated ASP and ALB optimisation, GA-based algorithms performed well in optimising the problem with low and medium difficulties. However, the performance of GA-based algorithms deteriorates when faced with high difficulty problems, especially for problems with large number of tasks (Ab Rashid, Hutabarat and Tiwari, 2012). Besides that, researcher was also implemented Ant Colony Optimisation (ACO) for integrated ASP and ALB (Lu and Yang, 2016). However, it was tested with only small tasks number.

There has been, thus far, no work on integrated ASP and ALB optimisation beyond SALBP type. This work therefore aims to initiate the optimisation of integrated ASP and ALB for GALBP, more specifically, the class of mixed-model assembly problems. A mixed-model assembly line runs different product models in arbitrarily intermixed sequence on a single assembly line (Roshani and Nezami, 2017). This type of assembly line is widely used in various industries to produce a wide variety of products (Zhu et al., 2012). The mixed-model assembly line is important in industry because of the significant cost savings made possible by sharing different model of products in the same assembly line. The mixed-model assembly line can also absorb significant fluctuation of demand of the different models using an assembly line (Hu et al., 2008). It is crucially important to set-up the assembly line for a long term period. Any changes on the existing the assembly line will incur a lot of cost to the manufacturer (Shankar, Summers and Phelan, 2017). Therefore, by integrating the ASP and ALB optimisation for mixed-model assembly, the benefits from integrated optimisation and mixed-model assembly can be obtained.

The integrated mixed-model ASP and ALB problem is more challenging compared to mixed-model ALB and integrated simple ASP and ALB. Separate ASP and ALB problems are individually categorised as NP-hard combinatorial problems, where the solution space are excessively increased when the number of task increased (Lin et al., 2012). When the optimisation of both activities is performed together, the problem difficulties will be increased since all the related factors such as geometric information, assembly tool and time are concurrently considered in this stage (Tseng et al., 2008). Furthermore, compared with simple assembly problem, it is more difficult to achieve optimum solution for all models in the mixed-model assembly problem (Becker and Scholl, 2006; Zhong, 2017). Therefore, a formulation of the integrated mixed-model ASP and ALB problem will be more challenging to solve and to optimise, when compared with optimisation of mixed-model ALB and also integrated ASP and ALB for simple assembly.

The main contribution of this work is a new model of integrated mixed-model ASP and ALB problem. Later, we implement Multi-Objective Discrete Particle Swarm Optimization (MODPSO) algorithm to optimise this problem. Section 2 presents the modelling of integrated mixed-model ASP and ALB, including the objective functions for this problem. Section 3 explains the mechanism of MODPSO algorithm. Section 4 presents the experimental design and performance indicators for optimisation algorithms. Section 5 presents the results of experiment and Section 6 discusses these results that analyse various algorithms to optimise integrated mixed-model ASP and ALB problems. Finally Section 7 concludes the findings from this work.

2. Integrated Mixed-model ASP and ALB

An example of a mixed-model assembly line is found in vehicle production, where the assembly line runs one specific car type, but with different model variants, such as right or left hand drive and manual or automatic transmission. In addition, some of the cars require additional accessories to fulfil specific customer requirements. In this assembly line, there is only one product, that is, a specific car type, but the assembly process will vary due to differences between models. Assembly problems are commonly represented by assembly precedence graphs and assembly data table. The precedence graph consists of a set of nodes and arcs that represent assembly tasks and their precedence constraints. The outgoing arc from node i towards node j meaning the assembly task i must be completed before starting the assembly task j. Meanwhile the assembly data table represent the assembly information such as assembly direction, tool and time for particular assembly tasks.

The most common approach to express the mixed-model assembly problem is by transforming the precedence graphs into a joint graph as used in many existing Mixed-model Assembly Line Balancing works (Kara et al., 2011; Buyukozkan et al., 2016). The joint graph represents the precedence constraint for all models.

When the precedence diagram of model y is represented by a graph $G_y = (V_y, C_y)$, where V_y is the set of tasks of model y and C_y is the set of precedence relations, the combined graph is G = (V,C), where $V = \cup_y V_y$ and $C = \cup_y C_y$. An arc (i, j) is redundant if there exists another path from i to j in G. The mixed-model defines the number of units to be produced from each model during a shift of T time units. The processing time of $y \in V$ is equal to the total time required for the processing of this task in a given mixed-model.

For example, an assembly line runs two model of product, Model A and Model B. The precedence graphs for both models are shown in Figure 1(a) and (b). To establish the joint graph, the follower for specific tasks in each models are bundled together in one graph. For example in Figure 1, the followers for task 1 in Model A are task 2 and 3, while task 3 and 4 in Model B. The combination of task 1 followers from both model are task 2, 3 and 4 as shown in joint graph. The joint graph is updated by removing the shortest repetitive routes from the graph. In example below, the route connecting task 4 and 7 in Model B is removed from the Joint Model because task 7 cannot be started although task 4 has been performed, because there is dependence on completion of task 6 in Model A. Once the joint graph has been established, similar representation scheme as in simple assembly line problem can be used, except for assembly data representation.

[Figure 1: Precedence graph of (a) Model A, (b) Model B and (c) Joint Model]

In mixed-model assembly line, the assembly data set should represent data for each model. In this case, the assembly data for similar tasks within different models might be different, depending on the actual processing task.

2.1 Objective Functions and Constraints

There are known objective functions to evaluate single-model assembly problems. To evaluate the fitness in mixed-model assembly problem, the objective function is evaluated for every model, and the mean of these values is used as the fitness value. For mixed-model assembly problem with M model:

Objective 1: Minimise the mean of the total direction changes

$$\overline{n}_{dc} = \frac{1}{M}\sum_{m=1}^{M}\sum_{s=1}^{n-1} d_s^m; \qquad d_s^m = \begin{cases} 1 \text{ if direction } s \neq \text{direction } s+1 \text{ for } m^{\text{th}} \text{ model} \\ 0 \text{ if direction } s = \text{direction } s+1 \text{ for } m^{\text{th}} \text{ model} \end{cases} \qquad \text{Eq. 1}$$

Objective 2: Minimise the mean of the total tool changes

$$\overline{n}_{tc} = \frac{1}{M}\sum_{m=1}^{M}\sum_{s=1}^{n-1} t_s^m; \qquad t_s^m = \begin{cases} 1 \text{ if tool } s \neq \text{tool } s+1 \text{ for } m^{\text{th}} \text{ model} \\ 0 \text{ if tool } s = \text{tool } s+1 \text{ for } m^{\text{th}} \text{ model} \end{cases} \qquad \text{Eq. 2}$$

Objective 3: Minimise the mean of cycle times

$$\overline{ct} = \frac{1}{M}\sum_{m=1}^{M} ct_m; \qquad ct_m\text{: Cycle time for } m^{\text{th}} \text{ model} \qquad \text{Eq. 3}$$

Objective 4: Minimise number of workstations

Number of workstation (nws) is determined once the assembly tasks assignment completed. The number of workstation that generated for all models will be the same because similar tasks within different model are assigned into similar workstation.

Objective 5: Minimise the mean of workload variations

$$\overline{v} = \frac{1}{M}\sum_{m=1}^{M}\frac{\sum_{i=1}^{nws}(ct_m - pt_i^m)}{nws}; \qquad \text{Eq. 4}$$

pt_i^m: processing time in i^{th}workstation for model m

nws : total number of workstation

Subjected to:

$$\sum_{k=1}^{nws} x_{ik} = 1 \qquad i = 1, \dots, n \qquad \text{Eq. 5}$$

$$\sum_{k=1}^{nws} x_{ak} - \sum_{k=1}^{nws} x_{bk} \leq 0 \qquad a \in n, b \in F_a \qquad \text{Eq. 6}$$

$$\sum_{i=1}^{n} t_i^m x_{ik} \leq ct_m \qquad \forall m, \forall k \qquad \text{Eq. 7}$$

The first constraint (Eq. 5) ensures that an assembly task is assigned into one workstation. This constraint also means that the same assembly task from different model will be assemble in similar workstation. Eq. 6 represents the precedence constraint that needs to be followed. The F_a refers to the set of successor for task i. In different word, this constraint ensures that the successor/s for task i will be assigned in similar or the following workstation. The constraint in Eq.7 ensures that the maximum cycle time for respective model (ct_m) is obeyed. In the case of any ct_m constraint is violated, the particular assembly task cannot be assigned into that workstation.

3. Multi-objective Discrete Particle Swarm Optimisation

Various algorithms have been developed to optimise combinatorial optimisation problem. For instance, Hu et al., (2014) implemented new discrete Particle Swarm Optimisation for a combinatorial problem, involving a machining scheme selection. Besides that, researcher also introduced probability increment based swarm algorithm to optimise combinatorial optimization problem in printed circuit board assembly (Zeng et al., 2014). Another popular algorithm to optimise combinatorial optimisation problem is genetic algorithm, as implemented for scheduling and vehicle routing problems (Mirabi, 2015; Rahman, Sarker and Essam, 2017).

In this work, we implement Multi-objective Discrete Particle Swarm Optimisation (MODPSO) to optimise the integrated mixed-model ASP and ALB (Ab. Rashid, 2013). The general procedure of MODPSO is presented in Figure 2.

3.1 Initialisation

The initialisation stage start with defining the number of particle (n_{par}), the maximum iteration ($iter_{max}$), the inertia weight (c_1) and learning coefficients (c_2, c_3). In this work, the default coefficient values for PSO are used (i.e. $c_1 = 0.4$, $c_2 = c_3 = 1.4$). Next, the initial population is generated. The initial population consists of n_{par} particles. Each of position/solution contains random integer permutation, $X_i^t = x_{i,1}^t, x_{i,2}^t, \ldots x_{i,n}^t$. Since the solution is randomly generated, the solution most probably will violate the precedence constraint. Therefore, the sorting procedure based on the earliest position in position X is applied. The example of this procedure is presented in Figure 3.

[Figure 2: Flowchart of MODPSO algorithm]

3.2 Evaluation

The evaluation is conducted using five objective functions as explain in section 2.1. Since we use Pareto approach, the objective functions are calculated independently. Next, we conduct non-dominated sorting to identify the non-dominated solutions. The detail of non-dominated sorting procedure is available in Deb (Deb, 2002).

[Figure 3: Example of decoding procedure]

3.3 Update Pbest and Gbest

The Pbest represent the best solution over the iterations within the similar particle. Meanwhile the Gbest is the best solution among all the particles. In original PSO, the Pbest and Gbest are simply determined based on the fitness of solution. However, in multi-objective with Pareto based approach, we cannot determine the Pbest and Gbest using the fitness value. Therefore, we calculate the Crowding Distance to decide the Pbest and Gbest. The detail of Crowding Distance procedure is adopted from (Deb, 2002).

For Pbest, the Crowding Distance is calculated among the solution within a local particle from different iterations (CD_p). Meanwhile to determine Gbest, the Crowing Distance is measured among the non-dominated solutions (CD_{ND}). The higher Crowding Distance solution is preferable since it will lead to explore the solution in the less crowded region.

3.4 Update Position and Velocity

In PSO, the particle reproduction process is performed using two formulas:

$$V_i^{t+1} = c_1 V_i^t + c_2\left(Pbest_i^t - X_i^t\right) + c_3\left(Gbest_t - X_i^t\right) \qquad \text{Eq. 8}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \qquad \text{Eq. 9}$$

Eq. 8, calculate the velocity for $(t+1)^{th}$ iteration. This formula takes into account the current velocity and distance between Pbest and Gbest with the current position, X_i^t. Meanwhile, Eq. 9 updates the position for $(t+1)^{th}$ iteration, X_i^{t+1}. For the discrete representation, the following procedures are applied (Rameshkumar, Suresh and Mohanasundaram, 2005).

Subtraction operator (position – position): $(X_1 – X_2)$.

If the j^{th} element of X_1, $x_{1,j} = x_{2,j}$ then $v_{1,j} = 0$, else $v_{1,j} = x_{1,j}$

Multiplication operator (coefficient x velocity): $(V_2 = c.V_1)$.

If rand<c, $v_2 = v_1$, else, $v_2 = 0$

$c \in [0,1]$

Addition operator (velocity + velocity): $(V = V_1 + V_2)$

The j^{th} element of V can be derived as follows:

$$v_j = \begin{cases} v_{1,j} \text{ if } v_{1,j} \neq 0, v_{2,j} = 0 \\ v_{1,j} \text{ if } v_{1,j} \neq 0, v_{2,j} \neq 0, r < cp \\ v_{2,j} \text{ otherwise} \end{cases} \qquad \text{Eq. 10}$$

r is a random number between 0 and 1, while $cp \in [0, 1]$.

Addition operator (position + velocity): $(X_1^t + V_1)$.

If the j^{th} element of V_1, $v_{1,j} = 0$ then $x_{1,j}^{t+1} = x_{1,j}^t$, else $x_{1,j}^{t+1} = v_{1,j}$

4. Experiment Design

In previous work, a tuneable test problem generator to provide sufficient test problem for integrated ASP and ALB has been developed (Ab Rashid, Hutabarat and Tiwari, 2012). The results indicate that the ASP and ALB problem difficulties can be increased using larger number of tasks (n), lower Order Strength (OS), lower Time Variability Ratio (TV) and higher Frequency Ratio (FR). For the testing of integrated mixed-model ASP and ALB, we proceed as follows:

1. The tuneable test problem generator creates a precedence graph that is assumed as the joint model.
2. The original tuneable test problem generator creates one assembly data set that corresponds to the precedence graph. This is modified, such that three sets of assembly data, representing different product models, are generated instead.

For the purpose of this experiment, every input variable is divided into five levels from low to high difficulty values as shown in Table 1. Then a reference variable setting (datum) is selected as a baseline, while the rest of the problem variable setting are generated by changing only one variable value at a time. In total, there are 17 test problems (including the reference setting) generated from one reference variable setting. In order to confirm algorithm performance, three different reference variable setting will be used (Level 1, 3 and 5). Therefore, the complete number of test problem in this experiment is 51 problems as shown in Table 2. The bolded problem setting (Problem 1, 18 and 35) represent the reference variable setting for Level 1, 3 and 5 respectively. The detail of the test problem is accessible at the following link.

(https://drive.google.com/file/d/0B1FocUClXEMUSmFNdDN2ZFV4QTA/view?usp=sharing)

Table 1: Level of tuneable input setting

Level	n	OS	TV	FR
1	15	0.6	8	0.2
2	20	0.5	6	0.3
3	40	0.4	4	0.4
4	60	0.3	3	0.6
5	80	0.2	2	0.8

Table 2: Experimental design for mixed-model ASP and ALB

Test Problem Variable for Reference Setting at Level 1					Test Problem Variable for Reference Setting at Level 3					Test Problem Variable for Reference Setting at Level 5				
Problem	n	OS	TV	FR	Problem	n	OS	TV	FR	Problem	n	OS	TV	FR
1	**15**	**0.6**	**8**	**0.2**	**18**	**40**	**0.4**	**4**	**0.4**	**35**	**80**	**0.2**	**2**	**0.8**
2	20	0.6	8	0.2	19	15	0.4	4	0.4	36	15	0.2	2	0.8
3	40	0.6	8	0.2	20	20	0.4	4	0.4	37	20	0.2	2	0.8
4	60	0.6	8	0.2	21	60	0.4	4	0.4	38	40	0.2	2	0.8
5	80	0.6	8	0.2	22	80	0.4	4	0.4	39	60	0.2	2	0.8
6	15	0.5	8	0.2	23	40	0.6	4	0.4	40	80	0.6	2	0.8
7	15	0.4	8	0.2	24	40	0.5	4	0.4	41	80	0.5	2	0.8
8	15	0.3	8	0.2	25	40	0.3	4	0.4	42	80	0.4	2	0.8
9	15	0.2	8	0.2	26	40	0.2	4	0.4	43	80	0.3	2	0.8
10	15	0.6	6	0.2	27	40	0.4	8	0.4	44	80	0.2	8	0.8
11	15	0.6	4	0.2	28	40	0.4	6	0.4	45	80	0.2	6	0.8
12	15	0.6	3	0.2	29	40	0.4	3	0.4	46	80	0.2	4	0.8
13	15	0.6	2	0.2	30	40	0.4	2	0.4	47	80	0.2	3	0.8
14	15	0.6	8	0.3	31	40	0.4	4	0.2	48	80	0.2	2	0.2
15	15	0.6	8	0.4	32	40	0.4	4	0.3	49	80	0.2	2	0.3
16	15	0.6	8	0.6	33	40	0.4	4	0.6	50	80	0.2	2	0.4
17	15	0.6	8	0.8	34	40	0.4	4	0.8	51	80	0.2	2	0.6

In general, the integrated ASP and ALB for single model employed three types of algorithms; Evolutionary algorithms (including the hybridised version), Ant colony optimization (ACO) and Discrete PSO algorithms. This work therefore will compare the MODPSO with the following algorithms for optimization purpose:

i. Multi-Objective Genetic Algorithm (MOGA): This algorithm is one of the most frequently used algorithms to optimise independent ASP and ALB problem, according to the survey (Rashid, Hutabarat and Tiwari, 2011).

ii. Ant Colony Optimisation (ACO): The ACO algorithm has been implemented for single model integrated ASP and ALB optimisation (Yang, Lu and Zhao, 2013; Lu and Yang, 2016).

iii. Hybrid Genetic Algorithm (HGA): The HGA that proposed by Chen is the most cited published work on integrated ASP and ALB optimisation for single model (Chen, Lu and Yu, 2002). This algorithm combined the heuristic approach in line balancing with Genetic Algorithm. The output solution from the heuristic approaches will be inserted into the initial population for Genetic Algorithm.

iv. Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II): NSGA-II was introduced by (Deb, 2002). This algorithm is selected because of its popularity in solving multi-objective optimisation.

v. Multi-Objective Particle Swarm Optimisation (MOPSO): The MOPSO algorithm introduced to extend the PSO application for multi-objective optimisation (Coello Coello and Lechuga, 2002).

vi. Discrete Particle Swarm Optimisation (DPSO): DPSO present the discrete updating procedure to update position and velocity (Rameshkumar, Suresh and Mohanasundaram, 2005). The discrete representation is suitable to be used for ASP and ALB problem.

In addition to this experiment, another set of computational experiment was conducted to identify the best coefficient values for MODPSO. There are three coefficients that influence the MODPSO performance: inertia weight (c_1), cognitive coefficient, (c_2) and social coefficient (c_3). In MODPSO, c_1 coefficient influences the particle velocity, while c_2 and c_3 influence the exploring and exploiting of the search space, respectively. The limit for these coefficients is suggested as follows: c_1 [0, 1], c_2 and c_3 [0, 3]. In this study, a Taguchi approach with L9 orthogonal array is used. The three levels of coefficient values are as follows::

$c_1 = \{0.2, 0.5, 0.8\}$, $c_2 = \{0.5, 1.5, 2.5\}$ and $c_3 = \{0.5, 1.5, 2.5\}$

In this experiment, 15 test problems from Table 2 are selected, which consist of 5 problems in each reference setting. The selected problems are problems 1 – 5, 18 – 22 and 35 – 39.

In this work, the population or swarm size is set at 20 with 500 iterations. For each problem, 30 runs with different random seeds are performed and the output from each run are collected and filtered to find the non-dominated solution set.

4.1 Performance indicators

To evaluate the performance of each algorithm when dealing with different complexity problems, the following performance indicators adopted from (Deb, 2002) and (Yoosefelahi et al., 2012) are used.

i. Number of non-dominated solution in Pareto optimal, $\tilde{\eta}$: Shows the number of non-dominated solutions generated by each algorithm in the Pareto solution set. The higher $\tilde{\eta}$ indicates better algorithm performance.
ii. Error Ratio, ER: ER counts the number of solutions which are not members of the Pareto optimal set, divided by the number of solutions generated by algorithm. Smaller ER indicates better algorithm performance.
iii. Generational Distance, GD: GD calculation yields an average distance of solution with the nearest Pareto optimal solution. Smaller GD value indicates better algorithm performance.
iv. Spacing: This indicator measures the relative distances between each solution. Smaller Spacing index indicates better solution set, having better spacing between each solution.
v. Maximum Spread, $Spread_{max}$: Measures the spread of solutions found by each algorithm. Larger maximum spread is the better.

5. Results of Computational Experiment

Due to the large size data from the optimization, the results were simplified the data by using standard competition rank approach. The best algorithm for a particular indicator and test problem was assigned rank 1 while the worst was assigned as rank 7. When the algorithm performance is a tie, an equal rank will be assigned and the next rank will be left empty. Table 3 present the frequency of the rank obtained by each algorithm for different indicator and test problem. For the non-dominated solution in Pareto optimal ($\tilde{\eta}$) indicator, the MODPSO comes out with better solution sets in 96% of test problems, while the remaining 4% belong to NSGA-II. The Error Ratio (ER) indicator also shows that the leading algorithms are MODPSO and NSGA-II. The MODPSO and NSGA-II show better performance in 41.5% and 58.5% respectively. Both algorithms also dominate the best performance for Generational Distance (GD) indicator with 43% of better performance for MODPSO and 53% for NSGA-II. Meanwhile, the Spacing indicator shows different pattern, where the largest percentages of better performance are MOPSO (22), followed by HGA (20%), ACO (19%), DPSO (17%), MOGA (14%), MODPSO (6%) and NSGA-II (2%). On the other hand, the $Spread_{max}$ indicator that measure the extent of solution distribution presents that the MODPSO algorithm produce better solution in 70% of the problem. The MOPSO perform better in 18%, while the remaining balances are shared among DPSO (6%), MOGA (4%) and HGA (2%).

Table 3: Frequency of the rank obtained by each algorithm

Indicator	Rank	MOGA	ACO	HGA	NSGA-II	MOPSO	DPSO	MODPSO
	1	0	0	0	5	0	0	47
	2	0	1	8	37	1	0	4
	3	11	2	22	5	12	8	0
$\tilde{\eta}$	4	15	10	8	3	18	11	0
	5	12	5	8	1	11	14	0
	6	8	8	4	0	6	18	0
	7	5	25	1	0	3	0	0
	1	0	0	0	31	0	0	23
	2	0	0	5	17	1	0	26
	3	10	6	20	3	4	5	2
ER	4	19	8	11	0	18	7	0
	5	8	5	11	0	13	16	0
	6	10	9	4	0	11	19	0
	7	4	23	0	0	4	4	0
	1	0	0	0	30	0	0	24
	2	2	0	6	17	1	0	22
	3	17	3	23	2	2	1	3
GD	4	20	6	10	1	7	6	1
	5	8	7	8	0	17	10	1
	6	3	5	3	1	19	21	0
	7	1	30	1	0	5	13	0
	1	7	8	7	8	13	5	3
	2	10	2	4	10	10	6	9
	3	11	7	6	7	13	5	2
Spacing	4	1	3	4	5	4	14	20
	5	10	13	10	6	7	3	2
	6	9	11	13	10	2	4	2
	7	3	8	6	5	3	13	13

	1	2	1	1	0	10	4	33
	2	3	1	8	2	16	15	7
	3	5	2	14	4	11	12	2
Spread$_{max}$	4	7	3	12	7	8	12	2
	5	13	6	11	6	5	5	5
	6	13	15	3	16	1	2	1
	7	8	23	2	16	0	1	1

Table 4 presents the mean of performance indicators for all test problems. Based on the mean values, the best performance of $\tilde{\eta}$ indicator is observed in MODPSO and the followed by NSGA-II algorithms. Meanwhile, the best mean performance for ER and GD indicators is achieved by NSGA-II, while the MODPSO in second place. In the meantime, two PSO-based algorithms, MOPSO and DPSO are leading the mean of Spacing indicator. Furthermore, the PSO-based algorithms also show better performance compared with other algorithms in Spread$_{max}$ indicator.

Table 4: Mean of performance indicators

Indicator	Algorithm						
	MOGA	ACO	HGA	NSGA-II	MOPSO	DPSO	MODPSO
$\tilde{\eta}$[1]	4.7843	1.9020	8.0588	27.2353	5.2745	3.4902	41.0196
ER[2]	0.9037	0.9632	0.8592	0.1952	0.9230	0.9444	0.2046
GD[2]	1.9951	2.4650	2.0017	0.1753	2.3219	2.3682	0.2696
Spacing[2]	1.0281	1.1410	0.9819	1.2898	0.9479	0.9537	1.2318
Spread$_{max}$[1]	15.7278	14.6364	16.5250	14.9729	17.1868	16.8720	18.4656

[1] Larger the better indicator [2] Smaller the better indicator

Table 5: Average CPU time for different problem size

Problem Size	Average CPU Time (s)						
	MOGA	ACO	HGA	NSGA-II	MOPSO	DPSO	MODPSO
15	42.34	35.86	45.14	93.55	43.97	49.83	51.34
20	76.58	53.87	80.04	165.08	35.61	86.72	88.09
40	376.34	225.76	376.88	753.36	202.48	388.28	401.82
60	1067.18	672.30	1057.98	2230.19	629.33	1077.80	1101.58
80	2295.72	1694.14	2394.84	4902.50	1611.51	2386.95	2419.80

Table 5 shows the average CPU time for different problem size. In general, the ACO and MOPSO were among the fastest algorithm to complete the iteration. Meanwhile, the MODPSO was roughly in the second last position, in front of NSGA-II in term of CPU time. For comparison, the MODPSO was just 2 – 3% behind the DPSO. In DPSO and MODPSO, a longer time is taken to conduct discrete updating procedures compared with regular updating procedures in MOPSO. However, the NSGA-II required mostly double CPU time compared with MODPSO. This is because the NSGA-II combined the existing population and new offspring for the non-dominated sorting procedure. Therefore, the time taken to complete the iteration was increased compared with other algorithms.

5.1 MODPSO Coefficient Tuning

Table 6 shows the results of the MODPSO coefficient experiment. The experimental table was designed using Taguchi L9 orthogonal array. Based on the general observation, experiment number 4 led in term of the best solution of cardinality, which was represented by $\tilde{\eta}$ and ER. The same experiment also came out with the best accuracy (i.e. GD indicator).

Table 6: MODPSO coefficient experiment results

Experiment Number	Coefficients			Mean of performance indicators				
	c_1	c_2	c_3	$\tilde{\eta}$	ER	GD	Spacing	$Spread_{max}$
1	0.2	0.5	0.5	34.1264	0.2178	0.3102	1.4178	18.2750
2	0.2	1.5	1.5	38.7028	0.1786	0.2156	1.3328	17.2138
3	0.2	2.5	2.5	39.9842	0.1755	0.1881	1.3925	16.0811
4	0.5	0.5	1.5	45.8421	0.1141	0.1070	1.5397	18.4845
5	0.5	1.5	2.5	41.8148	0.1712	0.2046	1.3979	17.9512
6	0.5	2.5	0.5	35.5908	0.1991	0.2662	1.5330	16.5959
7	0.8	0.5	2.5	40.3503	0.1763	0.1784	1.2381	16.6047
8	0.8	1.5	0.5	35.7739	0.2115	0.2770	1.4051	18.4490
9	0.8	2.5	1.5	37.0553	0.1961	0.2683	1.2402	18.2447

Meanwhile, Figure 4 presents the main effect plots of c_1, c_2 and c_3 for different performance indicators. Based on the main effect plots, medium c_1, low c_2 and medium c_3 coefficients were preferable as observed in $\tilde{\eta}$ and ER plots to produce a solution with good cardinality. Similar coefficients' levels were also required to generate accurate solutions as represented by the GD indicator. On the other hand, the main effect plots by Spacing and $Spread_{max}$ indicated that high c_1, medium c_2 and medium c_3 coefficients' respective levels contributed to better solution distribution.

[Figure 4: Effect of w, c_1 and c_2 on Performance Indicators: (a) $\tilde{\eta}$, (b) ER, (c) GD, (d) Spacing, and (e) $Spread_{max}$]

6. Discussion of Results

In general, the result from experiment shows that the performance of algorithms in optimising integrated mixed-model ASP and ALB appear to be dominated by NSGA-II and proposed MODPSO algorithms, especially in four performance indicators (i.e. $\tilde{\eta}$, ER, GD and $Spread_{max}$). However further analyses are required to quantify the results. Therefore, a statistical test is conducted to measure the significance of the improvements achieved by the MODPSO in optimising integrated mixed-model ASP and ALB.

The Analysis of Variance (ANOVA) test was then carried out to evaluate any significant improvement between the results obtained by different algorithms. The 'null hypothesis' stated that there is no significant improvement among the means of all algorithm results. The alternative hypothesis state that there is significant improvement among means in the result of at least one algorithm. The null hypothesis will be accepted when the calculated f-value is smaller than critical f-value (f*) as suggested in the f-distribution table (Coolidge, 2000). The result of ANOVA test is presented in Table 7.

Table 7: Summary of ANOVA test

	$\tilde{\eta}$	ER	GD	Spacing	$Spread_{max}$
f*	3.69	3.69	3.69	3.69	3.69
f	186.081	262.1808	45.8928	12.8327	17.4301

f*: critical f-value   f: calculated f-value

The result shows that the calculated f-value for all performance indicators are consistently larger than f* at 0.05 confidence interval. Therefore, the null hypothesis is rejected and the alternative is accepted for all indicators, which bring the meaning that there are significant improvements achieved for all indicators in at least one algorithm. However, the ANOVA test cannot differentiate the exact improvement of one algorithm in comparison with another algorithm.

Therefore a posteriori test known as Tukey's Honestly Significant Difference (HSD) is performed. This test is performed by calculating the absolute mean difference between the results of one algorithm over another algorithm, which is then compared with the critical HSD (HSD*) value. The HSD* value for algorithm i is calculated as follows.

$$\mathrm{HSD}_i^* = q.\sqrt{\frac{\mathrm{MSW}_i}{n}} \qquad \text{Eq. 11}$$

The q value is acquired from Tukey's table, MSW is the mean squares within groups from ANOVA test, and n is the number of data in each group. When the absolute mean difference is larger than HSD*, a significant improvement has been identified in one algorithm over another algorithm. At this point, we are interested to know the performance of MODPSO over the other algorithms. Table 8 presents the HSD* and absolute mean difference between MODPSO and the other algorithms.

Table 8: Summary of Tukey's HSD test for MODPSO algorithm

		Absolute Mean Difference Between MODPSO and Comparison Algorithm				
Indicator (HSD*)		$\tilde{\eta}$ (4.5902)	ER (0.0906)	GD (0.6131)	Spacing (0.1621)	Spread_{max} (1.3337)
Comparison Algorithm	MOGA	**36.2353**[1]	**0.6991**[1]	**1.7255**[1]	0.2037[2]	**2.7379**[1]
	ACO	**39.1176**[1]	**0.7586**[1]	**2.1954**[1]	0.0908[2]	**3.8292**[1]
	HGA	**32.9608**[1]	**0.6547**[1]	**1.7321**[1]	0.2499[2]	**1.9406**[1]
	NSGA-II	**13.7843**[1]	0.0094[2]	0.0944[2]	0.0580[1]	**3.4928**[1]
	MOPSO	**35.7451**[1]	**0.7184**[1]	**2.0523**[1]	0.2839[2]	1.2789[1]
	DPSO	**37.5294**[1]	**0.7399**[1]	**2.0985**[1]	0.2781[2]	**1.5936**[1]

[1]Better absolute mean difference for MODPSO
[2]Better absolute mean difference for comparison algorithm

In Table 8, the values that are labelled '1' show the MODPSO has a better mean difference over the comparison algorithm, while the values labelled '2' mean that the comparison algorithm has a better mean difference over MODPSO. On the other hand, the bold values in Table 8 indicate the significant improvements achieved by MODPSO over other algorithms, since the absolute mean difference is larger than HSD*. Based on Table 8, the MODPSO algorithm shows better performance and significant improvement when compared with the set of algorithms for $\tilde{\eta}$ indicator. The MODPSO also show significant improvements for ER and GD indicators compared with other algorithms, with the exception of NSGA-II. In both indicators, the NSGA-II algorithm shows better mean difference compare with MODPSO, however, the difference is not significant because the absolute mean difference are smaller than HSD*.

Meanwhile, the Spacing indicator did not show any significant improvement of MODPSO although it has a better mean difference when compared with NSGA-II. Except for NSGA-II, all other algorithms show better performance over MODPSO, where significant improvements are presented by four algorithms (MOGA, HGA, MOPSO and DPSO). For $Spread_{max}$ indicator, the MODPSO algorithm shows significant improvement compared with other algorithms, except MOPSO. In comparison with MOPSO, although no significant improvement is achieved, the MODPSO algorithm still produces better solution.

In this work, the solution quality towards Pareto optimal are measured using three performance indicators i.e. $\tilde{\eta}$, ER and GD. The Spacing indicator measures the uniformity of the found solutions and $Spread_{max}$ measures the ability of algorithm to explore the extreme solutions within the solution space. The results from statistical test explain that, the MODPSO algorithm shows significant improvement in term of finding better solution towards Pareto optimal over comparison algorithms, with the exception of NSGA-II at 0.05 confidence intervals.

Furthermore, the $Spread_{max}$ result means that the MODPSO algorithm is significantly able to explore better extreme solutions when compared with MOGA, ACO, HGA, DPSO and NSGA-II. Meanwhile, in term of uniformity of solution spread, the MODPSO algorithm did not perform significantly better than other algorithms. The Spacing indicator considers all non-dominated solutions that was found by a particular algorithm, regardless of Pareto or non-Pareto optimal solutions. In general, for similar search space, the algorithm that generated more non-dominated solutions has greater chances to produce better Spacing. From the experiment, the mean number of non-dominated solutions generated by the algorithms (regardless of Pareto or non-Pareto optimal), in ascending order, are: NSGA-II (33.84), ACO (46.9), MODPSO (55.37), MOGA (56.14), HGA (68.91), DPSO (80.47) and MOPSO (85.02). These numbers clearly present the algorithms that show significant improvement over MODSPO for Spacing indicator are the algorithms with larger mean of generated solutions.

The result from experiments and statistical tests summarise that the MODPSO has shown significant improvement over the majority of compared algorithms in $\tilde{\eta}$, ER, GD and Spread$_{max}$ indicators. In comparison with all other algorithms, the performance of MODPSO is closely followed by NSGA-II, where the MODPSO only show significant improvement over NSGA-II in $\tilde{\eta}$ and Spread$_{max}$ indicators. In order to compare performance of NSGA-II, the mean difference between NSGA-II and other algorithms are calculated and presented in Table 9.

Table 9: Summary of Tukey's HSD test for NSGA-II

		Absolute Mean Difference Between NSGA-II and Comparison Algorithm				
Indicator (HSD*)		$\tilde{\eta}$ (4.5902)	ER (0.0906)	GD (0.6131)	Spacing (0.1621)	Spread$_{max}$ (1.3337)
Comparison Algorithm	MOGA	**22.4510[1]**	**0.7085[1]**	**1.8199[1]**	0.2618[2]	0.7549[2]
	ACO	**25.3333[1]**	**0.7680[1]**	**2.2897[1]**	0.1489[2]	0.3365[1]
	HGA	**19.1765[1]**	**0.6640[1]**	**1.8264[1]**	0.3079[2]	1.5522[2]
	MOPSO	**21.9608[1]**	**0.7278[1]**	**2.1466[1]**	0.3419[2]	2.2139[2]
	DPSO	**23.7451[1]**	**0.7492[1]**	**2.1929[1]**	0.3361[2]	1.8991[2]
	MODPSO	13.7843[2]	0.0094[1]	0.0944[1]	0.0580[2]	3.4928[2]

[1]Better absolute mean difference for NSGA-II
[2]Better absolute mean difference for comparison algorithm

Table 9 indicates that the NSGA-II has significant improvement for solution quality leading to Pareto optimal compared with other algorithms except the MODPSO. Besides that, the NSGA-II did not show any significant improvement for solution uniformity (Spacing) and extreme solution exploration (Spread$_{max}$). Based on the significant improvement achieved by MODPSO (Table 8) and NSGA-II

(Table 9) over other algorithms, the MODPSO is found to perform better than NSGA-II. This is because the MODPSO have shown significant improvement over NSGA-II in two of indicators (i.e. $\tilde{\eta}$ and Spread_{max}), while there is no significant improvement of NSGA-II over MODPSO algorithm. Furthermore, for Spread_{max} indicator, the NSGA-II did not show any significant improvement as MODPSO shows when compared with all other algorithms. In addition, the NSGA-II required double CPU time to complete the iteration compared with MODPSO as presented in Table 5. These facts give more advantages to MODPSO in term of solution quality and also algorithm effort.

The result from Tukey's HSD test for integrated mixed-model ASP and ALB clearly shows that the MODPSO performed better than other algorithms for all test problems. Another question that arises is the problem category that the MODPSO algorithm performed best and worst. Therefore, the Tukey's HSD test based on different problem reference setting is conducted. The result of Tukey's HSD test for different problem setting is presented in Table 10. Based on Table 10, the MODPSO shows significant improvement in $\tilde{\eta}$ indicator over all algorithms for all reference setting. For ER indicator, the MODPSO consistently demonstrates significant improvement over other algorithms except NSGA-II. Meanwhile for GD indicator in low level reference setting (Level 1), significant improvements for MODPSO are only found over ACO, MOPSO and DPSO algorithms. However, when the reference setting is changed to medium (Level 3) and high (Level 5) levels, significant improvements are also observed in comparison with MOGA and NSGA-II.

On the other hand, the MODPSO consistently did not show any significant improvement over any algorithm for Spacing indicator. For Spread_{max} indicator, the proposed algorithm also did not show significant improvements in low level reference setting. However, when the reference setting is moved to medium level, the MODPSO shows significant improvement over MOGA, ACO and NSGA-II. Finally, in the problem with high level reference setting, significant improvements are achieved by MODPSO over all other algorithms. From this result, the best performance of MODPSO is found in the problem with high reference setting. Meanwhile, the weakest performance is in the problem with low level reference setting, even though the overall performance in this problem category is still better than other algorithms.

Table 10: Summary of Tukey's HSD test for MODPSO by reference setting level

Reference Setting	Algorithm	Absolute Mean Difference Between MODPSO and Algorithm				
		$\tilde{\eta}$	ER	GD	Spacing	Spread_{max}
Level 1	HSD*	9.3491	0.1684	0.9264	0.3109	2.3693
	MOGA	**32.3529**[1]	**0.5009**[1]	0.7618[1]	0.0116[2]	0.8080[1]
	ACO	**37.8235**[1]	**0.6412**[1]	**1.2778**[1]	0.1152[1]	2.1079[1]
	HGA	**25.8824**[1]	**0.4142**[1]	0.6812[1]	0.0322[2]	0.8376[1]
	NSGA	**20.8235**[1]	0.0966[1]	0.0903[1]	0.1924[1]	2.2812[1]
	MOPSO	**31.9412**[1]	**0.5373**[1]	**0.9917**[1]	0.0924[2]	0.1576[2]
	DPSO	**35.7059**[1]	**0.5927**[1]	**1.1062**[1]	0.0839[2]	0.0122[1]
Level 3	HSD*	7.2889	0.1436	0.8001	0.2325	1.9374
	MOGA	**40.5882**[1]	**0.7850**[1]	**2.0228**[1]	0.2749[2]	**2.7749**[1]
	ACO	**43.2941**[1]	**0.8292**[1]	**2.5720**[1]	0.2616[2]	**3.7624**[1]
	HGA	**38.8824**[1]	**0.7747**[1]	**2.0196**[1]	0.3462[2]	1.5431[1]
	NSGA	**13.0588**[1]	0.0448[2]	0.2236[2]	0.0036[1]	**3.3176**[1]
	MOPSO	**40.0000**[1]	**0.7975**[1]	**2.3401**[1]	0.3913[2]	0.8924[1]
	DPSO	**41.7059**[1]	**0.8142**[1]	**2.3461**[1]	0.3576[2]	1.2606[1]
Level 5	HSD*	5.4125	0.1002	0.9376	0.2957	2.6973
	MOGA	**35.7647**[1]	**0.8115**[1]	**2.3920**[1]	0.3247[2]	**4.6306**[1]
	ACO	**36.2353**[1]	**0.8054**[1]	**2.7364**[1]	0.1261[2]	**5.6173**[1]
	HGA	**34.1176**[1]	**0.7751**[1]	**2.4955**[1]	0.3712[2]	**3.4412**[1]
	NSGA	**7.4706**[1]	0.0799[2]	0.1497[2]	0.0219[2]	**4.8795**[1]
	MOPSO	**35.2941**[1]	**0.8204**[1]	**2.8249**[1]	0.3679[2]	**3.1018**[1]
	DPSO	**35.1765**[1]	**0.8127**[1]	**2.8433**[1]	0.3927[2]	**3.5081**[1]

[1]Better absolute mean difference for MODPSO
[2]Better absolute mean difference for comparison algorithm

The performance of MODPSO in optimising integrated mixed-model ASP and ALB because this algorithm was specifically developed for discrete multi-objective optimisation problem. This algorithm use similar procedure for Initialisation, Evaluation and Selection strategies as in NSGA-II. The NSGA-II is another algorithm that specifically developed for multi-objective optimisation problems that also performed well in this application. However, it does not have the fine tuning feature. The fine tuning feature means ability of algorithm to make small adjustments to solution in order to achieve the best or a desired performance. This is an important feature for ASP and ALB, where small changes may lead to sudden improvement in results.

The discrete updating procedure in MODPSO is designed to enable fine tuning towards the end of iterations. According to discrete updating procedure (Subtraction operator (X_i-X_j)) in Section 3.4, zero velocity is given when similar element in X_i and X_j is found (this is the case when all particles move towards the best solution at the end of iterations). When majority of velocity elements are zero, only small changes occur in assembly sequence as presented by Addition operator (X_i+V_i) in Section 3.4. This feature allows fine tuning of the assembly sequences in MODPSO.

7. Conclusions

This paper formulates and studies the optimisation of integrated mixed-model Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) problem using Multi-objective Discrete Particle Swarm Optimisation (MODPSO). A set of test problems with different range of difficulties has been used to test the performance of MODPSO in optimising integrated mixed-model ASP and ALB. In addition, MODPSO coefficient tuning has been conducted to identify the best settings for optimisation.

The experimental results indicate that, in general, the MODPSO algorithms performed better than other comparison algorithms. Statistical test concluded that the MODPSO has shown significant improvement in converging to Pareto optimal solution and exploring the extreme solutions in search space. The statistical test also concluded that the MODPSO performed best in the problem with high level of difficulty. Meanwhile the weakest performance is in the problem at low difficulty level, although it still performed better than comparison algorithms. The MODPSO coefficient tuning suggested that the optimum performance for solution cardinality and accuracy was achieved when the inertia weight and social coefficient were at the medium level, while cognitive coefficient was at the low level.

The work in this paper has initiates the study on integrated mixed-model ASP and ALB optimisation. At the same time, it also indicates that the MODPSO algorithm is able to optimise this problem better than comparison algorithms. One of the MODPSO's downside is incapability of generating uniformly spaced solutions as presented by Spacing indicator. In future, an effort to improve the algorithm performance, especially in solution uniformity is proposed to improve overall solution quality. The first suggestions to improve solution uniformity is to consider the historical data in the crowding distance. This will make the unselected solutions because of mating pool capacity, will be taken into account when calculating the crowding distance. Besides that, the solution quality also could be improved by including the extreme solutions as a part of MODPSO updating procedure. It will influence the MODPSO convergence direction towards the extreme solution, besides the Gbest. Therefore, the search direction become more diverse.

Funding: This work was supported by Universiti Malaysia Pahang under grant RDU180331.

Conflict of Interest: The authors declare that they have no conflict of interest.

References

Ab. Rashid, M. F. F. (2013) Integrated Multi-Objective Optimisation of Assembly Sequence Planning and Assembly Line Balancing using Particle Swarm Optimisation, PhD Thesis, Cranfield University.

Ab. Rashid, M. F. F., Tiwari, A. and Hutabarat, W. (2017) 'Comparison of Sequential and Integrated Optimisation Approaches for ASP and ALB', Procedia CIRP. The Author(s), 63, pp. 505–510. doi: 10.1016/j.procir.2017.03.083.

Ab Rashid, M. F. F., Hutabarat, W. and Tiwari, A. (2012) 'Development of a tuneable test problem generator for assembly sequence planning and assembly line balancing', Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 226(11), pp. 1900–1913. doi: 10.1177/0954405412457621.

Becker, C. and Scholl, A. (2006) 'A survey on problems and methods in generalized assembly line balancing', European Journal of Operational Research, 168(3), pp. 694–715. doi: 10.1016/j.ejor.2004.07.023.

Buyukozkan, K. et al. (2016) 'Lexicographic bottleneck mixed-model assembly line balancing problem: Artificial bee colony and tabu search approaches with optimised parameters', Expert Systems with Applications, 50, pp. 151–166. doi: http://dx.doi.org/10.1016/j.eswa.2015.12.018.

Chen, R., Lu, K. and Yu, S. (2002) 'A hybrid genetic algorithm approach on multi-objective of assembly planning problem', Engineering Applications of Artificial Intelligence, 15(2002), pp. 447–457.

Coello Coello, C. A. and Lechuga, M. S. (2002) 'MOPSO: a proposal for multiple objective particle swarm optimization', in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02* (Cat. No.02TH8600). IEEE, pp. 1051–1056. doi: 10.1109/CEC.2002.1004388.

Coolidge, F. L. (2000) Statistics: A Gentle Introduction. London: SAGE Publication.

Deb, K. (2002) Multi-Objective Optimization using Evolutionary Algorithms. Sussex, United Kingdom: John Wiley & Sons.

Hu, S. J. et al. (2008) 'Product variety and manufacturing complexity in assembly systems and supply chains', {CIRP} Annals - Manufacturing Technology, 57(1), pp. 45–48. doi: http://dx.doi.org/10.1016/j.cirp.2008.03.138.

Hu, Y.-J. et al. (2014) 'Machining scheme selection based on a new discrete particle swarm optimization and analytic hierarchy process', Artificial Intelligence for Engineering Design, Analysis and Manufacturing. Cambridge University Press, 28(1), pp. 71–82. doi: 10.1017/S0890060413000504.

Jusop, M. and Ab Rashid, M. F. F. (2015) 'A review on simple assembly line balancing type-e problem', IOP Conference Series: Materials Science and Engineering, 100, p. 12005. doi: 10.1088/1757-899X/100/1/012005.

Kara, Y. et al. (2011) 'Multi-objective approaches to balance mixed-model assembly lines for model mixes having precedence conflicts and duplicable common tasks', The International Journal of Advanced Manufacturing Technology, 52(5), pp. 725–737. doi: 10.1007/s00170-010-2779-z.

Lin, M.-C. et al. (2012) 'A modularized contact-rule reasoning approach to the assembly sequence generation for product design', Concurrent Engineering, 20(3), pp. 203–221. doi: 10.1177/1063293X12452419.

Lu, C. and Yang, Z. (2016) 'Integrated assembly sequence planning and assembly line balancing with ant colony optimization approach', The International Journal of Advanced Manufacturing Technology. Springer London, 83(1), pp. 243–256. doi: 10.1007/s00170-015-7547-7.

Marian, R. M. (2003) Optimisation of Assembly Sequences using Genetic Algorithm. University of

South Australia.

Mirabi, M. (2015) ‘A novel hybrid genetic algorithm for the multidepot periodic vehicle routing problem’, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 29(1), pp. 45–54. doi: 10.1017/S0890060414000328.

Penciuc, D. et al. (2016) ‘Product life cycle management approach for integration of engineering design and life cycle engineering’, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 30(4), pp. 379–389. doi: 10.1017/S0890060416000366.

Rahman, H. F., Sarker, R. and Essam, D. (2017) ‘A genetic algorithm for permutation flowshop scheduling under practical make-to-order production system’, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 31(1), pp. 87–103. doi: 10.1017/S0890060416000196.

Rameshkumar, K., Suresh, R. K. and Mohanasundaram, K. M. (2005) ‘Discrete Particle Swarm Optimization (DPSO) Algorithm for Permutation Flowshop Scheduling to Minimize Makespan’, Advances in Natural Computation, 3612, pp. 572–581.

Rashid, M. F. F., Hutabarat, W. and Tiwari, A. (2011) ‘A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches’, The International Journal of Advanced Manufacturing Technology, 59(1–4), pp. 335–349. doi: 10.1007/s00170-011-3499-8.

Roshani, A. and Nezami, F. G. (2017) ‘Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach’, Assembly Automation, 37(1), pp. 34–50. doi: 10.1108/AA-02-2016-016.

Shankar, P., Summers, J. D. and Phelan, K. (2017) ‘A verification and validation planning method to address change propagation effects in engineering design and manufacturing’, Concurrent Engineering, 25(2), pp. 151–162. doi: 10.1177/1063293X16671771.

Tasan, S. O. and Tunali, S. (2008) ‘A review of the current applications of genetic algorithms in assembly line balancing’, Journal of Intelligent Manufacturing, 19(1), pp. 49–69. doi: 10.1007/s10845-007-0045-5.

Tseng, H.-E. et al. (2008) ‘Hybrid evolutionary multi-objective algorithms for integrating assembly sequence planning and assembly line balancing’, International Journal of Production Research, 46(21), pp. 5951–5977. doi: 10.1080/00207540701362564.

Tseng, H.-E. and Tang, C.-E. (2006) ‘A sequential consideration for assembly sequence planning and assembly line balancing using the connector concept’, International Journal of Production Research, 44(1), pp. 97–116. doi: 10.1080/00207540500250606.

Yang, Z., Lu, C. and Zhao, H. W. (2013) ‘An Ant Colony Algorithm for Integrating Assembly Sequence Planning and Assembly Line Balancing’, Applied Mechanics and Materials, 397–400, pp. 2570–2573. doi: 10.4028/www.scientific.net/AMM.397-400.2570.

Yoosefelahi, A. et al. (2012) ‘Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model’, Journal of Manufacturing Systems, 31(2), pp. 139–151. doi: 10.1016/j.jmsy.2011.10.002.

Zeng, K. et al. (2014) ‘Probability increment based swarm optimization for combinatorial optimization with application to printed circuit board assembly’, Artificial Intelligence for Engineering Design, Analysis and Manufacturing. Cambridge University Press, 28(4), pp. 429–437. doi: 10.1017/S0890060413000632.

Zhong, Y. (2017) ‘Hull mixed-model assembly line balancing using a multi-objective genetic algorithm simulated annealing optimization approach’, Concurrent Engineering, 25(1), pp. 30–40. doi: 10.1177/1063293X16666204.

Zhu, X. et al. (2012) ‘A complexity model for sequence planning in mixed-model assembly lines’, Journal of Manufacturing Systems, 31(2), pp. 121–130. doi: http://dx.doi.org/10.1016/j.jmsy.2011.07.006.

Author Biographies

Mohd Fadzil Faisae Ab. Rashid received his PhD from Cranfield University, United Kingdom in 2013. During the early of his career, he works in a multi-national company as a Production Engineer. Currently, he is an Associate Professor at the Faculty of Mechanical and Manufacturing Engineering, Universiti Malaysia Pahang. He is also a Chartered Engineer under the Institution of Mechanical Engineers. His research interests are in engineering optimization, particularly focus on manufacturing system, metaheuristics and discrete event simulation techniques.

Ashutosh Tiwari is Professor of Manufacturing Infomatics and attained his PhD degree from Cranfield University in Evolutionary Computing Techniques for Handling Variable Interaction in Engineering Design Optimisation. His PhD research was carried out as part of an EPSRC project and was successfully completed in 2001 within the two-year duration of the project. He was awarded the Academic Excellence Award for his achievements in the course. He is a Fellow of the Higher Education Academy and was awarded the PGCert in Learning, Teaching and Assessment in Higher Education in 2006.

Windo Hutabarat is a Research Fellow at the Cranfield's Manufacturing Informatics Centre, which he joined in 2008 to work on Product-Service Systems research project funded by the EPSRC. At Cranfield he pioneered the concept of applying innovations from the gaming sector into manufacturing and services. He is also active in the area of engineering optimisation and currently involved in an AMSCI funded project with Cosworth and Flexeye aimed at optimising a flexible manufacturing system.