Artificial Intelligence for Engineering Design, Analysis and Manufacturing

cambridge.org/aie

Integrated optimization of mixed-model assembly sequence planning and line balancing using Multi-objective Discrete Particle Swarm Optimization

Mohd Fadzil Faisae Ab Rashid[1], Ashutosh Tiwari[2] and Windo Hutabarat[2]

[1]Faculty of Mechanical Engineering, Universiti Malaysia Pahang, 26600 Pekan, Malaysia and [2]Manufacturing and Materials Department, Cranfield University, Bedford MK43 0AL, UK

Research Article

Cite this article: Ab Rashid MFF, Tiwari A, Hutabarat W (2019). Integrated optimization of mixed-model assembly sequence planning and line balancing using Multi-objective Discrete Particle Swarm Optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **33**, 332–345. https://doi.org/10.1017/S0890060419000131

Received: 20 June 2018
Revised: 19 March 2019
Accepted: 27 March 2019
First published online: 6 May 2019

Key words:
Assembly sequence planning; concurrent optimization; line balancing; manufacturing systems; Particle Swarm Optimization

Author for correspondence:
Mohd Fadzil Faisae Ab Rashid,
E-mail: ffaisae@ump.edu.my

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 the simple assembly problem that only runs one homogenous product. This paper therefore models and optimizes the integrated mixed-model ASP and ALB using Multi-objective Discrete Particle Swarm Optimization (MODPSO) concurrently. This is a new variant of the integrated assembly problem. The integrated mixed-model ASP and ALB is modeled using task-based joint precedence graph. In order to test the performance of MODPSO to optimize the 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 optimize the problem. The results from this experiment indicated that the MODPSO algorithm presents a significant improvement in term of solution quality toward Pareto optimal and demonstrates the ability to explore the extreme solutions in the mixed-model assembly optimization 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 optimize the integrated ASP and ALB research for mixed-model assembly problem.

Introduction

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

Assembly line problems are categorized into simple (SALBP) and generalized ALB problem (GALBP) (Becker and Scholl, 2006). The SALBP only considers the production of one homogeneous product on serial line layout, while the 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 optimization of integrated ASP and ALB problem focusing on SALBP. Chen *et al.* proposed a hybrid Genetic Algorithm (GA) to optimize the integrated ASP and ALB, where GA is combined with heuristic search (Chen *et al.*, 2002). Tseng and Tang studied combining ASP together with ALB based on assembly "connectors" (i.e. the connector basis) by using GA. 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 the recent work of integrated ASP and ALB optimization, GA-based algorithms performed well in optimizing 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 a large number of tasks (Ab Rashid *et al.*, 2012). Besides that, the researcher also implemented Ant Colony

© Cambridge University Press 2019

CAMBRIDGE UNIVERSITY PRESS

Optimization (ACO) for integrated ASP and ALB (Lu and Yang, 2016). However, it was tested with only small task numbers.

There has been, thus far, no work on integrated ASP and ALB optimization beyond SALBP type. This work therefore aims to initiate the optimization 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 the industry because of the significant cost savings made possible by sharing different models 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 in the existing assembly line will incur a lot of cost to the manufacturer (Shankar *et al*., 2017). Therefore, by integrating the ASP and ALB optimization for mixed-model assembly, the benefits from integrated optimization and mixed-model assembly can be obtained.

The integrated mixed-model ASP and ALB problem is more challenging compared with the mixed-model ALB and integrated simple ASP and ALB. Separate ASP and ALB problems are individually categorized as NP-hard combinatorial problems, where the solution space is excessively increased when the number of tasks increased (Lin *et al*., 2012). When the optimization 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 the simple assembly problem, it is more difficult to achieve an 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 optimize, when compared with the optimization 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 the Multi-objective Discrete Particle Swarm Optimization (MODPSO) algorithm to optimize this problem. Section "Integrated mixed-model ASP and ALB" presents the modeling of the integrated mixed-model ASP and ALB, including the objective functions for this problem. Section "Multi-objective Discrete Particle Swarm Optimization" explains the mechanism of MODPSO algorithm. Section "Experiment design" presents the experimental design and performance indicators for optimization algorithms. Section "Results of computational experiment" presents the results of the experiment and section "Discussion of results" discusses these results that analyze various algorithms to optimize the integrated mixed-model ASP and ALB problems. Finally, section "Conclusions" concludes the findings from this work.

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 fulfill 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 toward node j meaning the assembly task i must be completed before starting the assembly task j. Meanwhile the assembly data table represents the assembly information such as assembly direction, tool, and time for the 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 ALB 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 $V = U_y\ V_y$ and $C = U_y\ C_y$. 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 models of the 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 followers for specific tasks in each model are bundled together in one graph. For example, in Figure 1, the followers for task 1 in Model A are tasks 2 and 3, while tasks 3 and 4 in Model B. The combination of task 1 followers from both models is tasks 2, 3, and 4 as shown in the joint graph. The joint graph is updated by removing the shortest repetitive routes from the graph. In the example below, the route connecting tasks 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.

In the mixed-model assembly line, the assembly data set should represent the 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.

Objective functions and constraints

There are known objective functions to evaluate single-model assembly problems. To evaluate the fitness in the 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 the mixed-model assembly problem with M model:

Objective 1: Minimize the mean of the total direction changes

$$\bar{n}_{\mathrm{dc}} = \frac{1}{M}\sum_{m=1}^{M}\sum_{s=1}^{n-1} \mathrm{d}_s^m; \qquad \mathrm{d}_s^m = \begin{cases} 1 \text{ if } \text{ direction } s \neq \text{ direction } s+1 \text{ for } m^{\text{th}} \text{ model} \\ 0 \text{ if } \text{ direction } s = \text{ direction } s+1 \text{ for } m^{\text{th}} \text{ model} \end{cases}. \quad (1)$$

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

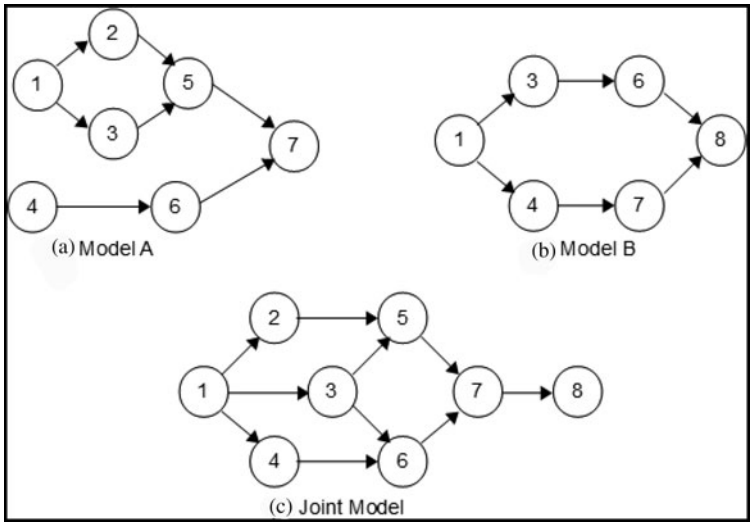


Fig. 1. Precedence graph of (a) Model A, (b) Model B, and (c) Joint Model.

Objective 2: Minimize the mean of the total tool changes

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

Objective 3: Minimize the mean of cycle times

$$\overline{\text{ct}} = \frac{1}{M}\sum_{m=1}^{M} \text{ct}_m; \quad (3)$$
$$\text{ct}_m : \text{ Cycle time for } m^{\text{th}} \text{ model.}$$

Objective 4: Minimize the number of workstations

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

Objective 5: Minimize the mean of workload variations

$$\bar{v} = \frac{1}{M}\sum_{m=1}^{M}\frac{\sum_{i=1}^{\text{nws}}(\text{ct}_m - \text{pt}_i^m)}{\text{nws}}; \quad (4)$$
$$\text{pt}_i^m : \text{ processing time in } i^{\text{th}}\text{workstation for model } m$$
$$\text{nws} : \text{total number of workstation.}$$

Subjected to:

$$\sum_{k=1}^{\text{nws}} x_{ik} = 1 \qquad \text{i} = 1, \ldots, n \quad (5)$$

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

$$\sum_{\text{i}=1}^{n} t_i^m x_{ik} \leq \text{ct}_{\text{m}} \qquad \forall m, \forall k. \quad (7)$$

The first constraint [Eq. (5)] ensures that an assembly task is assigned to one workstation. This constraint also means that the same assembly task from different models will be assembled in a similar workstation. Equation (6) represents the precedence constraint that needs to be followed. The F_a refers to the set of the successor for task i. In a different word, this constraint ensures that the successor/s for task i will be assigned in a similar or the following workstation. The constraint in Eq. (7) ensures that the maximum cycle time for a 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.

Multi-objective Discrete Particle Swarm Optimization

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

In this work, we implement MODPSO to optimize the integrated mixed-model ASP and ALB (Ab. Rashid, 2013). The general procedure of MODPSO is presented in Figure 2.

Initialization

The initialization stage starts with defining the number of particles (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 the position/solution contains random integer permutation, $X_i^t = x_{i,1}^t,\ x_{i,2}^t,\ \ldots\ x_{i,\text{n}}^t$. Since the

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

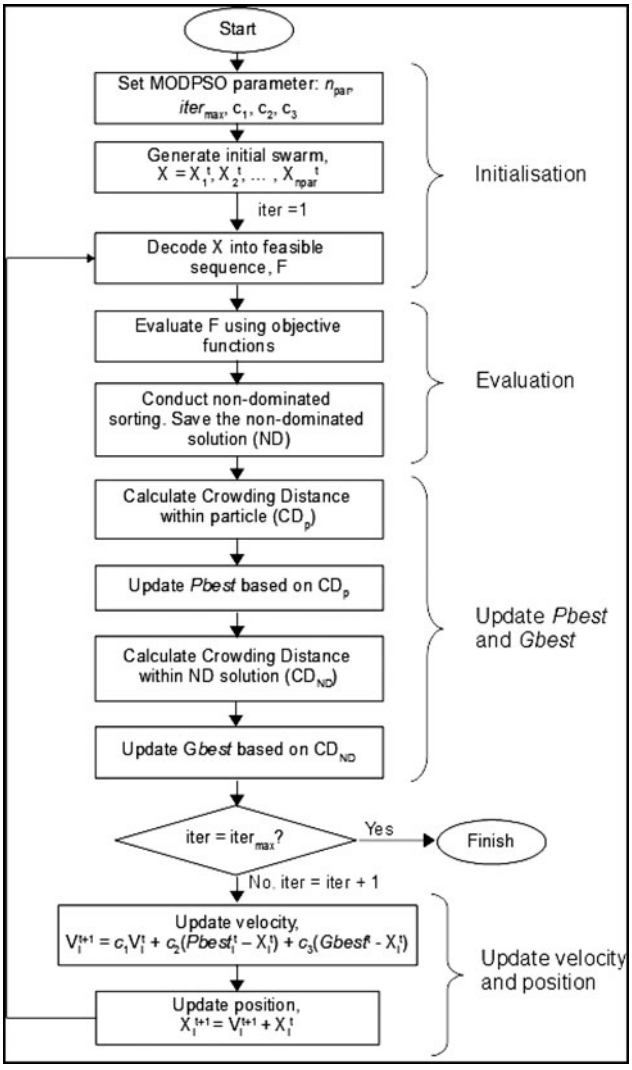


Fig. 2. Flowchart of MODPSO algorithm.

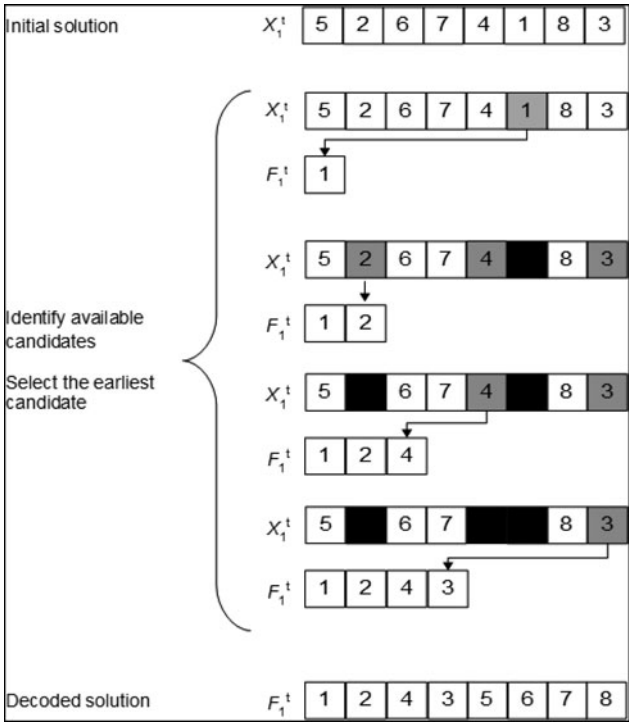


Fig. 3. Example of decoding procedure.

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

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.

Evaluation

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

Update Pbest and Gbest

The Pbest represents the best solution over the iterations within a similar particle. Meanwhile the Gbest is the best solution among all the particles. In the original PSO, the Pbest and Gbest are simply determined based on the fitness of solution. However, in the 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.

Update position and velocity

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

$$V_i^{t+1} = c_1 V_i^t + c_2(\text{Pbest}_i^t - X_i^t) + c_3(\text{Gbest}_t - X_i^t) \quad (8)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (9)$$

Equation (8), calculate the velocity for $(t+1)^{\text{th}}$ iteration. This formula takes into account the current velocity and distance between

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

Pbest and Gbest with the current position, X_i^t. Meanwhile, Eq. (9) updates the position for $(t+1)^{\text{th}}$ iteration, X_i^{t+1}. For the discrete representation, the following procedures are applied (Rameshkumar *et al.*, 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 × velocity): $(V_2 = c.V_1)$.
If r and$<c$, $v_2 = v_1$, 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:

$$v_j = \begin{cases} v_{1,j} \text{ if } v_{1,j} \neq 0,\ v_{2,j} = 0 \\ v_{1,j} \text{ if } v_{1,j} \neq 0,\ v_{2,j} \neq 0,\ r < \text{cp}, \\ v_{2,j} \text{ otherwise} \end{cases} \quad (10)$$

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 $x_{1,j}^{t+1} = v_{1,j}$

Experiment design

In the previous work, a tuneable test problem generator to provide sufficient test problem for integrated ASP and ALB has been developed (Ab Rashid *et al.*, 2012). The results indicate that the ASP and ALB problem difficulties can be increased using a 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 settings 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 settings will be used (Levels 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 settings (Problems 1, 18, and 35) represent the reference variable settings for Levels 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).

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

(i) Multi-objective GA (MOGA): This algorithm is one of the most frequently used algorithms to optimize the independent ASP and ALB problem, according to the survey (Rashid *et al.*, 2011).
(ii) ACO: The ACO algorithm has been implemented for a single-model integrated ASP and ALB optimization (Yang *et al.*, 2013; Lu and Yang, 2016).
(iii) Hybrid GA (HGA): The HGA that is proposed by Chen *et al.* is the most cited published work on integrated ASP and ALB optimization for a single model (Chen *et al.*, 2002). This algorithm combined the heuristic approach in line with balancing GA. The output solution from the heuristic approaches will be inserted into the initial population for GA.
(iv) Elitist Non-Dominated Sorting GA (NSGA-II): NSGA-II was introduced by Deb (2002). This algorithm is selected because of its popularity in solving multi-objective optimization.
(v) Multi-objective Particle Swarm Optimization (MOPSO): The MOPSO algorithm introduced to extend the PSO application for multi-objective optimization (Coello Coello and Lechuga, 2002).
(vi) Discrete Particle Swarm Optimization (DPSO): DPSO presents the discrete updating procedure to update the position and velocity (Rameshkumar *et al.*, 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:

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

In this experiment, 15 test problems from Table 2 are selected, which consist of five 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 is collected and filtered to find the non-dominated solution set.

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, $\hat{\eta}$: Shows the number of non-dominated solutions generated by each algorithm in the Pareto solution set. The higher $\hat{\eta}$ 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 the algorithm. Smaller ER indicates better algorithm performance.

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

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

Table 3. Frequency of the rank obtained by each algorithm

Indicator	Rank	MOGA	ACO	HGA	NSGA-II	MOPSO	DPSO	MODPSO
$\hat{\eta}$	1	0	0	0	5	0	0	47
	2	0	1	8	37	1	0	4
	3	11	2	22	5	12	8	0
	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
ER	1	0	0	0	31	0	0	23
	2	0	0	5	17	1	0	26
	3	10	6	20	3	4	5	2
	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
GD	1	0	0	0	30	0	0	24
	2	2	0	6	17	1	0	22
	3	17	3	23	2	2	1	3
	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
Spacing	1	7	8	7	8	13	5	3
	2	10	2	4	10	10	6	9
	3	11	7	6	7	13	5	2
	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
$\text{Spread}_{\text{max}}$	1	2	1	1	0	10	4	33
	2	3	1	8	2	16	15	7
	3	5	2	14	4	11	12	2
	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

(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 a better solution set, having better spacing between each solution.
(v) Maximum Spread, $\text{Spread}_{\text{max}}$: Measures the spread of solutions found by each algorithm. Larger maximum spread is the better.

Results of computational experiment

Due to the large size data from the optimization, the results simplified the data by using a 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 presents the frequency of the rank obtained by each algorithm for different indicators and test problems. For the non-dominated solution in

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

Table 4. Mean of performance indicators

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

[a]Larger the better indicator.
[b]Smaller the better indicator.

Table 5. Average CPU time for different problem size

	Average CPU time (s)						
Problem size	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

Pareto optimal ($\hat{\eta}$) 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 with 41.5% and 58.5%, respectively. Both algorithms also dominate the best performance for Generational Distance (GD) indicator with 43% of the better performance for MODPSO and 53% for NSGA-II. Meanwhile, the Spacing indicator shows different patterns, 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 $\text{Spread}_{\text{max}}$ indicator that measures the extent of solution distribution presents that the MODPSO algorithm produces a better solution in 70% of the problem. The MOPSO performs better with 18%, while the remaining balances are shared among DPSO (6%), MOGA (4%), and HGA (2%).

Table 4 presents the mean of performance indicators for all test problems. Based on the mean values, the best performance of $\hat{\eta}$ indicator is observed in the MODPSO followed by the 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 $\text{Spread}_{\text{max}}$ indicator.

Table 5 shows the average CPU time for different problem sizes. 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.

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 $\hat{\eta}$ and ER. The same experiment also came out with the best accuracy (i.e. GD indicator).

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 $\hat{\eta}$ 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 $\text{Spread}_{\text{max}}$ indicated that high c_1, medium c_2, and medium c_3 coefficients' respective levels contributed to better solution distribution.

Discussion of results

In general, the result from the experiment shows that the performance of algorithms in optimizing the integrated mixed-model

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

Table 6. MODPSO coefficient experiment results

	Coefficients			Mean of performance indicators				
Experiment number	c_1	c_2	c_3	$\hat{\eta}$	ER	GD	Spacing	$\text{Spread}_{\text{max}}$
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

ASP and ALB appears to be dominated by NSGA-II and the proposed MODPSO algorithms, especially in four performance indicators (i.e. $\hat{\eta}$, ER, GD, and $\text{Spread}_{\text{max}}$). 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 optimizing the 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 states that there is a 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 the critical f-value (f^*) as suggested in the f-distribution table (Coolidge, 2000). The result of the ANOVA test is presented in Table 7.

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.

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

The q value is acquired from Tukey's table, MSW is the mean squares within groups from the 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.

In Table 8, the values that are labeled "a" show the MODPSO has a better mean difference over the comparison algorithm, while the values labeled "b" 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 $\hat{\eta}$ 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 compared with MODPSO; however, the difference is not significant because the absolute mean difference is 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 $\text{Spread}_{\text{max}}$ 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 a better solution.

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

Furthermore, the $\text{Spread}_{\text{max}}$ 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

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

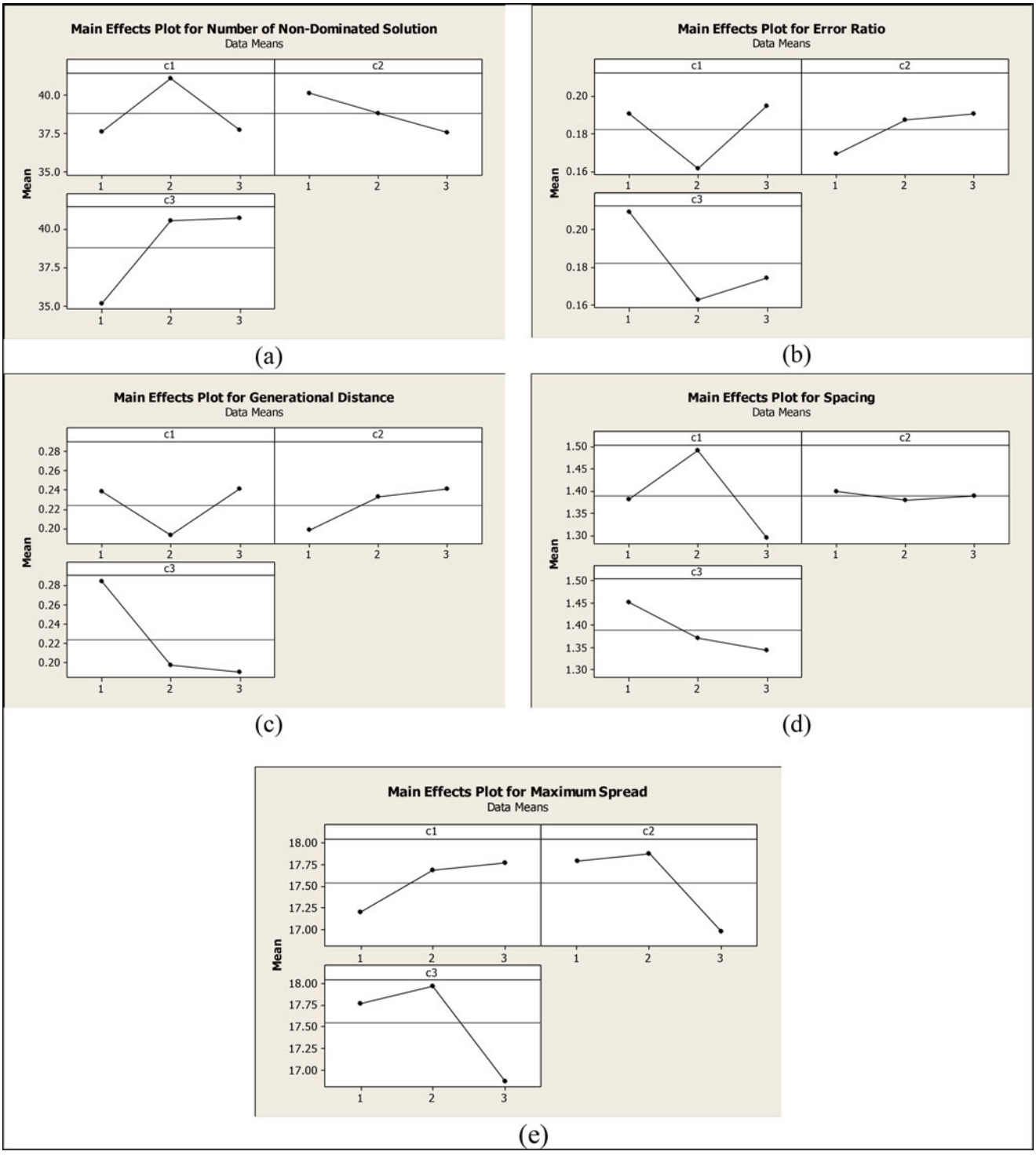


Fig. 4. Effect of w, c_1, and c_2 on performance indicators: (a) $\hat{\eta}$, (b) ER, (c) GD, (d) Spacing, and (e) $\text{Spread}_{\text{max}}$.

other algorithms. The Spacing indicator considers all non-dominated solutions that were 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 a larger mean of generated solutions.

The results from the experiments and statistical tests summarize that the MODPSO has shown significant improvement over the majority of compared algorithms in $\hat{\eta}$, ER, GD, and

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

Table 7. Summary of ANOVA test

	$\bar{\eta}$	ER	GD	Spacing	$\text{Spread}_{\max}$
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.

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

	Absolute mean difference between MODPSO and comparison algorithm				
Indicator (HSD*)	$\bar{\eta}$ (4.5902)	ER (0.0906)	GD (0.6131)	Spacing (0.1621)	$\text{Spread}_{\max}$ (1.3337)
Comparison algorithm					
MOGA	**36.2353**[a]	**0.6991**[a]	**1.72551**	0.2037[b]	**2.7379**[a]
ACO	**39.1176**[a]	**0.7586**[a]	**2.19541**	0.0908[b]	**3.8292**[a]
HGA	**32.9608**[a]	**0.6547**[a]	**1.73211**	0.2499[b]	**1.9406**[a]
NSGA-II	**13.7843**[a]	0.0094[b]	0.0944[b]	0.0580[a]	**3.4928**
MOPSO	**35.7451**[a]	**0.7184**[a]	**2.0523**[a]	0.2839[b]	1.2789[a]
DPSO	**37.5294**[a]	**0.7399**[a]	**2.0985**[a]	0.2781[b]	**1.5936**[a]

[a]Better absolute mean difference for MODPSO.
[b]Better absolute mean difference for comparison algorithm.

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

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

[a]Better absolute mean difference for NSGA-II.
[b]Better absolute mean difference for comparison algorithm.

$\text{Spread}_{\max}$ indicators. In comparison with all other algorithms, the performance of MODPSO is closely followed by NSGA-II, where MODPSO only shows significant improvement over NSGA-II in $\bar{\eta}$ and $\text{Spread}_{\max}$ indicators. In order to compare the performance of NSGA-II, the mean difference between NSGA-II and other algorithms is calculated and presented in Table 9.

Table 9 indicates that the NSGA-II has a 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 ($\text{Spread}_{\max}$). 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 has shown a significant improvement over NSGA-II in two of the indicators (i.e. $\bar{\eta}$ and $\text{Spread}_{\max}$), while there is no significant improvement of NSGA-II over MODPSO algorithm. Furthermore, for $\text{Spread}_{\max}$ 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 terms 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 $\bar{\eta}$ indicator over all algorithms for all reference setting. For ER indicator, the MODPSO consistently demonstrates

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

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

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

[a]Better absolute mean difference for MODPSO.
[b]Better absolute mean difference for comparison algorithm.

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 $\text{Spread}_{\max}$ indicator, the proposed algorithm also did not show significant improvements in the low-level reference setting. However, when the reference setting is moved to a 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.

The superior performance of MODPSO in optimising integrated mixed-model ASP and ALB arises because this algorithm was specifically developed for discrete multi-objective optimisation problem. This algorithm uses a similar procedure for Initialization, Evaluation, and Selection strategies as in NSGA-II. The NSGA-II is another algorithm that specifically developed for multi-objective optimization problems that also performed well in this application. However, it does not have a fine tuning feature. The fine tuning feature means the ability of an algorithm to make small adjustments to the 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 toward the end of iterations. According to discrete updating procedure [Subtraction operator $(X_i - X_j)$] in section "Update position and velocity", zero velocity is given when a similar element in X_i and X_j is found (this is the case when all particles move toward the best solution at the end of iterations). When the majority of velocity elements are zero, only small changes occur in assembly sequence as presented by Addition operator $(X_i + V_i)$ in section "Update Position and velocity". This feature allows fine tuning of the assembly sequences in MODPSO.

Conclusions

This paper formulates and studies the optimization of integrated mixed-model ASP and ALB problem using MODPSO. A set of

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

test problems with different range of difficulties has been used to test the performance of MODPSO in optimizing the integrated mixed-model ASP and ALB. In addition, MODPSO coefficient tuning has been conducted to identify the best settings for optimization.

The experimental results indicate that, in general, the MODPSO algorithms performed better than other comparison algorithms. The statistical test concluded that the MODPSO has shown a 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 the 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 initiated the study on integrated mixed-model ASP and ALB optimization. At the same time, it also indicates that the MODPSO algorithm is able to optimize 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 the future, an effort to improve the algorithm performance, especially in solution uniformity, is proposed to improve the overall solution quality. The first suggestion to improve the solution uniformity is to consider the historical data in the crowding distance. This will make the unselected solutions, because of mating pool capacity, to 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 toward the extreme solution, besides the Gbest. Therefore, the search direction becomes more diverse.

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

Conflict of interest. None.

References

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

Ab Rashid MFF, 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**, 1900–1913.

Ab. Rashid MFF, Tiwari A and Hutabarat W (2017) Comparison of sequential and integrated optimisation approaches for ASP and ALB. *Procedia CIRP*. The Author(s) **63**, 505–510.

Becker C and Scholl A (2006) A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research* **168**, 694–715.

Buyukozkan K, Kucukkoc I, Satoglu SI and Zhang DZ (2016) Lexicographic bottleneck mixed-model assembly line balancing problem: artificial bee colony and tabu search approaches with optimised parameters. *Expert Systems with Applications* **50**, 151–166.

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**, 447–457.

Coello Coello CA and Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. in *Proceedings of the* 2002 *Congress on Evolutionary Computation. CEC*'02 (*Cat. No.*02*TH*8600). IEEE, pp. 1051–1056. doi: 10.1109/CEC.2002.1004388.

Coolidge FL (2000) *Statistics: A Gentle Introduction*. London: SAGE Publication.

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

Hu SJ, Zhu X and Koren Y(2008) Product variety and manufacturing complexity in assembly systems and supply chains. *{CIRP} Annals – Manufacturing Technology* **57**, 45–48.

Hu Y-J, Wang Y, Wang Z-L, Wang Y-Q and Zhang B-C (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**, 71–82.

Jusop M and Ab Rashid MFF (2015) A review on simple assembly line balancing type-e problem. *IOP Conference Series: Materials Science and Engineering* **100**, 12005.

Kara Y, Özgüven C, Seçme NY and Chang C-T (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**, 725–737.

Lin M-C, Lin C-C, Tai Y-Y, Chen M-S and Tseng C-C (2012) A modularized contact-rule reasoning approach to the assembly sequence generation for product design. *Concurrent Engineering* **20**, 203–221.

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**, 243–256.

Marian RM (2003) *Optimisation of Assembly Sequences using Genetic Algorithm*. Ph.D. Thesis. Adelaide, Australia: 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**, 45–54.

Penciuc D, Duigou JL, Daaboul J, Vallet F and Eynard B (2016) Product life cycle management approach for integration of engineering design and life cycle engineering. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **30**, 379–389.

Rahman HF, 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**, 87–103.

Rameshkumar K, Suresh RK and Mohanasundaram KM (2005) Discrete Particle Swarm Optimization (DPSO) algorithm for permutation flowshop scheduling to minimize makespan. *Advances in Natural Computation* **3612**, 572–581.

Rashid MFF, 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**, 335–349.

Roshani A and Nezami FG (2017) Mixed-model multi-manned assembly line balancing problem: a mathematical model and a simulated annealing approach. *Assembly Automation* **37**, 34–50.

Shankar P, Summers JD and Phelan K (2017) A verification and validation planning method to address change propagation effects in engineering design and manufacturing. *Concurrent Engineering* **25**, 151–162.

Tasan SO and Tunali S (2008) A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing* **19**, 49–69.

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**, 97–116.

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**, 5951–5977.

Yang Z, Lu C and Zhao HW (2013) An Ant Colony Algorithm for integrating assembly sequence planning and assembly line balancing. *Applied Mechanics and Materials* **397–400**, 2570–2573.

Yoosefelahi A, Aminnayeri M, Mosadegh H and DavariArdakani H (2012) Type II robotic assembly line balancing problem: an evolution strategies

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press

algorithm for a multi-objective model. *Journal of Manufacturing Systems* **31**, 139–151.

Zeng K, Tan Z, Dong M and Yang P (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**, 429–437.

Zhong Y (2017) Hull mixed-model assembly line balancing using a multi-objective genetic algorithm simulated annealing optimization approach. *Concurrent Engineering* **25**, 30–40.

Zhu X, Hu SJ, Koren Y and Huang N (2012) A complexity model for sequence planning in mixed-model assembly lines. *Journal of Manufacturing Systems* **31**, 121–130.

Mohd Fadzil Faisae Ab. Rashid received his PhD from Cranfield University, UK in 2013. During his early career, he worked 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 a Professor of Manufacturing Informatics and attained his PhD degree from Cranfield University in Evolutionary Computing Techniques for Handling Variable Interaction in Engineering Design Optimization. His PhD research was carried out as part of an EPSRC project and was successfully completed in 2001 within the 2-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 optimization and is currently involved in an AMSCI-funded project with Cosworth and Flexeye aimed at optimizing a flexible manufacturing system.

https://doi.org/10.1017/S0890060419000131 Published online by Cambridge University Press