Computers & Industrial Engineering 69 (2014) 89–104

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

A Pareto biogeography-based optimisation for multi-objective two-sided assembly line sequencing problems with a learning effect ☆

Parames Chutima *, Wanwisa Naruemitwong [1]

Department of Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand

ARTICLE INFO

Article history:
Received 11 February 2013
Received in revised form 30 December 2013
Accepted 5 January 2014
Available online 16 January 2014

Keywords:
Biogeography-based optimisation
Mixed-model products
Two-sided assembly line sequencing
Learning effect

ABSTRACT

This research presents a Pareto biogeography-based optimisation (BBO) approach to mixed-model sequencing problems on a two-sided assembly line where a learning effect is also taken into consideration. Three objectives which typically conflict with each other are optimised simultaneously comprising minimising the variance of production rate, minimising the total utility work and minimising the total sequence-dependent setup time. In order to enhance the exploration and exploitation capabilities of the algorithm, an adaptive mechanism is embedded into the structure of the original BBO, called the adaptive BBO algorithm (A-BBO). A-BBO monitors a progressive convergence metric in every certain generation and then based on this data it will decide whether to adjust its adaptive parameters to be used in the next certain generations or not. The results demonstrate that A-BBO outperforms all comparative algorithms in terms of solution quality with indifferent solution diversification.

1. Introduction

Two-sided assembly lines (2SALs) are ubiquitously utilised in many manufacturing systems where large-sized products are produced, e.g. small utility vehicles (Bartholdi, 1993), buses and trucks (Kim, Kim, & Kim, 2000), automobiles (Lee, Kim, & Kim, 2001), appliances (Lapierre & Ruiz, 2004), domestic products (Baykasoglu & Dereli, 2008) and motorcycles (Cortés, Onieva, & Guadix, 2010).

A 2SAL is characterised by one or more mated stations connecting in series. Each mated station accommodates a pair of directly facing stations, called left and right stations. This pair of opposite stations in the mated station terms the other as a companion (Lee et al., 2001). A material handling system, typically a conveyor, runs through the middle of the mated stations along the 2SAL to mobilise in-process workpieces from one mated station to another. Two workers, each of which is responsible for one side of the mated station, perform their non-overlapping tasks on the same individual workpiece in parallel without interfering with one another. An example of the 2SAL with one mated station using the average task time in sequencing each task is shown in Fig. 1.

Operational advantages offered by 2SALs over traditional one-sided assembly lines (1SALs) comprise setup avoidance or reduced setup time, efficient utilisation of space, reduced material handling costs, minimised number of stations (line length) and staff, reduced worker movement, increased teamwork, reduced throughput time, and increased possibility of tool and fixture sharing (Bartholdi, 1993; Lee et al., 2001).

To increase flexibility and efficiency in manufacturing large-sized products, it is necessary to allow operations to be performed from both sides of the 2SAL in parallel. However, some tasks have preferred operational directions since their assembly operations could be manipulated more easily from a particular side. Hence, a preferred operational direction has to be defined for each task as follows: left-type task (L), right-type task (R), or either-type task (E) without any preference in an operational direction (Kim et al., 2000).

A mixed-model sequencing problem of the 2SAL is related to the determination of the sequence for introducing diversified models onto the assembly line. It is a crucial proponent for an effective implementation of just-in-time (JIT) concepts particularly for diversified small lot production. A remarkable constraint that causes the sequencing problem in 2SALs different from 1SALs is the sequence-dependent finish time of tasks (Lee et al., 2001; Simaria & Vilarinho, 2009). This constraint is an inheritance of the line balancing in 2SALs by which tasks are allocated to given stations within a limited cycle time based on their preferred operational directions and precedence relationships.

As mentioned earlier that tasks in 2SALs have to be performed in parallel by two facing workers. For example, for some mated stations, the left-type tasks may have to wait for the right-type tasks in the companion stations to finish before the immediate predecessors of the left-type tasks could be started, and vice versa. Due to

☆ This manuscript was processed by Area Editor Manoj Tiwari.

* Corresponding author. Tel.: +662 2186847; fax: +662 2513969.
E-mail addresses: cparames@chula.ac.th (P. Chutima), n.wanwisa@gmail.com (W. Naruemitwong).
[1] Tel.: +662 2186847; fax: +662 2513969.

0360-8352/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cie.2014.01.001

Task	Model			Average Task Time	Preferred Direction
	A	B	C		
1	3	2	1	2	L
2	5	3	4	4	E
3	4	1	4	3	L
4	2	1	3	2	R

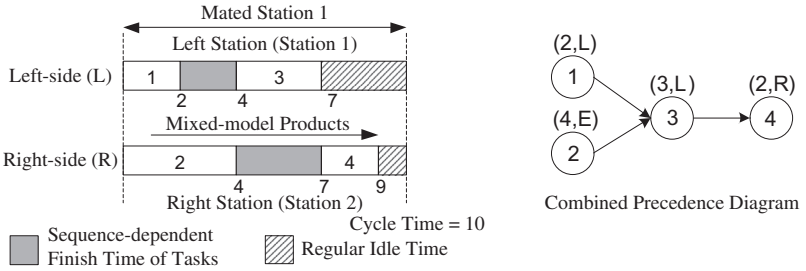


Fig. 1. An example of a 2SAL with one mated station.

the waiting, the earliest starting times of some tasks have to be postponed by other tasks assigned to the companion stations resulting in a more complicated operational planning of 2SALs. This is illustrated in Fig. 1 where the worker in the left station who completes task 1 has to wait until his/her companion in the right station to finish task 2 (called sequence-dependent finish time of task 2), before he/she can start task 3. This constraint has never been appeared in 1SALs but it has to be seriously considered in 2SALs. Disregarding this constraint, the development of a task sequence may end up with an unfeasible schedule.

Learning effects are the situations when the processing time required for producing a single item continuously decreases with the additional production of a product as a result of an increased worker efficiency (Mosheiov, 2001). Although many researchers have studied assembly line sequencing problems, they mostly assumed that the task times, even for repetitive tasks, are independent from the learning of the workers. However, in practice, the workers can gain better skills, knowledge and experience over time on how to work more efficiently with a higher quality level after working on the same or similar operations repeatedly. As a result, tasks can be performed more quickly if they are scheduled later in the sequence. To demonstrate the impact of learning effect to the tasks sequencing on the 2SAL, Fig. 2 shows that the throughput time of model A (as previously shown in Fig. 1) is reduced from 11 to 8.7995 when the position-based learning of 90% learning rate is assumed for the mated station 1.

In this paper, the multi-objective mixed-model two-sided sequencing problem with a learning effect (MMTSPLE) is considered. To the best of our knowledge, no research on this issue has been published before. Since this problem falls into an NP-hard class, the biogeography-based optimisation (BBO) approach, a novel meta-heuristic, is developed to optimise three conflicting objectives simultaneously, i.e. minimising the variance of production rate, minimising the total setup time and minimising the total utility work.

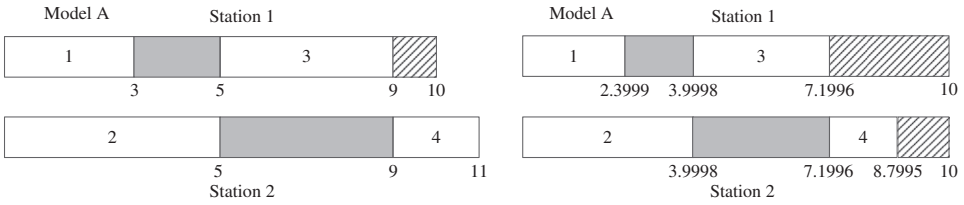
The remainder of this paper is organised as follows. In the following section, the related literature is reviewed. The detailed description of the multi-objective optimisation problem is presented in Section 3. The solution procedures and numerical example are discussed in Section 4, followed by Section 5 which explains the experimental design and results. Finally, the concluding remarks are elaborated in Section 6.

2. Literature survey

2.1. Mixed-model assembly line sequencing

Although the sequencing problems of mixed-model assembly lines (MALs) have been widely studied in the scientific literature, the sequencing problems of 1SALs, rather than 2SALs, are the main emphasis. Boysen, Fliedner, and Scholl (2009) gave a recent review in this research area and exemplified various objectives for assessing the performance of mixed-model sequencing algorithms, e.g. minimum workload variation, minimum part usage rate variation, minimum utility work, minimum setup cost, minimum risk of line stoppage, minimum total idle time, etc. It is obvious that single objective optimisation used to be a normal practice for earlier publications, e.g. Miltenburge (1989), Miltenburg, Steiner, and Yeomans (1990) and Stiener and Yeomans (1993).

Recently, researchers have tried to imitate the management decision processes to optimise more than one objective simultaneously in developing the best sequence for a given product mix. Since these objectives often conflict with each other, a compromised solution becomes inevitable. Metaheuristics, general-purpose algorithms being applicable to large-scale problem and normally can deliver satisfactory solutions in a reasonable time, seem to be prominent since the MAL sequencing problem is NP-hard in nature. Both families of metaheuristics (Talbi, 2009) have been appeared in this research area including single-solution based algorithms which are exploitation oriented (e.g. tabu search (Bard, Shtub, & Joshi, 1994; McMullen, 1998), simulated annealing algorithm (McMullen & Frazier, 2000) and memetic algorithm (Tavakkoli-Moghaddam & Rahimi-Vahed, 2006; Chutima and Pinkoompee, 2009)) and population-based algorithms which are exploration



(a) Model A without learning effect (b) Model A with 90% learning effect

Fig. 2. The impact of a learning effect to 2SAL sequencing.

oriented (e.g. Pareto stratum-niche cubicle genetic algorithm (Hyun, Kim, & Kim, 1998), genetic algorithm (Mansouri, 2005; McMullen, Tarasewich, & Frazier, 2000; McMullen, 2001a), ant colony optimisation (McMullen, 2001b) and multi-objective scatter search (Rahimi-Vahed, Rabbani, Tavakkoli-Moghaddam, Torabi, & Jolai, 2007)).

As mentioned earlier that an additional constraint has to be considered in sequencing 2SALs as opposed to 1SALs, i.e. the sequence-dependent finish time of tasks. The existence of such constraint enhances the complexity of the problem substantially. Consequently, instead of considering just the launching order of a product mix, the interaction between the workers performing tasks in the same mated station has to be taken into account for effective sequencing 2SALs.

2.2. Learning effects

Learning effects play a significant role when changes occur in the production environment and employees need to gain acquaintance with such new situations, e.g. new workers without relevant experiences, newly installed machinery, newly implemented workflows, new jobs that have never been done before, etc. (Biskup, 2008). The traditional formulation of learning effects developed under the mass production concept is as follows.

$$p_{[k]} = p_{[1]}k^a \tag{1}$$

where $p_{[k]}$ is the processing time of the kth item on the cumulative production quantity k, $p_{[1]}$ is the processing time of the first item, and $a = \log_2 LR \leqslant 0$ is the learning index depending on the learning rate LR.

In scheduling problems, the processing times of jobs are not necessarily a fixed parameter since workers are normally assigned with various activities that are subject to learning effects (Pinedo, 2008). Unsurprisingly, many researchers tried to incorporate the effects of learning into scheduling contexts. As a pioneer researcher who brought the concept of learning effects into scheduling problems, Biskup (1999) proposed a formula for position-based learning (position-dependent or time-independent learning effects) in scheduling environments under the assumption that learning occurs primarily by the number of times that jobs are repeatedly produced regardless of the processing times of the jobs that have already been produced, such as machine setups, controlling and operating machines, and reading data. As a result, the processing time of a job is shortened if it is scheduled at a later position in the sequence. This assumption is practical for the case that the processing times of the jobs are purely machine-driven without or with very little human intervention in high technology manufacturing processes. The formulation is as follows.

$$p_{ir} = p_i r^a \tag{2}$$

where p_{ir} is the processing time of job i being scheduled in the position rth and p_i is the normal processing time needed to perform job i ($i = 1, \ldots, n$) without any learning effects. Alternatively, Kuo and Yang (2006) suggested another form called sum-of-processing-time based learning (time-dependent learning effects) which considers the experience that the worker gains while performing jobs, e.g. offset printing, inspection steel plates in a foundry shop, etc. In other words, the more processing times the worker practices doing jobs, the higher experience the worker gains, leading to less processing times needed on the subsequent jobs. The formulation is as follows.

$$p_{ir} = p_i(1 + p_{[1]} + \cdots + p_{[r-1]})^a = p_i\left(1 + \sum_{k=1}^{r-1} p_{[k]}\right)^a \tag{3}$$

It is noticeable that the actual processing time of job i is affected by the processing times of the previous $(r-1)$ jobs in the sequence. A recent survey indicated that although the learning effect has been widely employed in management science, particularly in single machine and flow shop scheduling problems (Biskup, 2008; Janiak, Krysiak, & Trela, 2011), it has rarely been studied under the general context of production scheduling in 2SALs.

3. Multi-objective optimisation

3.1. Multi-objective functions

The operational condition of the 2SAL in this research is adapted from Hyun et al. (1998). A mixed-model 2SAL with CT cycle time is considered. An automatic conveyor is employed to transport large-sized workpieces from one mated station to another at a constant speed v_c. The line balancing decision, i.e. the medium-term planning, partitions the 2SAL into N_M mated stations with a fairly uniform workload distribution. The sequential number for each mated station is $n_m = 1, 2, \ldots, N_M$. A pair of stations resided in each mated station are given serial numbers as $n_w = 2n_m - 1$ and $2n_m$, for the left-side and right-side stations, respectively ($n_w = 1, 2, \ldots, N_W = 2N_M$). Workers are allowed to perform assembly operations only within their own station boundaries. The task times are deterministic and the walking times of the workers are negligible. Assume that a new product mix is launched into the 2SAL so that learning can occur at all stations, and the position-based learning is gained when workers perform setups and assembly operations, meaning that learning is affected by the number of setups and jobs being produced.

Let M be the number of models to be produced during a planning horizon, D_m be the demand of product model $m (m = 1, 2, \ldots, M)$, and h be the greatest common divisor of $(D_1, D_2, \ldots, D_M)$. The vector $\underline{d} = (d_1, d_2, \ldots, d_M)$, where $d_m = D_m/h$, represents a model mix in a production cycle, called a minimum part set (MPS). The number of products to be produced in one cycle is $I = \sum_{m=1}^{M} d_m$. The products are launched onto the conveyer at a constant rate with the launch interval $\gamma = T/(I * N_M)$, where T = total operation time required to produce one cycle of the products in an MPS. In the 2SAL, the work in a mated station is completed when both workers finish all of their assigned tasks. Let $t_{j,m}$ be the total operation time for model m at workstation j, and $Y_{j,m}$ be the total unavoidable idle time caused by the sequence-dependent finish time of tasks for model m at workstation j which is determined from the Gantt chart of each model (Chutima & Jitmetta, 2013). The formulation of T is as follows.

$$T = \sum_{n_m=1}^{N_M} max\left[\sum_{m=1}^{M}(t_{2n_m-1,m}d_m + Y_{2n_m-1,m}), \sum_{m=1}^{M}(t_{2n_m,m}d_m + Y_{2n_m,m})\right] \tag{4}$$

Three objectives to be simultaneously optimised in this paper include minimising the variance of production rate, minimising the total utility work and minimising the total setup time. These objectives are among the top rankings in several industries, e.g. automobile industry. which can be explained and formulated as follows.

3.1.1. Minimising the variance of production rate

A constant rate of part usage is one of the basic requirements in JIT production systems since it can minimise work-in-process and finished goods inventories while meeting customer demand. The objective targets at minimising the discrepancy between the ratio of actual production and that of demand to all models so that parts can be continuously supplied to the assembly line in a stable manner. The formulation is as follows (Hyun et al., 1998).

$$minimise\ f_1\ (x) = \sum_{i=1}^{I}\sum_{m=1}^{M}\left[\sum_{l=1}^{i}\frac{X_{l,m}}{i} - \frac{d_m}{I}\right]^2 \tag{5}$$

where $X_{l,m} = 1$, if the lth product in a sequence is model m; and $X_{l,m} = 0$, otherwise. The first and second terms are the production ratio and the demand ratio of model m, respectively.

3.1.2. Minimising the total utility work

The utility work is a reaction on imminent work overload. Whenever the operations assigned to a regular worker (i.e. a worker who performs operations in either side of a mated station) cannot be completed within the boundary of a station, the incomplete operation is called the *utility work*. Utility workers are called upon to help regular workers if work overload is noticed. Minimising the utility work renders reducing labour cost and the risk of line stoppage.

sequence at mated station n_m. Chutima and Jitmetta (in press) developed the formulation to compute the utility work in a 2SAL as follows.

$$minimise\ f_2\ (x) = \sum_{n_m=1}^{N_M}\left(\sum_{i=1}^{I}U_{i,n_m} + Z_{(i+1),n_m}/v_c\right) \tag{6}$$

where

$$U_{i,n_m} = \left\{\begin{array}{l}\max\left[0, \frac{\left(Z_{i,n_m} + v_c\sum_{m=1}^{M}X_{i,m}\{t_{2n_m-1,m} + Y_{2n_m-1,m}\} - L_{n_m}\right)}{v_c}\right] \\ +\max\left[0, \frac{\left(Z_{i,n_m} + v_c\sum_{m=1}^{M}X_{i,m}\{t_{2n_m,m} + Y_{2n_m,m}\} - L_{n_m}\right)}{v_c}\right]\end{array}\right\} \tag{7}$$

$$Z_{(i+1),n_m} = max\left\{\begin{array}{l}max\left[0, min\left(Z_{i,n_m} + v_c\sum_{m=1}^{M}X_{i,m}\{t_{2n_m-1,m} + Y_{2n_m-1,m}\} - \gamma v_c, L_{n_m} - \gamma v_c\right)\right], \\ max\left[0, min\left(Z_{i,n_m} + v_c\sum_{m=1}^{M}X_{i,m}\{t_{2n_m,m} + Y_{2n_m,m}\} - \gamma v_c, L_{n_m} - \gamma v_c\right)\right]\end{array}\right\} \tag{8}$$

From Fig. 3, let L_{n_m} be the fixed line length of mated station $n_m (L_{n_m} = v_c * CT)$, U_{i,n_m} be the amount of utility work required for the ith product in a sequence at mated station n_m, $X_{i,m}$ be the existent of product model m at the ith product sequence ($X_{i,m} = 1$, if the ith product in a sequence is model m; and $X_{i,m} = 0$, otherwise), and Z_{i,n_m} be the starting position of the work on the ith product in a

3.1.3. Minimising the total setup time

In some industries such as an automotive body factory, engine mounting or robotic lines, setup time could become a high proportion of task time and depend on the directed preceding models (Andrés, Miralles, & Pastor, 2008; Hyun et al., 1998). The formulation for sequence-dependent setups is as follows.

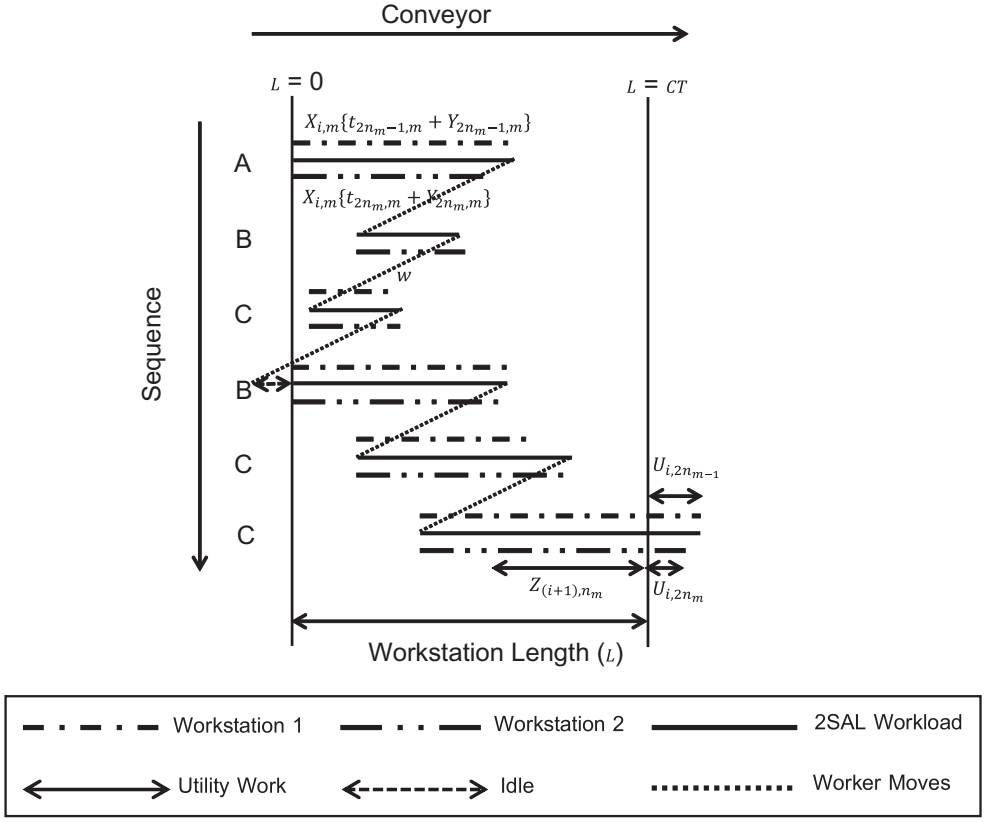


Fig. 3. Utility works on 2SAL caused by the product mix A–B–C–B–C–C.

$$minimise\ f_3\ (x) = \sum_{n_w=1}^{N_w}\sum_{k=1}^{I} s^{n_w}_{[k-1],[k]} \tag{9}$$

where $s^{n_w}_{[k-1],[k]}$ is the setup time incurred on workstation n_w if the products in position $k-1$ and k of the launch order are different, I is the total demand (also represents the number of positions in a sequence).

3.2. Pareto-optimal solutions

For most practical problems, two or more objectives have to be optimised simultaneously. Since these objectives often conflict with each other, obtaining a solution which achieves the best objectives for them all is very unlikely. Rather, a compromised solution with acceptable values of all objectives being trade-off and reflecting the current needs of the decision maker is much more pragmatic. The problems as such are known as multi-objective optimisation problems (MOPs). Without loss of generality, Coello (1999) proposed the formulation for minimising MOPs as follows:

$$Minimise_{x\in\Omega} F(X) \tag{10}$$

Eq. (10) states that we want to find $X \in R^k$ which minimises $F(X) = \{f_1(X), f_2(X), \ldots, f_k(X)\}$, where solution $X = \{x_1, x_2, \ldots, x_n\}$ is a vector of decision variables for the considered problem, Ω is the feasible solution space, and $f_i\ (X)$ is the ith objective function ($i = 1, 2, \ldots, k$). For a minimisation problem, a decision vector X is said to dominate a decision vector Y, written as $X \succ Y$, if and only if:

$$(a)\ f_i\ (X) \leqslant f_i(Y), \quad \text{for all } i \in \{1,2,\ldots,k\},\ \textit{and} \tag{11}$$

$$(b)\ f_i\ (X) < f_i(Y), \quad \text{for at least one } i \in \{1,2,\ldots,k\} \tag{12}$$

All solutions that dominate others but not themselves are called non-dominated solutions. By definition, a Pareto-optimal solution X is a local optimal solution if there exist no solution Y located within a bowl of centre X and of radius ε (where $\varepsilon > 0$ and is a real number) dominating the solution X. In addition, a Pareto-optimal solution X is a global optimal solution which is not dominated by any other solutions in the feasible solution space Ω.

A set of Pareto-optimal solutions is preferable for the MOP rather than a single solution and the set that contains all feasible Pareto-optimal solutions is called a Pareto-optimal set. The corresponding images of the points in the Pareto-optimal set along a curve in the objective space that has a set of attributes collectively dominating all other points not on the frontier are termed as the Pareto-optimal frontier.

Although various approaches have been developed for solving MOPs, the multi-objective evolutionary algorithms (MOEAs) are the most promising solution technique since they can systematically handle a set of alternative solutions simultaneously, allowing an entire set of Pareto optimal solutions to be found in a single run. An in-depth review of MOEAs was given by Coello (2006).

To demonstrate quantitatively the dominance among various MOEAs, three metrics are recommended, i.e. the convergence to the Pareto-optimal set, the maintenance of diversity among the solutions of Pareto-optimal set and the number of Pareto solutions (Kumar & Singh, 2007; Rahimi-Vahed et al., 2007). The details of each metric are given as follows.

3.2.1. Convergence of the Pareto-optimal solutions

The convergence of the Pareto-optimal solution obtained towards a true Pareto-set (A^*) is the difference between the obtained solution set and the approximated true-Pareto set. Mathematically, it is defined as follows.

$$convergence(A) = \frac{\sum_{i=1}^{|A^*|} dt_i}{|A^*|} \tag{13}$$

$$dt_i = min_{j=1}^{|A^*|}\sqrt{\sum_{k=1}^{2}\left[\frac{f_k(x) - f_k(y)}{f_k^{max} - f_k^{min}}\right]^2} \tag{14}$$

where $|A^*|$ is the number of elements in set$A$, dt_i is the Euclidean distance between the non-dominated solution ith in the true-Pareto frontier (y) and the solution (x) obtained and f_k^{max} and f_k^{min} are the maximum and minimum values of kth objective functions in the true-Pareto set, respectively. For this metric, a lower value indicates superiority of the solution set.

3.2.2. Spread of the Pareto-optimal solutions

This metric computes the distribution of the Pareto-solutions obtained by calculating a relative distance between consecutive solutions, which can be formulated as follows.

$$spread(A) = \frac{sd_f + sd_l + \sum_{i=1}^{|A|-1}\|sd_i - \bar{sd}\|}{sd_f + sd_l + (|A|-1)\bar{sd}} \tag{15}$$

$$sd_i = \sqrt{\sum_{k=1}^{2}\left[\frac{f_k(x_i) - f_k(x_{i+1})}{f_k^{max} - f_k^{min}}\right]^2} \tag{16}$$

where sd_f and sd_l are the Euclidean distances between the extreme solutions and the boundary solutions of the resulting Pareto-optimal, $|A|$ is the number of elements in the Pareto solutions, sd_i is the Euclidean distance between consecutive solutions in the Pareto-solutions obtained for $i = 1, 2, \ldots, |A|-1$, $\bar{sd}$ is the average Euclidean distance of sd_i, and the operator "$\|\ \|$" indicates an absolute value. If this metric is zero, the Pareto solutions are distributed uniformly; but when poor uniformity is found, this metric can exceed 1.0.

3.2.3. Ratio of obtained non-dominated solutions

This metric indicates the coverage of one set over another. Let A_j be a solution set ($j = 1, 2, \ldots, J$). For comparing each J solution set ($A = A_1 \cup A_2 \cdots \cup A_J$), the ratio of non-dominated measure of the solution set A_j to the J solution sets is the ratio of solutions in A_j that are not dominated by any other solution in A, which is defined as follows.

$$R_{NDS}(A_j) = \frac{|A_j - \{x \in A_j | \exists y \in A : y \prec x\}|}{|A_j|} \tag{17}$$

where $y \prec x$ represents when the solution x obtained is dominated by the true-Pareto solution y. A higher ratio indicates the superiority of one solution set over another.

4. Biogeography-based optimisation

4.1. Basic concept of BBO

BBO is a prominent example of evolutionary algorithms developed recently by Simon (2008). Its solution searching mechanism aims at optimising large-sized complex problems. BBO adopted the idea from the biogeography theory which describes the geographical distribution of biological organisms. In the context of BBO, each candidate solution is analogous to an island and each feature on the solution, called a suitability index variable (SIV), characterises habitability. Habitat suitability index (HSI) measures the goodness of each solution and a good solution will have a high HSI value.

While the optimisation process is underway, the fitness of each solution in the current population is evaluated, and migration and mutation operations are applied constantly until the global optimal is reached. BBO improves the population by migrating good

solution features among islands. Each solution has its own immigration probability λ and emigration probability μ. Good solutions with high HSI have low immigration rate and high emigration rate. In contrast, poor solutions with low HSI have a low emigration rate and high immigration rate. The migration is a key step for improving habitability of solutions since it provides good solutions with a high chance of sharing their salient features to poor solutions; hence, it is recognised as an intrinsic exploitation mechanism of BBO. In this study, the sinusoidal migration model is employed since it outperformed others in most test-bed problems (Ma, 2010).

For the sinusoidal migration model, in which the migration rates are sinusoidal functions of the number of species in the habitat, whenever there are k species in the habitat, the immigration rate (λ_k) and emigration rate (μ_k) can be formulated as follows.

$$\lambda_k = \frac{I}{2}\left(\cos\left(\frac{k\pi}{n}\right)+1\right) \tag{18}$$

$$\mu_k = \frac{E}{2}\left(-\cos\left(\frac{k\pi}{n}\right)+1\right) \tag{19}$$

$$P_{\lambda_k} = \lambda_k / \sum \lambda_k \tag{20}$$

$$P_{\mu_k} = \mu_k / \sum \mu_k \tag{21}$$

where k is the number of species of the kth individual ($k = 1, 2, \ldots, K$), n is the maximum number of species that can be supported by the habitat ($n = K + 1$), I is the maximum possible immigration rate, E is the maximum possible emigration rate, P_{λ_k} is the probability of immigration of the kth individual (Pareto frontier), and P_{μ_k} is the probability of emigration of the kth individual. Note that the relationship between the value of species counts and the rank of Pareto frontiers are in reverse orders.

Mutation may not be a necessary operator in BBO (Simon, 2008). However, in this research, it is embedded in the main body of BBO to allow randomly modifications of some features in the solutions to occur so that more diversified solutions can be explored. In addition, mutation could provide opportunities for poor solutions to transform themselves to become good solutions abruptly.

4.2. Adaptive BBO for MMTSPLE

For global optimisation problems with complex search spaces, most original algorithms are often trapped in the local optima of the objective function. As a result, the algorithms could end up with premature convergence and the quality of the obtained solution is rather poor and not even comparable with the global optima. This issue is more obvious and challenging for large-sized multi-objective problems; hence, the modification of the original algorithms deems necessary. Consequently, in this research, an adaptive mechanism is embedded into the original BBO, called adaptive BBO (A-BBO), to mitigate the aforementioned issue. The detailed explanation of A-BBO is given as follows.

Mutation could be an operator that plays a significant role in BBO. Its control parameter, i.e. mutation probability (p_m), has a direct impact to the diversity of the population. A high value of (p_m) could generate more diversified solutions resulting in somehow preventing the algorithm from being stuck in the local optima. However, too high p_m could cause the algorithm to converge very slowly. Normally, p_m is assumed to be fixed throughout the entire run. Since the requirement of the algorithm is changed over time in terms of more or less exploration need, the value of p_m should not be fixed but adaptive to the current circumstance.

Local searches are widely coupled with EAs to further improve the quality of solutions. It is used to strengthen the exploitation capability of the original algorithm by systematically moving potential individuals directly towards the local optima or perhaps unknown global optima. In this research, 2-Opt is selected as a local search since it often achieves good results with less computation time in our pilot experiments and previous research (Kumar & Singh, 2007). Only the top rank solutions are randomly selected to take part in the local search exercise with probability p_{LS}. Similar to p_m, p_{LS} should not be fixed but changed over time according to the exploitation need.

The parametric adaptation by which p_m and p_{LS} are control parameters is applied in this research. The performance of the algorithm is evaluated periodically so that appropriate subsequence adaptive measures can bemade dynamically. The progressive convergence (PC) metric is used to assess the behaviour of the algorithm. PC is defined as the gap between the elite solutions of the current generation and those of the previous G generation. Eqs. (13) and (14) are used to computed the value of PC, where $PC \in [0,1]$. If PC is close to 0, the convergence is high meaning not much or no improvement in the elite solutions is achieved during the previous G generations; otherwise, if it is close to 1, the convergence is low meaning significant improvement is obtained.

The adaptive mechanism described by Herrera and Lozano (2003) is adopted as follows. In every G generations, the performance of the algorithm is monitored and evaluated through PC. Depending on the values of PC, an appropriate adaptive decision is triggered which may bring about altering the values of p_m and p_{LS} to the new ones, i.e. p'_m and p'_{LS}, if necessary, in order to tune the algorithm by increasing/decreasing its exploration and exploitation strengths in the next G generations. Our pilot studies indicated that the G value should be set between 7 and 10; hence, G is set at 10 to reduce the computation burden. The following adaptive rules as shown in Table 1 are employed to respond to the need of the algorithm. Note that these rules are derived based on our experience learned during the pilot experiments.

To help the algorithm to converge faster and reach better solutions, utilising templates in the offspring generating mechanism seems to be a fruitful enhancement (Tsutsui, 2006). A template is a building block (substructure) of a high quality solution which will be inherited to the offspring solution and high quality solutions normally share similar building blocks among them. Hence, it is logical to employ templates as a starting point for effective offspring generation.

The offspring generating mechanism with templates begins with randomly selecting a template solution from the elite solutions. A new offspring is created by duplicating a part of the template solution and the remaining positions are generated by using the normal BBO algorithm. To specify which part of the template solution will become the template, the template solution is rolled over to form a circle in which the first and last positions are joined together. Two cut points in the rolled template solution are selected at random. Now the template solution is spitted into two sections and one of them is chosen at random as the template. The pictorial demonstration of the mechanism is shown in Fig. 4. Watson and Pollack (2000) found that purely integrating templates into the solution generating mechanism of the problems with multiple solutions could sometimes mislead the search and cause a

Table 1
Adaptive rules for controlling p_m and p_{LS}.

Rules	Antecedent			Consequent	
	Initial probability		PC	Adjusted probability	
1	$p_m = 0.1$	$p_{LS} = 0.3$	$PC \leqslant 0.1$	$p'_m = 0.3$	$p'_{LS} = 0.1$
2	$p_m = 0.1$	$p_{LS} = 0.3$	$PC > 0.1$	$p'_m = 0.1$	$p'_{LS} = 0.3$
3	$p_m = 0.3$	$p_{LS} = 0.1$	$PC > 0.1$	$p'_m = 0.1$	$p'_{LS} = 0.3$
4	$p_m = 0.3$	$p_{LS} = 0.1$	$PC \leqslant 0.1$	$p'_m = 0.3$	$p'_{LS} = 0.1$

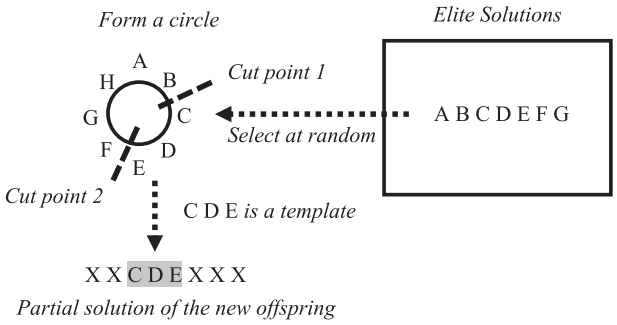


Fig. 4. Offspring generation by the template mechanism.

premature convergence. Hence, in this research, additional p_{tp}% of the number of offspring generated by BBO is included by using templates. The pseudo code for A-BBO is shown in Fig. 5.

4.3. Numerical example

Assume that three product models are intermixed in a 2SAL with MPS, (d_A, d_B, d_C), being $(1, 2, 3)$, and the position-based learning rate LR of 0.8 is assumed for setups and assembly operations. The setup time is sequence dependent and its value ranges between $U[0.05–0.1] \in R$ of the average task time. A six-bit integer is used to represent the mixed-model sequence as shown in Table 2.

Table 2
Encoding scheme.

Product model	A	B	B	C	C	C
Encoded bit number	1	2	3	4	5	6

Table 3
Initial solutions.

Solution number	Product mix sequence
1	3 5 1 2 6 4
2	2 6 3 1 4 5
3	1 2 5 3 4 6
4	3 6 1 5 4 2
5	4 1 5 2 6 3

Assume that the population size is 5; hence, five feasible sequences are generated in random as shown in Table 3.

Three objectives are computed for each sequence, i.e. the variance of production rate, the total utility work and the total setup time. The non-dominated sorting (Goldberg, 1989) is applied to the current population which results in three Pareto frontiers (fitness values) being found (Table 4). The value of species count on each Pareto frontier is determined by taking the reverse order of its corresponding Pareto frontier (Table 4) and the associated graph is shown in Fig. 6.

Adaptive BBO algorithm

Apply the non-dominated sorting approach to the population y
For each y_k, based on the Pareto frontier of y_k, find μ_k, λ_k, P_{λ_k} and P_{μ_k}
$z \leftarrow y$
For each z_k
  For each solution feature $z_k(s)$
    Use λ_k to decide whether to immigrate to $z_k(s)$
    If immigration is selected then
      Use μ_i of the frontier (roulette wheel selection) to select the emigrating frontier
        Randomly select solution y_j from the selected emigrating frontier
        $z_k(s) \leftarrow y_j(s)$
    End if
  Next solution feature
Next solution
Generate another p_{tp}% of z (z_{tp}) using the template mechanism
$z_k \leftarrow z_k + z_{tp}$
For each z_k
  Mutate z_k with probability p_m to z'_k
Next solution
Apply non-dominated sorting to the combined solutions of y, z and z'
  z'' is the first frontier of the combined solutions
For z'' of generation 1 and every G generations
  Apply 2-OPT local search with probability p_{LS} to z'''
Apply non-dominated sorting to the combined solutions of $z''$and z'''
  z'''' is the first front of the combined solutions
Update elitist list with z''''
$y \leftarrow z''''$
For every G generation
Compute PC
  Evaluate if the adaptation is needed
    If yes then
      Apply new p'_m and p'_{LS} for the next G generations
    Otherwise
      Continue using previous p_m and p_{LS} for the next G generations
  End if
Next generation

Fig. 5. Pseudo code for A-BBO.

Table 4
The fitness and crowding distance of each solution.

Solution number	Variance of production rate	Utility work	Setup time	Fitness	Species count
1	31.4556	15.6622	5.6877	2	2
2	40.1222	15.5409	5.6253	1	3
3	47.4556	16.9536	5.5354	1	3
4	22.6556	15.5363	5.6327	1	3
5	34.7889	15.8744	6.3763	3	1

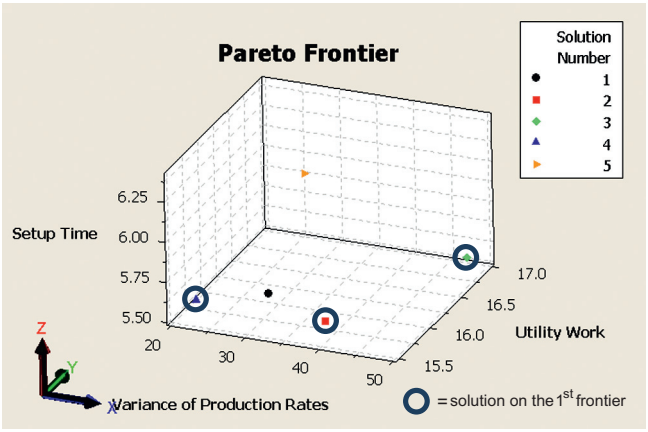


Fig. 6. Pareto frontier of the solutions.

Assume $E = I = 1$ and the sinusoidal migration model (Ma, 2010) is used, the values of λ_k, μ_k, P_{λ_k} and P_{μ_k} for each solution are shown in Table 5.

The migration process begins with copying the current population y to a temporary population z. For demonstration, assume that the temporary population (z_2) is selected, i.e. 263145. The values of $z_2(i)$ are 2, 6, 3, 1, 4 and 5, where $i = 1, \ldots, 6$, respectively. The immigration rate $z_2(\lambda_{z_2})$ is 0.1464 meaning that z_2 has a 14.64% chance of immigration. A number $r \in U[0,1]$ is randomly generated for each bit in $z_2(i)$ to determine whether the bit should be immigrated or not. If $r < \lambda_{z_2}$ for any bit, it will be immigrated; otherwise, no immigration occurs. Suppose $r = 0.1212 < \lambda_{z_2} = 0.1464$ for $z_2(4)$, immigration occurs to bit number 4 of z_2.

The immigration bit is determined from a roulette wheel selection based on emigration rates μ_k of species counts. Suppose that species count number 2 is selected. Since only one solution is in this species count, the solution number 1 with $y_1(4) = 2$ is selected. The value of a temporary bit $x = z_2(4) = 1$ and then $z_2(4) = y_1(4) = 2$. The migration process continues to the end bit of z_2 which results in the new solution of 263245. Since this solution is unfeasible, the repair process is called upon. The bit of z_2 which now takes the value of 2 (i.e. $z_2(1)$) has to replace its value with the value of the temporary bit x. Therefore, the repaired solution of z_2 is 163245. The new solutions after migration are shown in Table 6.

Let p_{tp} is 0.2 but no elite solution is found for the first generation; hence the template mechanism is not triggered. Assume that the initial value of p_m is 0.1 and the reciprocal exchange method

Table 6
Solutions after the migration process.

Solution number	Product mix sequence
1	2 5 1 3 6 4
2	1 6 3 2 4 5
3	4 2 5 3 1 6
4	3 6 1 5 4 2
5	5 1 3 2 6 4

(Goldberg, 1989) is used as the mutation operator. The number of solutions involved in the mutation is $0.1*5 = 0.5 \approx 1$. Assume that the solution 513264 is randomly selected, its bits 2 and 4 are selected in random to take part in the mutation, and consequently the new solution after mutation is 523164.

The current population is combined with those obtained after migration and mutation, and the replicated solutions are deleted from the combined solutions which result in 10 remaining solutions as shown in Table 7. The objectives for each solutions are computed and non-dominated sorting is applied to them which results in 5 solutions being on the first frontier (fitness = 1).

Since the generation number is 1, local search is applied to the solutions on the first frontier. Assume that the initial value of p_{LS} is 0.3 and 2-OPT is used as the local search operator. The number of solutions to be involved in local search is $5 * 0.3 = 1.5 \approx 2$. Assume that the solutions 163245 and 513264 are selected and their solutions after local search are 162345 and 462315, respectively. These two solutions are combined with five solutions on the first frontier, then non-dominated sorting is applied to these solutions, and five solutions are found on the new first frontier (Table 8).

To prevent the loss of desirable features occurred during the evolution process and maintain the best solutions in the population, elitism is applied. The final solutions on the first frontier of the current population are combined with the solutions in the elite list (initially empty). Non-dominated sorting is applied to the combined solutions and the solutions on the first frontier are kept in the new elite list. Since the generation number is 1, the adaptive mechanism to adjust the values of p_m and p_{LS} is not triggered and then the next generation is evolved.

Assume that now the algorithm progresses to the end of the 5th generation, the adaptive mechanism is triggered in every five generation (i.e. $G = 5$) and the non-dominated solutions kept in the elite list after completing the 5th generation are shown in Table 9. It is observed that the number of non-dominated solutions increases from 5 (in the 1st generation) to 10 (in the 5th generation). To find an appropriate adaptive decision, the progressive

Table 5
Probability of migration for each solution.

Solution number	String number	Species count	λ_k	μ_k	P_λ	P_μ
2	y_2	3	0.1464	0.8536	0.0976	0.5690
3	y_3					
4	y_4					
1	y_1	2	0.5000	0.5000	0.3333	0.3333
5	y_5	1	0.8536	0.1464	0.5690	0.0976

Table 7
Combined solutions.

Solution number	Group	Product mix sequence	Variance of production rate	Utility work	Setup time	Fitness
1	Parent	3 5 1 2 6 4	31.4556	15.6622	5.6877	2
2		2 6 3 1 4 5	40.1222	15.5409	5.6253	2
3		1 2 5 3 4 6	47.4556	16.9536	5.5354	2
4		3 6 1 5 4 2	22.6556	15.5363	5.6327	1
5		4 1 5 2 6 3	34.7889	15.8744	6.3763	3
6	Offspring	2 5 1 3 6 4	31.4556	15.6622	5.6877	2
7		1 6 3 2 4 5	41.7222	16.4929	4.7298	1
8		4 2 5 3 1 6	22.5222	15.1276	6.4237	1
9		5 1 3 2 6 4	34.3889	15.8214	4.8248	1
10		5 2 3 1 6 4	37.0556	15.4733	4.8227	1

Table 8
Solutions of the 1st generation after local search.

Solution number	Product mix sequence	Variance of production rate	Utility work	Setup time
1	1 6 3 2 4 5	41.7222	16.4929	4.7298
2	4 2 5 3 1 6	22.5222	15.1276	6.4237
3	1 6 2 3 4 5	41.7222	16.4929	4.7298
4	4 6 2 3 1 5	34.3889	14.4735	4.8227
5	3 6 1 5 4 2	22.6556	15.5363	5.6327

Table 9
Updated solutions kept in the elite list in the 5th generation.

Solution number	Product mix sequence	Variance of production rate	Utility work	Setup time
1	6 3 5 1 4 2	19.5889	15.2717	6.4344
2	4 5 6 2 1 3	70.9222	14.0759	4.7132
3	6 2 1 4 5 3	19.5889	15.8083	5.6973
4	5 6 2 4 1 3	44.1222	14.4168	5.6001
5	6 5 2 3 1 4	34.3889	14.4735	4.8227
6	2 6 5 1 4 3	22.6556	14.7049	5.6327
7	1 5 2 3 6 4	41.7222	16.4929	4.7298
8	6 2 5 3 1 4	22.5222	15.1276	4.4237
9	3 5 6 2 1 4	25.5889	14.6662	5.6253
10	5 3 6 4 1 2	32.2556	14.6600	5.6001

Table 10
Test-bed problems.

Problem		String length	MPS	Feasible solutions
Small	S1	12	5:3:2:1:1	3.326E+05
	S2	12	4:4:2:1:1	4.158E+05
	S3	15	7:3:2:2:1	1.081E+07
	S4	15	4:3:3:3:2	1.261E+08
	S5	15	3:3:3:3:3	1.682E+08
Medium	M1	20	8:7:2:2:1	2.993E+09
	M2	20	5:4:4:4:3	2.444E+11
	M3	20	7:5:1:1:1:1:1:1:1:1	4.023E+12
	M4	20	4:4:4:2:1:1:1:1:1:1	2.112E+15
	M5	20	7:3:4:6	4.655E+09
	M6	30	15:15	1.551E+08
	M7	30	5:7:8:10	2.997E+15
Large	L1	100	20:20:20:15:15:1:1:1:1:1:1:1:1:1:1:1	3.790E+78
	L2	100	20:20:15:15:10:6:6:1:1:1:1:1:1:1:1:1	4.901E+84
	L3	100	15:15:15:10:10:10:10:5:4:1:1:1:1:1:1:1	8.357E+91
	L4	100	15:15:10:10:10:10:10:10:4:1:1:1:1:1:1:1	9.959E+92
	L5	100	7:7:7:7:7:7:7:7:7:6:6:6:6:6	4.560E+106

convergence metric (PC) is computed using Eqs. (13) and (14) and its value is 0.441986. The adaptive rules as shown in Table 1 are sequentially checked. It is found that Rule 2 matches the antecedence of the current conditions ($p_m = 0.1$, $p_{LS} = 0.3$ and $PC > 0.1$); hence, the values of p_m and p_{LS} to be used in the next G generations are 0.1 and 0.3 respectively. The next generation is then started.

5. Experimental design and results

The performance of A-BBO in solving the MMTSPLE problems is benchmarked against four well-known algorithms, i.e. BBO (Simon, 2008), NSGA II (Deb, Pratap, Agarwal, & Meyarivan, 2002), DPSO (Kennedy, Eberhart, & Shi, 2001) and PSONK (Chutima & Chimklai,

Table 11
Parameter settings.

Parameter settings	NSGA II	DPSO	PSONK	BBO	A-BBO
Population size	100	100	100	100	100
Number of swarm & number of particles in each swarm		S1: 10& 10 S2: 5& 20 S3: 10& 10 S4: 10& 10 S5: 10& 10 M1: 10& 10 M2: 10& 10 M3: 4& 25 M4: 4& 25 M5: 4& 25 M6: 4& 25 M7: 4& 25 Ll: 5& 20 L2: 5& 20 L3: 5& 20 L4: 5& 20 L5: 5& 20	S1: 5& 20 S2: 10& 10 S3: 5& 20 S4: 4& 25 S5: 4& 25 Ml: 4& 25 M2: 10& 10 M3: 4& 25 M4: 10& 10 M5: 4& 25 M6: 4& 25 M7: 10& 10 Ll: 4& 25 L2: 5& 20 L3: 5& 20 L4: 5& 20 L5: 5& 20		
Crossover method	Weight mapping				
Mutation method	Reciprocal exchange			Reciprocal exchange	Reciprocal exchange
Pc	S1: 0.1 S2: 0.2 S3: 0.1 S4: 0.2 S5: 0.2 M1: 0.3 M2: 0.3 M3: 0.1 M4: 0.4 M5: 0.3 M6: 0.3 M7: 0.1 L1: 0.2 L2: 0.3 L3: 0.4 L4: 0.4 L5: 0.4				
Pm	S1: 0.1 S2: 0.2 S3: 0.1 S4: 0.2 S5: 0.2 M1: 0.3 M2: 0.3 M3: 0.1 M4: 0.4 M5: 0.3 M6: 0.2 M7: 0.1 L1: 0.2 L2: 0.3 L3: 0.3 L4: 0.3 L5: 0.3			0.01	0.1,0.3
Local search method					2-OPT
Ptp					0.2
Pis					0.1, 0.3
Learning factors		0.1	0.1		
Inertia weight		0.1	0.1		
Migration model				Sinusoidal	Sinusoidal

2012). The reason that NSGA II and DPSO were selected in this research was because they were among the original versions of high performance algorithms which could act as a good baseline in performance comparison with the original version of BBO. The problem set, parameters used in each algorithm and interesting experimental results are elaborated in the following sections.

5.1. MMTSPLE test-bed problems

The effectiveness of the proposed algorithm is evaluated with 17 MMTSPLE problems having various characteristics ranging from small to large sizes, different numbers of product models, MPSs and learning rates (Table 10). These problems are adopted from

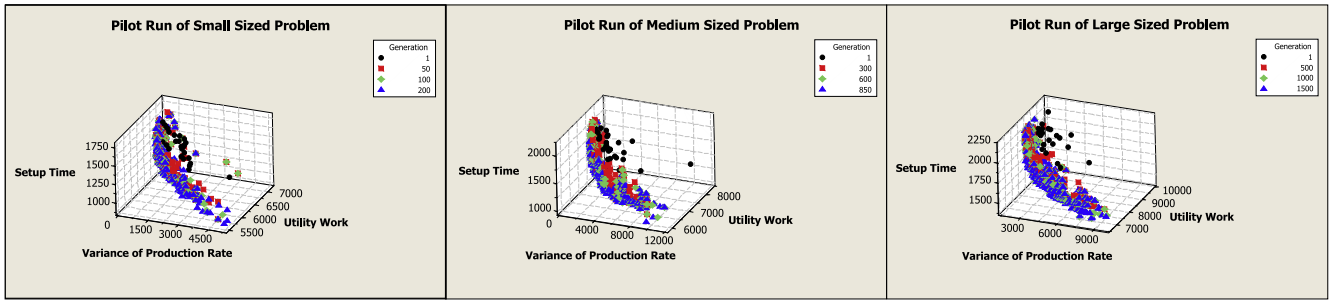


Fig. 7. Examples of pilot experiments on various sizes of problems.

well-known problems published in Bartholdi (1993), Lee et al. (2001) and McMullen (2001a). It is worth noting that the selected problems are those that have large search spaces in order not to restrain algorithms from demonstrating their effective strengths. Moreover, real data obtained from a leading European car manufacturer in Thailand (Subudom, 2009), i.e. Problem M6, are also used to demonstrate the industrial applicability of the proposed approach. Hence, five small-sized problems (S1–S5 with 12 and 15 string lengths), seven medium-sized problems (M1–M7 with 20 and 30 string lengths) and five relatively large-sized problems (L1–L5 with 100 string length) constitute this data set. The size of the search space (i.e. number of feasible solutions) for each problem is also computed to demonstrate its combinatorial complexity (Hyun et al., 1998). It is obvious that the number of feasible solutions increases exponentially as the problem size increases; hence, multi-objective optimal solutions seem unachievable for large-sized problems. It is assumed that the position-based learning rates *LR* of 80% and 90% are imposed on setups and assembly operations. In addition, the setup time is sequence dependent and its value ranges between $U[0.05–0.1] \in R$ of the average task time (Lummus, 1995). Note that before conducting the search for the solutions of the MMTSPLE problems, the associated assembly line balancing problems are assumed to be solved resulting in known number of stations and configuration of task assignment.

5.2. Parameter settings

Since the characteristic of each problem is different, the design and analysis of experiments (Montgomery, 2013) are extensively conducted on the sets of parameters to fine tune all tested algorithms to best suit the given problem. The parameters governing the behaviour of the algorithm towards high performance cited in pervious literature are used as a starting point for selecting suitable levels of them to be experimented. Table 11 summarises the parameters found to be effective in terms of solution quality for each algorithm.

Pilot experiments are conducted to investigate the approximated number of generations that the algorithms tend to terminate. Graphical plots of non-dominated solutions on the domain of three objective functions are exemplified in Fig. 7. It is obvious that the initial solutions of the first generation are rather dispersed and floated far apart from the origin where the axes of three objective functions intersect. The non-dominated solutions gradually head down towards to the origin as the number of generations increases. After passing a great number of generations, the numbers and shape of non-dominated solutions are unaltered and then the evolution process is ceased.

To allow each algorithm to fully search and prevent premature convergence, a significantly greater number of generations than the one founded in the pilot experiments are used. Hence, the terminating conditions of all algorithms are set at 1500 generations for small- and medium-sized problems and 2000 generations for large-sized problems, or when no improvement has been notified for 30 consecutive generations.

For every experimental problem, each algorithm is replicated only twice to obtain an experimental error estimate due to lengthy computational time, especially for large-sized problems. Obviously, the Pareto-frontier evaluation (non-dominated sorting) step

Table 12
Performances of the algorithms on small-sized problems with 80% learning rate.

Efficiency	Algorithm	Problems					Average	Rank
		S1	S2	S3	S4	S5		
Convergence	NSGAII	0.030367	0.029634	0.029889	0.050240	0.048111	0.037648	4
	DPSO	0.028561	0.054826	0.044233	0.071336	0.075547	0.054901	5
	PSONK	0.016567	0.019491	0.026143	0.039538	0.027542	0.025856	2
	BBO	0.029501	0.038294	0.024479	0.024447	0.034261	0.030197	3
	A-BBO	0.009538	0.009144	0.009037	0.012317	0.013758	0.010759	1
Spread	NSGAII	0.573608	0.697061	0.712418	0.682207	0.701924	0.673444	5
	DPSO	0.560583	0.613627	0.621042	0.612747	0.612746	0.604149	2
	PSONK	0.583042	0.627881	0.593260	0.619952	0.620436	0.608914	3
	BBO	0.503287	0.583247	0.605315	0.638524	0.577626	0.581600	1
	A-BBO	0.615324	0.658114	0.604318	0.668988	0.623009	0.633951	4
R_{NDS}	NSGAII	0.243986	0.234501	0.070761	0.022329	0.025569	0.119429	4
	DPSO	0.065292	0.048518	0.045394	0.006380	0.001827	0.033482	5
	PSONK	0.295533	0.315364	0.206943	0.154705	0.298837	0.254276	2
	BBO	0.302405	0.229111	0.260347	0.293461	0.166463	0.250357	3
	A-BBO	0.604811	0.630728	0.531375	0.622010	0.584141	0.594613	1
CPU time	NSGAII	304,920	366,274	563,328	234,640	355,781	364,989	5
	DPSO	8282	2585	15,990	6794	7265	8183	1
	PSONK	24,475	47,370	260,559	132,634	135,353	120,078	3
	BBO	5548	11,159	21,235	12,846	13,542	12,866	2
	A-BBO	163,714	306,096	369,272	205,255	243,812	257,630	4

Table 13
Performances of the algorithms on medium-sized problems with 80% learning rate.

Efficiency	Algorithm	Problems							Average	Rank
		M1	M2	M3	M4	M5	M6	M7		
Convergence	NSGAII	0.0553686	0.0628893	0.0983772	0.0791980	0.1050270	0.167632	0.1471980	0.1022414	4
	DPSO	0.0579040	0.0979858	0.1032226	0.1386045	0.1243750	0.271557	0.1536920	0.1353344	5
	PSONK	0.0550810	0.0420817	0.0232459	0.0369215	0.0162430	0.062641	0.0172720	0.0362123	2
	BBO	0.0279825	0.0349835	0.0211307	0.0285486	0.0544550	0.114979	0.0554910	0.0482243	3
	A-BBO	0.0191767	0.0143409	0.0076732	0.0111272	0.0096140	0.016907	0.0159110	0.0135357	1
Spread	NSGAII	0.5892490	0.6193950	0.5776780	0.6687730	0.5898440	0.667013	0.5738050	0.6122510	4
	DPSO	0.6623030	0.5784940	0.6009460	0.4789100	0.4983940	0.632930	0.5846480	0.5766607	1
	PSONK	0.6027860	0.6829570	0.5746360	0.5515780	0.5136070	0.569998	0.5960490	0.5845159	2
	BBO	0.5706750	0.5742280	0.6946020	0.6141220	0.5420290	0.663937	0.5960190	0.6079446	3
	A-BBO	0.6315110	0.6997460	0.7180910	0.6592570	0.6254400	0.750438	0.7258960	0.6871970	5
R_{NDS}	NSGAII	0.012481	0.020800	0.002353	0.024173	0.006494	0.000000	0.000000	0.0094714	4
	DPSO	0.004680	0.001600	0.001177	0.001272	0.003247	0.000000	0.000000	0.0017108	5
	PSONK	0.658346	0.192000	0.052941	0.195929	0.444805	0.393939	0.396419	0.3334828	2
	BBO	0.099844	0.291200	0.443529	0.232824	0.029221	0.030303	0.058824	0.1693922	3
	A-BBO	0.224649	0.494400	0.500000	0.545802	0.568182	0.575758	0.544757	0.4933639	1
CPU time	NSGAII	493,125	284,080	992,000	295,525	323,450	61,236	471,500	417,274	5
	DPSO	22,605	9491	5976	6816	5736	3052	9193	8981	1
	PSONK	189,452	21,744	119,132	75,213	38,339	9289	206,473	94,235	3
	BBO	106,253	91,345	160,865	146,784	19,214	5374	98,564	89,771	2
	A-BBO	221,097	250,292	541,872	277,737	283,171	50,846	443,578	295,513	4

consumes most of the computation time since three objective functions have to be optimised simultaneously in this research. In addition, as the number of non-dominated solutions in the elitist list grows (i.e. during the end of the run) the computation time is greatly intensified. All algorithms were coded in MATLAB R2009a and all experiments were conducted on a personal computer with Intel (R) Core™i7 CPU2670 2.20 GHz and 4 GB RAM using Microsoft Windows 7. While coding all the algorithms, care has been taken to reduce as much bias as possible in the computational comparison.

5.3. Comparative results

The proposed A-BBO algorithm is applied to the test-bed problems and its performance is compared against rival algorithms, i.e. NSGA II, DPSO, PSONK and BBO. Tables 12–17 show comparative results obtained from these algorithms under various complexities and two learning rates (*LR*) of 80% and 90%. The experimental results are presented and analysed with respect to four performance metrics as mentioned previously. The results are provided in average values for each problem.

5.3.1. Solution quality

The relative solution quality of different algorithms is benchmarked through two metrics including convergence and the ratio of the obtained non-dominated solutions (R_{NDS}). The convergence metric measures the difference between the obtained non-dominated solution set of the given algorithm and the approximated true-Pareto set, and the lower value is the better. Tables 12–17 show clearly that the proposed A-BBO, the adaptive version of BBO, is superior to its original one. Specifically, A-BBO achieves substantially lower convergence metrics than BBO in every test

Table 14
Performances of the algorithms on large-sized problems with 80% learning rate.

Efficiency	Algorithm	Problems					Average	Rank
		L1	L2	L3	L4	L5		
Convergence	NSGAII	0.178978	0.154032	0.155615	0.189429	0.221673	0.179945	4
	DPSO	0.283493	0.345828	0.296980	0.210281	0.392168	0.305750	5
	PSONK	0.068585	0.116360	0.126273	0.107609	0.133072	0.110380	2
	BBO	0.111535	0.138372	0.142955	0.100722	0.176497	0.134016	3
	A-BBO	0.014317	0.017089	0.007408	0.019534	0.008730	0.013416	1
Spread	NSGAII	0.649678	0.552981	0.592227	0.688265	0.601103	0.616851	4
	DPSO	0.532721	0.544961	0.559134	0.533091	0.593597	0.552701	3
	PSONK	0.517488	0.491373	0.514304	0.521020	0.541034	0.517044	1
	BBO	0.508776	0.539977	0.544992	0.553621	0.581098	0.545693	2
	A-BBO	0.660116	0.684777	0.673512	0.781543	0.712546	0.702499	5
R_{NDS}	NSGAII	0.000000	0.005911	0.000000	0.002310	0.000460	0.001736	4
	DPSO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5
	PSONK	0.144366	0.153371	0.038851	0.173210	0.052387	0.112437	2
	BBO	0.010563	0.081796	0.004178	0.027714	0.005552	0.025961	3
	A-BBO	0.845070	0.759171	0.957573	0.796767	0.941818	0.860080	1
CPU time	NSGAII	317,175	324,156	315,126	302,050	321,684	316,038	5
	DPSO	12,513	15,321	16,132	22,184	20,156	17,261	1
	PSONK	117,154	118,235	115,324	106,501	125,314	116,506	3
	BBO	41,945	43,952	46,251	46,171	48,224	45,309	2
	A-BBO	293,166	295,123	281,569	290,820	298,156	291,767	4

Table 15
Performances of the algorithms on small-sized problems with 90% learning rate.

Efficiency	Algorithm	Problems					Average	Rank
		S1	S2	S3	S4	S5		
Convergence	NSGAII	0.040849	0.023759	0.037127	0.054677	0.057139	0.042710	4
	DPSO	0.043611	0.037485	0.048147	0.073030	0.098545	0.060163	5
	PSONK	0.021799	0.012242	0.014953	0.026121	0.032490	0.021521	2
	BBO	0.031603	0.022430	0.038530	0.051963	0.040866	0.037078	3
	A-BBO	0.007102	0.002970	0.002709	0.003892	0.028333	0.009001	1
Spread	NSGAII	0.696545	0.890552	0.782325	0.851672	0.596515	0.763522	4
	DPSO	0.676799	0.766487	0.806877	0.641906	0.512772	0.680968	1
	PSONK	0.774998	0.861226	0.795220	0.649296	0.562948	0.728738	2
	BBO	0.792455	0.980778	0.752656	0.652153	0.575859	0.750780	3
	A-BBO	0.736103	0.914140	0.864173	0.738269	0.609590	0.772455	5
R_{NDS}	NSGAII	0.117647	0.127168	0.004274	0.008811	0.086331	0.068846	4
	DPSO	0.050420	0.017341	0.008547	0.000000	0.000000	0.015262	5
	PSONK	0.403361	0.381503	0.205128	0.149780	0.368846	0.301724	2
	BBO	0.462185	0.202312	0.051282	0.035242	0.189053	0.188015	3
	A-BBO	0.563025	0.728324	0.807692	0.854626	0.396362	0.670006	1
CPU time	NSGAII	457,036	143,040	247,068	261,065	282,535	278,149	5
	DPSO	2097	1572	6529	4735	5162	4019	1
	PSONK	11,685	6186	4312	7702	8934	7764	3
	BBO	2831	10,040	3788	5873	6189	5744	2
	A-BBO	319,455	126,021	244,364	256,304	271,235	243,476	4

Table 16
Performances of the algorithms on medium-sized problems with 90% learning rate.

Efficiency	Algorithm	Problems							Average	Rank
		M1	M2	M3	M4	M5	M6	M7		
Convergence	NSGAII	0.077650	0.137311	0.064698	0.077855	0.149581	0.139030	0.121239	0.109623	4
	DPSO	0.089895	0.131439	0.112087	0.170259	0.161054	0.179133	0.232842	0.153816	5
	PSONK	0.014155	0.021175	0.013637	0.019770	0.022555	0.049523	0.026506	0.023903	2
	BBO	0.072330	0.076649	0.034950	0.022913	0.049687	0.090365	0.073215	0.060015	3
	A-BBO	0.004096	0.020809	0.005417	0.005792	0.013258	0.038566	0.004808	0.013249	1
Spread	NSGAII	0.697877	0.730272	0.725851	0.896156	0.724376	0.790496	0.801092	0.766589	4
	DPSO	0.627887	0.561411	0.532933	0.523627	0.562928	0.546402	0.656482	0.573096	1
	PSONK	0.697911	0.526661	0.783828	0.699968	0.718967	0.548763	0.656263	0.661766	2
	BBO	0.498222	0.587505	0.863664	0.755901	0.719145	0.587078	0.848700	0.694316	3
	A-BBO	0.725341	0.748853	0.937582	0.809688	0.706795	0.905863	0.855420	0.812792	5
R_{NDS}	NSGAII	0.016556	0.003759	0.017588	0.005076	0.028571	0.086957	0.000000	0.022644	4
	DPSO	0.003311	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000473	5
	PSONK	0.324503	0.383459	0.201005	0.065990	0.417143	0.173913	0.094828	0.237263	2
	BBO	0.029801	0.018797	0.206030	0.365482	0.057143	0.173913	0.051724	0.128984	3
	A-BBO	0.649007	0.597744	0.575377	0.565144	0.502857	0.565217	0.853448	0.615542	1
CPU time	NSGAII	220,162	144,120	147,176	229,600	199,276	21,756	226,502	169,799	5
	DPSO	5646	4000	2632	3969	4293	3795	9981	4902	1
	PSONK	49,024	29,581	36,095	43,842	28,742	16,046	36,999	34,333	3
	BBO	9250	2714	9575	26,428	15,842	10,175	40,126	16,301	2
	A-BBO	172,519	140,646	146,452	250,139	199,385	18,528	170,033	156,815	4

case. These observations suggest that the adaptive mechanism of A-BBO enables the non-dominated solutions of BBO escaping from local optimal areas and moving towards global ones. It is observed that the performance gap between A-BBO and BBO tends to be widened for large-sized problems with the higher learning rate.

The R_{NDS} metric indicates the coverage of the non-dominated solution set provided by the given algorithm over the approximated true-Pareto set. In other words, this ratio signifies the number of the non-dominated solutions getting out of the given algorithm belonging to the approximated true-Pareto solutions, and the higher ratio is the better. It is obvious that A-BBO demonstrates its superiority over BBO on the R_{NDS} metric by providing higher ratios as shown in Tables 12–17. Similar to the convergence metric, their significant differences are even pronounced in large-sized problems.

In comparison with the other algorithms (i.e. NSGA II, DPSO and PSONK) with respect to the convergence and R_{NDS} metrics, the obtained performances of each algorithm in each problem's size are averaged and ranked. The results are shown in the last two columns of Tables 12–17. Based on the average convergence and R_{NDS} metrics, A-BBO is always ranked first, followed by PSONK, NSGA II and DPSO, respectively. On average, this means that A-BBO provides Pareto frontiers which are not only closer to the approximated true-Pareto frontiers (i.e. lower convergence metric), but also produces more numbers of non-dominated solutions located on the approximated true-Pareto frontiers (i.e. higher R_{NDS} metric) than its contestant algorithms. Substantial performance differences in both metrics are noticeable in large-sized problems with the high learning rate. These results suggest that the quality of solutions provided by A-BBO is con-

Table 17
Performances of the algorithms on large-sized problems with 90% learning rate.

Efficiency	Algorithm	Problems					Average	Rank
		L1	L2	L3	L4	L5		
Convergence	NSGAII	0.225746	0.127428	0.250650	0.247143	0.228077	0.215809	4
	DPSO	0.426698	0.321105	0.379868	0.372212	0.350239	0.370024	5
	PSONK	0.061326	0.088699	0.184554	0.121199	0.149246	0.121005	2
	BBO	0.096180	0.116585	0.238155	0.247582	0.203664	0.180433	3
	A-BBO	0.003151	0.016336	0.000097	0.001510	0.012189	0.006657	1
Spread	NSGAII	0.721579	0.610626	0.662258	0.757602	0.635211	0.677455	4
	DPSO	0.565647	0.546956	0.562631	0.594691	0.587808	0.571547	3
	PSONK	0.560768	0.527782	0.552963	0.527974	0.533731	0.540644	2
	BBO	0.558710	0.513280	0.546738	0.468354	0.519625	0.521341	1
	A-BBO	0.745046	0.675473	0.692130	0.768769	0.718602	0.720004	5
R_{NDS}	NSGAII	0.000000	0.001451	0.000000	0.000000	0.000000	0.000290	4
	DPSO	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5
	PSONK	0.032258	0.128794	0.001268	0.005195	0.055482	0.044599	2
	BBO	0.006452	0.006071	0.000000	0.000000	0.000000	0.002505	3
	A-BBO	0.961290	0.864513	0.998863	0.994805	0.944644	0.952823	1
CPU time	NSGAII	266,980	262,110	272,023	263,360	281,651	269,225	5
	DPSO	3510	2595	31,102	19,243	22,416	15,773	1
	PSONK	138,682	162,534	172,564	119,936	161,531	151,049	3
	BBO	35,509	34,621	38,012	42,935	45,202	39,256	2
	A-BBO	252,583	242,358	255,122	250,287	261,233	252,317	4

sistently better than the other algorithms reflecting the robustness of A-BBO.

With a holistic view, BBO and PSONK achieve better performances than NSGA II and DPSO. The algorithm with the poorest performance seems to be DPSO which obtains very few or in some cases of large-sized problems have never produced any solution located on the approximated true-Pareto solutions at all. In addition, apart from widening the gap differences more clearly in the high learning rate, different learning rates of 80% and 90% have no significant effect on the relative performances of the algorithms in terms of the solution quality since the harmonious performance patterns are manifested for both percentages of the learning rate. Although the performance of the original versions of DPSO and NSGA II are not good in this research, the other new variants of both algorithms (e.g. controlled elitist NSGA II) might worth considering as benchmark algorithms in the future research.

5.3.2. Diversity of solutions

The spread metric measures the diversity of the obtained non-dominated solutions by computing a relative distance between consecutive solutions, and the lower value is the better. As shown in the last two columns of Tables 12–17, on average, all algorithms seem to provide indifferent spread metric. This is not surprising since all algorithms employ the crowding distance approach (Deb et al., 2002) as the diversity preservation technique. More importantly, this metric should be considered in conjunction with the convergence and R_{NDS} metrics since it is not useful if the obtained non-dominated solutions are uniformly distributed but most of them are not on the approximated true-Pareto frontier. In other words, higher priority should be given to the convergence and R_{NDS} metrics than the spread metric. However, the average values of the spread metric in A-BBO are slightly higher than the other algorithms meaning that it could find non-dominated solutions with less scattered in comparison with the counterparts.

5.3.3. Time to solutions

Although it should not be considered as a major performance measurement, the computational time (CPU) of each algorithm is reported. As illustrated in Tables 12–17, on average, A-BBO consumes substantially higher computational time than its original version (BBO). For instance, in large-sized problems, A-BBO consumes about six times on average more than the computational time required by BBO. The higher value of computational time is rational since the adaptive mechanism is further embedded into the standard structure of BBO to become A-BBO. Although its computational time is higher than BBO, the ability of A-BBO in searching intelligently through potential areas of the search space with low chance of trapped on a local optimal to yield good non-dominated solutions seems to be much more beneficial. Furthermore, A-BBO always converges to better non-dominated solutions, which means it is much more effective in exploitation of the solution space than BBO.

It is observed that among five contestant algorithms A-BBO comes second in terms of highest computation time to reach the final solutions; whereas the first and last are NSGA II and DPSO, respectively. In addition, as the problem complexity increases, the computational time tends to be increased. If we want to classify the algorithms into groups based on the computational time requirement, it should be as follows: (1) NSGA II and A-BBO (high), (2) PSONK (moderate), and (3) BBO and DPSO (low).

Although the time to solutions of A-BBO seems to be an obvious drawback, its performances defeat other competitive algorithms completely in terms of the ability to achieve the Pareto-optimal solutions measured by convergence and R_{NDS} metrics. The adaptive mechanism of A-BBO could effectively monitor when the exploitation and exploration in the evolution process of the algorithm is needed. Specifically, when the non-dominated solutions start to converge or when the search is stagnated, A-BBO switches its role from exploitation to exploration by adjusting the controlled parameters to reduce the possibility of the solutions to be trapped in the local optima, bringing about achieving significantly better non-dominated solutions especially for large-sized problems. It is also observed that A-BBO produces much more alternative sequences along the approximated true-Pareto frontier than the other algorithms. This gives more chance and flexibility for the decision makers to decide on the desired sequence. Considering the solution quality and computational time trade off of A-BBO, the benefit definitely outweighs such trivial aforementioned drawback. It is worth mentioning that if a quick and fairly good performance algorithm is needed to be chosen as a benchmark algorithm, the conventional BBO algorithm should be considered as one of the potential options.

6. Conclusion and future work

In a mixed-model assembly line, joint production of diversified product portfolios is produced in an appropriate intermixed sequence to realise an efficient flow-line production and response to customers' preference. Determining the optimal sequence of the product mix to be assembled at such flexible line is challenging and recognised as an important operational decision for improving the production performance.

In this paper, a multi-objective sequencing problem for 2SALs with a learning effect is studied. Three conflicting objectives are optimised simultaneously including minimising the variance of production rate, minimising the total utility work and minimising the total sequence-dependent setup time. Since the problem is classified in an NP-hard type, traditional optimisation techniques are unlikely practical, especially for large-sized problems.

A-BBO, an instance of meta-heuristic, is developed aiming at achieving high quality non-dominated solutions. In A-BBO, a promising hybridisation mechanism is further embedded into the original version of BBO in such a way that they could harmoniously work together through monitoring the feedback status of the evolving population and triggering necessary adaptive commands to adjust the controlled parameters of the algorithm. With this approach, the exploitation and exploration capabilities of A-BBO are able to utilise appropriately in the time of need so that the shortfall of premature convergence is avoided.

To compare the performance of A-BBO against the well-known algorithms, i.e. NSGA II, DPSO, PSONK and BBO, four comparison metrics are used including convergence, R_{NDS}, spread metrics and computational time. The results indicate clearly that, apart from high computation time which may be considered as a trivial concern, A-BBO outperforms all other competitive algorithms in terms of solution quality. The success of A-BBO is due to the trade-off between the exploration ability of the underlying BBO and the exploitation ability of the local search embedded in an adaptive mechanism being utilised in an effective manner. Hence, A-BBO should be viable and considered as a potential benchmark algorithm for MMTSPLE in the future.

Some areas worth considering as an extension of this research are as follows: (i) inclusion of stochastic task times occurred due to human factors (i.e. instability of human operator) in terms of non-uniform work rates, fatigue, skill, etc. (Agrawal & Tiwari, 2008); (ii) investigation of other improved versions of NSGA II, e.g. fast-elitist NSGA II, controlled elitist NSGA II (Manupati, Thakkar, Wong, & Tiwari, 2013); (iii) benchmark against other meta-heuristics like territory defining evolutionary algorithm, ant colony optimisation algorithm, artificial bee colony algorithm, etc.; and (iv) simultaneous consideration of the interrelated line balancing and model sequencing problems.

References

Agrawal, S., & Tiwari, M. K. (2008). A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research, 6*(15), 1405–1429.

Andrés, C., Miralles, C., & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research, 187*(3), 1212–1223.

Bard, J. F., Shtub, A., & Joshi, S. B. (1994). Sequencing mixed-model assembly lines to level parts usage and minimize the length. *International Journal of Production Research, 32*(10), 2431–2454.

Bartholdi, J. J. (1993). Balancing two-sided assembly lines: a case study. *International Journal of Production Research, 31*(10), 2447–2461.

Baykasoglu, A., & Dereli, T. (2008). Two-sided assembly line balancing using ant-colony-based heuristic. *International Journal of Advanced Manufacturing Technology, 36*(5-6), 582–588.

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research, 115*, 173–178.

Biskup, D. (2008). A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research, 188*, 315–329.

Boysen, N., Fliedner, M., & Scholl, A. (2009). Sequencing mixed-model assembly lines: Survey, classification, and model critique. *European Journal of Operational Research, 192*(2), 349–373.

Chutima, P., & Chimklai, P. (2012). Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers and Industrial Engineering, 62*(1), 39–55.

Chutima, P., & Jitmetta, K. (2013). Adaptive biogeography-based optimisation for two-sided assembly line sequencing problems. *International Journal of Operational Research, 16*(4), 390–420.

Chutima, P., & Pinkoompee, P. (2009). Multi-objective sequencing problems of mixed-model assembly systems using memetic algorithms. *Science Asia, 35*, 295–305.

Coello, C. A. Coello (1999). A comprehensive survey of evolutionary-based multi-objective optimization techniques. *Knowledge and Information Systems, 1*(3), 269–308.

Coello, C. A. Coello (2006). 20 years of evolutionary multi-objective optimization: what has been done and what remains to be done. In Gary Y. Yen & David B. Fogel (Eds.), *Computational intelligence: Principles and practice* (pp. 73–88). IEEE Computational Intelligence Society.

Cortés, P., Onieva, L., & Guadix, J. (2010). Optimising and simulating the assembly line balancing problem in a motorcycle manufacturing company: A case study. *International Journal of Production Research, 48*(12), 3637–3656.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Goldberg, D. E. (1989). *Genetic algorithm in search, optimization, and machine learning*. Reading, MA: Addison-Westley.

Herrera, F., & Lozano, M. (2003). Fuzzy adaptive genetic algorithms: Design, taxonomy, and future directions. *Soft Computing, 7*, 545–562.

Hyun, C. J., Kim, Y., & Kim, Y. K. (1998). A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. *Computers and Operations Research, 25*(7-8), 675–690.

Janiak, A., Krysiak, T., & Trela, R. (2011). Scheduling problems with learning and ageing effects: A survey. *Decision Making in Manufacturing and Services, 5*(1-2), 19–36.

Kennedy, J., Eberhart, R. C., & Shi, Y. (2001). *Swarm intelligence*. San Francisco, CA: Morgan Kaufman.

Kim, Y. K., Kim, Y., & Kim, Y. J. (2000). Two-sided assembly line balancing: A genetic algorithm approach. *Production Planning and Control, 11*(1), 44–53.

Kumar, R., & Singh, P. K. (2007). Pareto evolutionary algorithm hybridized with local search for bi-objective TSP. *Studies in Computational Intelligence, 75*, 361–398.

Kuo, W. H., & Yang, D. L. (2006). Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect. *European Journal of Operational Research, 174*, 1184–1190.

Lapierre, S. D., & Ruiz, A. B. (2004). Balancing assembly lines: An industrial case study. *Journal of Operational Research Society, 55*, 589–597.

Lee, T. O., Kim, Y., & Kim, Y. K. (2001). Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering, 40*, 273–292.

Lummus, R. (1995). A simulation analysis of sequencing alternatives for JIT lines using kanbans. *Journal of Operations Management, 13*, 18–191.

Ma, H. (2010). An analysis of the equilibrium of migration models for biogeography-based optimization. *Information Sciences, 180*(18), 3444–3464.

Mansouri, S. A. (2005). A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines. *European Journal of Operational Research, 167*(3), 696–716.

Manupati, V. K., Thakkar, J. J., Wong, K. Y., & Tiwari, M. K. (2013). Near optimal process plan selection for multiple jobs in networked based manufacturing using multi-objective evolutionary algorithms. *Computers & Industrial Engineering, 66*, 63–76.

McMullen, P. R. (1998). JIT sequencing for mixed model assembly lines with setups in Tabu search. *Production Planning & Control, 9*(5), 504–510.

McMullen, P. R. (2001a). An efficient frontier approach to addressing JIT sequencing problems with setups via search heuristics. *Computers & Industrial Engineering, 41*(3), 335–353.

McMullen, P. R. (2001b). An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives. *Artificial Intelligence in Engineering, 15*(3), 309–317.

McMullen, P. R., & Frazier, G. V. (2000). A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line. *IIE Transactions, 32*(8), 679–686.

McMullen, P. R., Tarasewich, P., & Frazier, G. V. (2000). Using genetic algorithms to solve the multi-product JIT sequencing problem with setups. *International Journal of Production Research, 38*(12), 2653–2670.

Miltenburge, J. (1989). Level schedules for mixed-model assembly lines in just-in-time production systems. *Management Science, 35*(2), 192–207.

Miltenburg, J., Steiner, G., & Yeomans, S. (1990). A dynamic programming algorithm for scheduling mixed-model just-in-time production systems. *Mathematical Computation Modeling, 13*(3), 57–66.

Montgomery, D. C. (2013). *Design and analysis of experiments*. John Wiley & Sons, Inc..

Mosheiov, G. (2001). Scheduling problems with a learning effect. *European Journal of Operational Research, 132*, 687–693.

Pinedo, M. (2008). *Scheduling: Theory and systems*. Upper Saddle River, NJ: Prentice-Hall.

Rahimi-Vahed, A. R., Rabbani, M., Tavakkoli-Moghaddam, R., Torabi, S. A., & Jolai, F. (2007). A multi-objective scatter search for a mixed-model assembly line sequencing problem. *Advanced Engineering Informatics, 21*(1), 85–99.

Simaria, A. S., & Vilarinho, P. M. (2009). 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computer and Industrial Engineering, 56*(2), 489–506.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transaction on Evolutionary Computation, 12*(6), 702–713.

Stiener, G., & Yeomans, S. (1993). Level schedules for mixed-model, just-in-time assembly process. *Management Science, 39*(6), 728–735.

Subudom, P. (2009). *Two-sided assembly line balancing in automotive assembly plant*. Master Thesis, Industrial Engineering Department, Faculty of Engineering, Chulalongkorn University, Thailand.

Talbi, E.-G. (2009). *Metaheuristics*. John Wiley & Sons, Inc.

Tavakkoli-Moghaddam, R., & Rahimi-Vahed, A. R. (2006). Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system. *Applied Mathematics and Computation, 181*(2), 1471–1481.

Tsutsui, S. (2006). Node histogram vs. edge histogram: A comparison of probabilistic model-building genetic algorithms in permutation domains. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 1939–1946).

Watson, R. A., & Pollack, J. B. (2000). Recombination without respect: Schema combination and disruption in genetic algorithm crossover. In *Proceedings of the 2000 genetic and evolutionary computation* (pp. 112–119).