APEM journal

Advances in Production Engineering & Management
Volume 14 | Number 1 | March 2019 | pp 51–64
https://doi.org/10.14743/apem2019.1.311

ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Productivity improvement with parallel adjacent U-shaped assembly lines

Chutima, P.[a,b,*], **Suchanun, T.**[a]

[a]Industrial Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand
[b]Academy of Science, The Royal Society of Thailand, Bangkok, Thailand

A B S T R A C T

A novel configuration of assembly lines was proposed in this research, namely parallel adjacent U-shaped assembly lines (PAUL). Typically, in a multiple U-line facility, each U-line is designed to work independently which may cause some workstations were not fully functioned. The PAUL aimed at increasing the utilisation of the whole facility by allowing cross-trained workers to work on the opposite legs of the adjacent U-lines (multi-line workstations). This configuration is easier to implement than parallel U-lines due to no restriction in terms of the lengths of U-lines to be paralleled and hidden expenditures that could be incurred in shop floor reconstruction. Since the line balancing of the PAUL is NP-hard and many conflicting objectives need to be optimised simultaneously, the evolutionary meta-heuristic which was the hybridisation of the multi-objective evolutionary algorithm based on decomposition (MOEA/D) and particle swarm optimisation (PSO), namely MOEA/D-PSO, was developed to effectively solve the problem. In addition, the decoding algorithm to convert the solutions obtained from MOEA/D-PSO into the PAUL's configuration was proposed. The performance of MOEA/D-PSO was evaluated against MOEA/D and multi-objective particle swarm optimisation (MOPSO). The experimental results reveal that MOEA/D-PSO outperformed its rival algorithms under the convergence-related performance.

A R T I C L E   I N F O

Keywords:
Assembly line;
U-shaped assembly line;
Parallel adjacent assembly line;
Assembly line balancing;
Productivity improvement;
Multi-objective optimisation;
Evolutionary algorithm (MOEA/D);
Particle swarm optimisation (PSO)

**Corresponding author:*
cparames@chula.ac.th
(Chutima, P.)

Article history:
Received 29 October 2018
Revised 20 December 2019
Accepted 21 January 2019

1. Introduction

Nowadays, manufacturers are encountering shortened product life cycle because of ever-escalating technological innovation, fierce market competition and rapid change in customers' taste. To maintain their competitiveness and survive in heavily competitive businesses, the just-in-time (JIT) philosophy has been adopted to reduce wastes and excessive inventories in production systems. As a result of JIT implementation, the traditional straight-shaped production lines are switched to the U-shaped ones (hereafter called U-lines) to improve their production efficiencies. Moreover, the mixed-model production, another main integral ingredient of JIT, supersedes the mass production of one model of a single product to cope with the swift growth of product varieties in a cost-effective manner.

In view of the production, a U-line offers more options in task-to-workstation assignments since workers are allowed to work on both sides (Front and Back) of the line in crossover workstations apart from regular workstations which are placed only on any single side of the line. This provides a higher opportunity to consolidate more tasks into workstations resulting in higher compact workload, better workload distribution and greater line utilisation. The popularity of U-lines is attributed to lots of benefits gained from production volume flexibility, operator flexibility, less number of workstations, no special material handling equipment needed, less space requirement, greater workplace visibility, increased communication and teamwork, multi-skilled workers, lower inventory, easier production planning and better quality control [1].

In most manufacturing companies which adopt JIT, after a period of production, they may notice that their current capacities are not enough to fulfil increasing demands because their businesses are flourishing and extra orders are received continuously. These orders may be the same products previously offered or even new products never been done before. The simplest way to deal with this problem is to add more duplicated or newly designed lines into the system. Since the planning and operations of these lines are conducted independently as if they cannot interact with others, the system as a whole comprising many unrelated lines is normally underutilised.

Recently, Gökçen *et al.* [2] proposed a new line configuration, namely parallel lines, where one or more straight-lines are allowed to work simultaneously under common resources (e.g. workers or equipment) through multi-line workstations to increase the capacity of the system without the cost of additional lines. This concept was further adapted to fit in the environment of U-lines by Küçükkoç and Zhang [3], namely parallel U-shaped lines (PUL), where two U-lines are placed in parallel to one another to take advantages of multi-line workstations which bestride between both adjacent U-lines.

Although the concept of the PUL is theoretically sound, a number of practical issues could be uncovered as follows. First, the application of this configuration is limited only when the lengths of both adjacent U-lines are more or less the same. If the outer U-line is much longer than the inner U-line and multi-line workstations are located between the Bottoms of both lines, the workers who work in these workstations have to walk back and forth between two legs of the lines in a long distance. This makes multi-line workers unhappy because their jobs are more tired than those working in other places and perhaps they would refuse to work in such workstations.

In contrast, if the inner U-line is much longer than the outer U-line, there will be no multi-line workstation at the Bottoms of both lines since no Bottom of the outer line is existed. Without the Bottom of the outer line, two legs (Front and Back) of the outer line are split apart and hence material transfers between these legs need a great help from material handling equipment or floating workers bringing about additional operational costs.

Second, in order to reconfigure the system according to the PUL, in case of two U-lines are originally working independently, the lines must be reorganised by moving one U-line to encompass with the other. The cost of line relocation may be quite excessive, particularly when heavy machines are parts of the workstations and the foundation on the shop floor for machinery placement needs further reconstruction.

In this paper, an alternative layout of multiple U-lines is proposed, namely parallel adjacent U-lines (PAUL). This layout has no limitation as of the PUL and it is much easier to implement in practice since less machinery movement is required. The PAUL's environment comprises two or more U-lines adjacently located as ubiquitously found in industry. Instead of treating each U-line as an independent entity, multi-line workers are allowed to handle tasks on both legs of the adjacent U-lines, i.e. Front of one U-line and Back of the other U-line (Fig. 1). As a result, the underutilised capacity of independent lines on some workstations, if any, could be unleashed by implementing the PAUL. To our best knowledge, this novel configuration has never been addressed in literature before.

To systematically form the PAUL so that the production is flown smoothly, the disparity of workloads among workers should be minimised to minimise the unused capacity of the lines. This problem is known as the PAUL balancing problem (PAULBP). In this paper, we assume that each U-line of the PAUL produces mixed-model products to reflect real-life applications. In addition, many conflicting objectives are optimised simultaneously. Because the problem is NP-hard, evolutionary algorithms seem to be an effective solution technique. As a result, the multi-objective evolutionary algorithm based on decomposition (MOEA/D) hybridised with particle swarm optimisation (PSO) is proposed to solve the PAULBP.

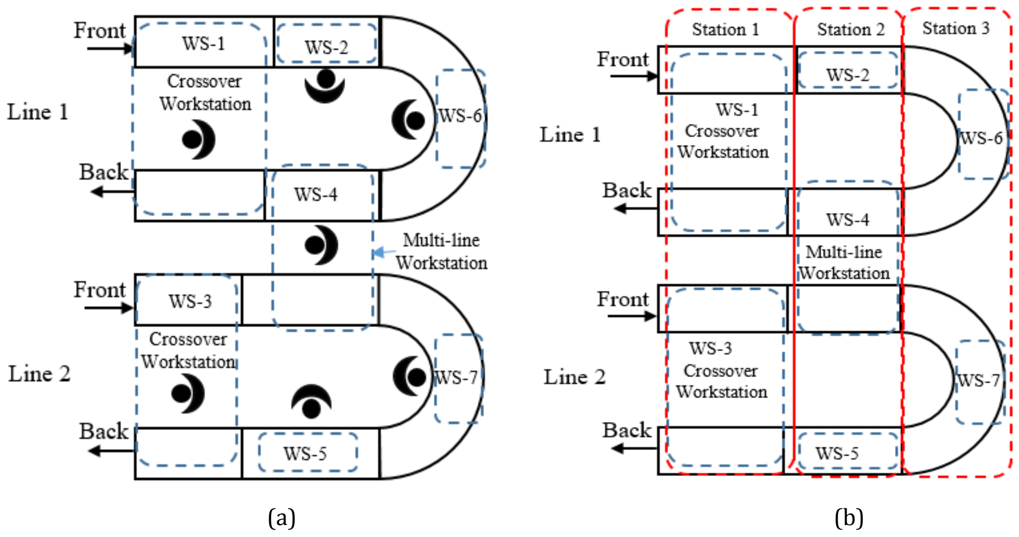

Fig. 1 PAUL configuration: (a) without station number, and (b) with station number

The remaining sections are organised as follows. The detailed description of the PAUL is presented in Section 2. The approach to solving the PAULBP is proposed in Section 3. Experimental designs, results and discussions are explained in Section 4. Finally, the conclusion and future research are discussed in Section 5.

2. Problem definition

Obviously, in a production system (e.g. electronics and electrical industry) where several U-lines are located close to each other (i.e. the walking time between adjacent U-lines of multi-line workers is negligible) could provide an opportunity for better line balancing. In this research, the layout of two U-lines laid out adjacently side-by-side is considered (Fig. 1). The entry and exit points of both lines are on the same side. By this arrangement, an additional form of workstations could be realised apart from regular and crossover workstations as typically found in traditional U-lines. This type of workstation is called a multi-line station which could occur between the Back of Line 1 and the Front of Line 2. The worker who is assigned to perform tasks in a multi-line station is cross-trained to manipulate different product families and models belonging to both U-lines of the PAUL. The assumptions used in this research are as follows: (a) both U-lines produce mixed-model products, (b) task times, cycle times and precedence diagrams of products on both U-lines are given and fixed, (c) cycle times of both U-lines could be different, (d) workers are cross-trained, and (e) walking time of the worker is negligible regardless of workstation types.

2.1 Objective functions

Five objectives related to line efficiency and workload distribution are employed in this research. The reasons for using many objectives as such is to evaluate the effectiveness of the proposed algorithm in dealing with the problems with high dimensional search space and the problem itself is a multi-objective optimisation in nature. Note that these objectives are conflicted with each other; therefore, trade-offs among them are inevitable. The followings are the formulations and descriptions of the objectives.

(1) Minimise the number of workstations (N_W): If the workloads assigned to all workstations in the PAUL are equal or nearly equal to the cycle time of the system, the efficiency of the system will be high resulting in less number of workstations required.

$$f_1 = Minimise\ N_W \tag{1}$$

(2) Minimise workload variations between workstations (B_b): This objective attempts to equalise the workload assigned to each workstation as much as possible. Highly unbalanced workload causes a bottleneck, inequality of workers and also fatigues to the worker who works in the bottleneck workstation.

$$f_2 = Minimise\ B_b = \frac{N_w}{N_w - 1} \sum_{k=1}^{N_w} \sum_{b=1}^{3} \left(\frac{S_{k,b}}{TAD} - \frac{1}{N_w} \right)^2 \tag{2}$$

(3) Minimise unbalanced workloads within the workstation (B_w): This objective is trivial in a single-model assembly line. However, it is vital for a mixed-model assembly line in which the task times of different product models could be varied. The effect of this problem is apparent while sequencing products into the line since a worker may have to work much longer than the cycle time in some cycle, but very short in another cycle. In some cases, utility workers may be called upon to ease the problem.

$$f_3 = Minimise\ B_w = \frac{M}{N_w(M-1)} \sum_{k=1}^{N_w} \sum_{m=1}^{M} \left(S_{km} - \frac{1}{M} \right)^2 \tag{3}$$

$$q_m = \frac{D_m}{\sum_{m=1}^{M} D_m};\ 0 \le q_m \le 1 \ \text{ and } \sum_{m=1}^{M} q_m = 1 \tag{4}$$

$$S_{km} = \begin{cases} 0, & \text{if } \sum_{m=1}^{M} q_m s_{km} \\ \dfrac{q_m s_{km}}{\sum_{m=1}^{M} q_m s_{km}}, & \text{otherwise} \end{cases} \tag{5}$$

(4) Minimise unrelatedness of tasks in the workstation (TUR): Two tasks assigned to a workstation are related to each other if both of them are directly interconnected in the precedence diagram. Their relationships are the predecessor or successor of the other. The related tasks normally require similar tools and skill of workers; hence, assigning related tasks to the same workstation facilitates skill development and expertise to a worker resulting in high system efficiency. Chutima and Chimklai [4] proposed the formulation to measure the unrelatedness of assigned tasks in the workstation as follows.

$$f_4 = Minimise\ TUR = N_w - \frac{N_w}{\sum_{k=1}^{N_w} SN_k} \tag{6}$$

where SN_k is the network of tasks that are directly related in workstation k.

(5) Minimise the number of stations (N_s): The line length of the system comprising many U-lines laid down adjacently can be measured by counting the number of stations which are compactly arranged in a longitudinal direction. Shop floor utilisation will be high if the system's line length is short. This objective can also be used to penalise the system configurations that are not an authentic PAUL which will be discussed in the next section.

$$f_5 = Minimise\ N_s \tag{7}$$

2.2 Authentic PAUL configuration

An authentic PAUL must consist of at least one crossover workstation and one multi-line workstation, possibly incorporated with or without any regular workstation. Many system configurations that claim to be the PAUL but, in reality, they are just look alike. As a result, the benefits of the PAUL may not be realised from such unauthentic PAULs, particularly low N_w and short line length which will be illustrated in the following examples.

Fig. 1 illustrates the authentic PAUL configuration in which one crossover workstation is found in Line 1 and Line 2, one multi-line workstation is located between the adjacent legs of Line 1 and Line 2, and two regular workstations are on Line 1 and Line 2. In this case, the number of stations in the system (i.e. line length N_s) is three.

Fig. 2 depicts the two-line system which consists of four regular workstations on Line 1 and Line 2, and one multi-line workstation located in between Line 1 and Line 2. This configuration is not an authentic PAUL since no crossover workstation exists. In fact, this system is parallel lines and the line length is five.

Another example of unauthentic PAULs is shown in Fig. 3. The system consists of a U-line and a straight line working in parallel. There are two regular workstations and one crossover workstation on Line 1, four regular workstations on Line 2, and one multi-line laid between Line 1 and Line 2. The line length of the system is 5.

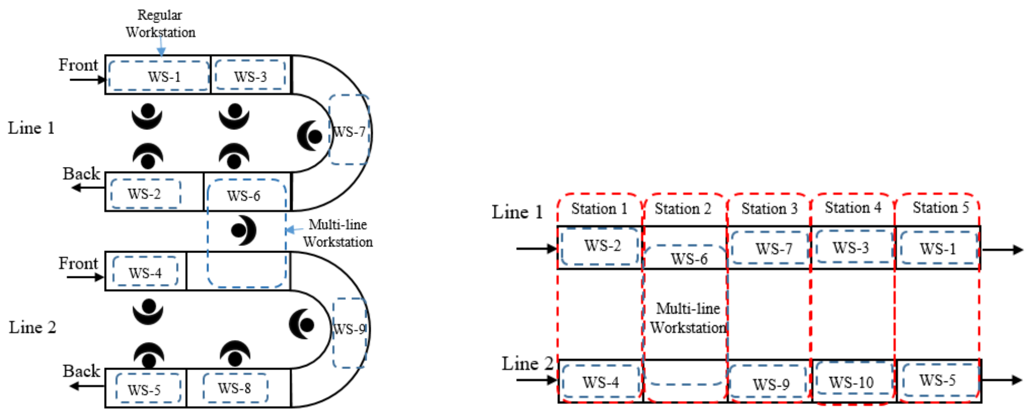

Fig. 2 Parallel lines ($N_s = 5$)

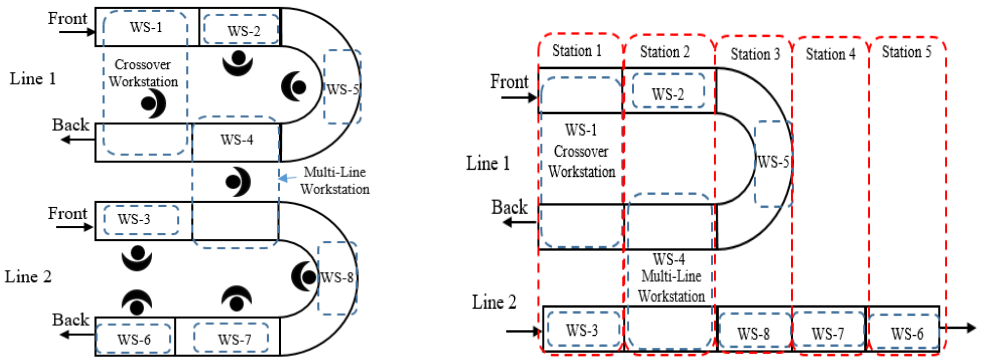

Fig. 3 One U-line and one straight line ($N_s = 5$)

3. Materials and methods

3.1 Hybridisation between MOEA/D and PSO (MOEA/D-PSO)

Much effort of the early research in the line balancing has been emphasised on a single objective optimisation. However, in practice, line balancing is a multi-objective optimisation problem (MOP) since many conflicting objectives need to be realised simultaneously as mentioned in Section 2. When solving this problem, we usually discover numerous optimal solutions which are non-dominated to each other, so-called Pareto optimal solutions (POSs). The best contour of these POSs plotted in the objective space is called the Pareto front (PF).

Various Pareto-based evolutionary multi-objective optimisation algorithms have been developed to solve MOPs. However, with these algorithms, when the number of objectives grows more than three, known as many-objective optimisation problems (MaOPs), the dominance among solutions becomes weakened considerably resulting in worsening selection pressure towards the PF and sluggish the convergence rate of the algorithm.

Recently, the novel algorithm to effectively tackle MaOPs namely multi-objective evolutionary algorithm based on decomposition (MOEA/D) in which an original MaOP is bisected into numerous subproblems using different weight vectors which are uniformly distributed along the objective domain was developed by Zhang and Li [5]. These single objective optimisation subproblems are collaboratively optimised simultaneously in each generation. Since the fitness assignments of MOEA/D are determined by the scalar aggregation function rather than the Pareto domination as conventional, the number of objectives has a minimal effect to its selection pressure.

MOEA/D normally uses genetic operators as an offspring generating mechanism, e.g. single point crossover. However, such genetic operator has a serious drawback in computational expensive when compares with particle swarm optimisation (PSO) [6]. As a result, the genetic operator of MOEA/D is replaced with the mechanism of PSO in this research to improve the convergence rate. The hybrid algorithm is afterwards called MOEA/D-PSO.

PSO was developed by Kennedy and Eberhart [7] to mimic the collaborative social movement of the large biological swarm in searching for food such as a bird flock or fish school. PSO is a population-based meta-heuristic which stochastically manipulates its offspring generation mechanism without any evolutionary operator like appearing in GA, e.g. crossover and mutation. As a result, many relative advantages offered by PSO include fast convergence, fewer parameter settings, less memory consumption, etc.

In the context of PSO, a particle denotes an individual solution of the problem, a swarm represents the population of solutions, and the optimum solution is the food. Each particle has three key attributes, i.e. position (its current solution), velocity (magnitude and direction of the trajectory towards the optimum solution), and fitness (relative performance). The particle navigates its flight by regularly adjusting its velocity with the supervision from two sources, i.e. its own best flying experience (personal best known as *P*best) and entire population best flying experience (global best known as *G*best). The pseudo code of MOEA/D-PSO is described as follows.

Pseudo code of MOEA/D-PSO

```
1.  Generate weight vector $\lambda$ for the population size $N$ using simplex lattice design.
2.  Define the parameters of the algorithm, i.e. the neighbourhoods ($T$) of each weight vector, inertia weight $w$, and learning factors $c_1$ and $c_2$.
3.  Determine particle position vector $X_j$ for each particle $j = 1, \dots, N$ whose size equals the total number of tasks ($d$) in the PAUL using $R[0,1]$.
4.  Determine particle velocity vector ($V_j$) for $j = 1, \dots, N$ whose size equals the total number of tasks ($d$) in the PAUL using $R[0,1]$.
5.  Determine the configuration of the PAUL using the decoding algorithm (to be explained in the next section) and calculate all $m$ objectives of each $X_j$.
6.  Determine the minimum and maximum values of each objective, i.e. $z_m^*$ and $f_{m(max)}$.
7.  Set the initial values of Pbest ($P$) and Gbest ($G$) of each particle by $P_j = X_j$ and $G_j = X_j$.
8.  Find the non-dominated solutions in the current population, add them into the external population ($EP$), find the non-dominated solution in $EP$, and trash all dominated solutions from $EP$ (note that the initial $EP = \phi$).
9.  Find the velocity vector of each particle $j$ using $V_j(\mathrm{t}+1) = wV_j(t) + c_1r_1\big(P_j(t) - X_j(t)\big) + c_2r_2\big(G_j(t) - X_j(t)\big)$ and then find its new position vector using $X_j(\mathrm{t}+1) = X_j(t) + V_j(t+1)$.
10. Find the new configuration of the PAUL using the decoding algorithm and calculate all $m$ objectives of each $X_j$.
11. Update $z_m^*$ and $f_{m(max)}$.
12. Find normalised values of each objectives using $\overline{f_i} = \frac{f_i - z_i^*}{f_{i(max)} - z_i^*}$ for each $X_j$, $P_j$ and $G_j$.
```

13. Update Pbest of each X_j by comparing the Tchebycheff function (minimize $g_j(\mathrm{x}) = \max_{1 \le i \le m} \{\lambda_{ij}|\overline{f_i}|\}$) between P_j and X_j. If X_j is better than P_j, set $P_j = X_j$; otherwise, maintain current P_j.
14. Update Gbest by comparing the Tchebycheff function between G_j of the randomly selected neighbourhoods and X_j. If X_j is better than G_j, set $G_j = X_j$; otherwise, maintain current G_j.
15. If the termination condition is not met, go to step 8; otherwise, stop the algorithm.

3.2 Decoding algorithm

The solution string in this paper is represented by a priority-based scheme where the number under the task indicates its assignment priority. For example, nine (A1,…, A9) and seven (B1,…, B7) tasks are produced by Line 1 and Line 2, respectively, as shown in Fig. 4. According to the arrangement of String 1, the priorities of tasks from highest to lowest are B6, B4, B7,…, A5 and A8.

String	Task															
	A1	A2	A3	A4	A5	A6	A7	A8	A9	B1	B2	B3	B4	B5	B6	B7
1	8	13	12	10	15	9	11	16	6	5	4	7	2	14	1	3

Fig. 4 An example of solution strings

The decoding algorithm to transform the solution string into the corresponding PAUL is as follows.

Decoding algorithm

1. Determine the total number of to-be-assigned tasks by adding the number of tasks in Line 1 with Line 2.
2. Determine the assignable task list which is a set of tasks that are eligible for the assignment without violating the precedence constraint. For U-line, the assignable tasks are those that locate on the left-hand side of the precedence diagram without any predecessor (assign to the Front) and on the right-hand side of the precedence diagram without any successor (assign to the Back).
3. Let S1 be the set of tasks that can be assigned to the Front of Line 1, S2 be the set of tasks that can be assigned to the Back of Line 1, S3 be the set of tasks that can be assigned to the Front of Line 2, and S4 be the set of tasks that can be assigned to the Back of Line 2. Group all assignable tasks into their appropriated sets, i.e. S1, S2, S3 and S4.
4. Open a new workstation.
5. Select the task with the highest priority from the assignable task list and assign it to the corresponding side (i.e. S1, S2, S3 and S4) of the U-line.
6. Reduce the number of to-be-assigned tasks by 1. If the number of to-be-assigned tasks is equal to 0, then the algorithm is completed; otherwise, go to (7).
7. Compute the remaining available time of the workstation.
8. Update the assignable task list by considering only those that their task times are less than the remaining available time of the current workstation without violating the precedence constraint as follows:
   a) If the assigned task is in S1 (the Front of Line 1), the assignable tasks are those that are in S1 (to form a regular workstation) and S2 (to form a crossover workstation).
   b) If the assigned task is in S2 (the Back of Line 1), the assignable tasks are those that are in S1 (to form a crossover workstation), S2 (to form a crossover workstation) and S3 (to form a multi-line workstation).
   c) If the assigned task is in S3 (the Front of Line 2), the assignable tasks are those that are in S2 (to form a multi-line workstation), S3 (to form a regular workstation) and S4 (to form a crossover workstation).
   d) If the assignable task is in S4 (the Back of Line 2), the assignable tasks are those that are in S3 (to form a multi-line workstation) and S4 (to form a regular workstation).

9. If the assignable task list is empty and the to-be-assigned tasks still exist, then go to (4); otherwise, select the task with the highest priority from the assignable task list and assign it to the classified side of the line.
10. Reduce the number of to-be-assigned tasks by 1. If the number of to-be-assigned tasks is empty, then the algorithm is completed; otherwise, go to (11).
11. Compute the remaining available time of the workstation.
12. Update the assignable task list by considering only those that their task times are less than the remaining available time of the current workstation without violating the precedence constraint as follows:
    a) If the recently assigned task still forms a regular workstation with the previously assigned tasks, the assignable tasks can be found in the same way as 8(a)-8(d).
    b) If the recently assigned task forms a crossover workstation with the previously assigned tasks, the assignable tasks are those that belong to S1 and S2, or S3 and S4, depending on the line (Line 1 or Line 2) where the crossover workstation is located.
    c) If the recently assigned task forms a multi-line workstation with the previously assigned tasks, the assignable tasks are those that belong to S2 and S3 only.
13. Repeat steps 9-12.

Assume that the precedence diagrams of the products to be assembled on Line 1 and Line 2 are shown in Fig. 5. The common cycle time of the lines is 30. The numerical example of the algorithm is demonstrated in Table 1. Fig. 6 depicts the resultant PAUL which consists of three regular workstations on Line 1, one crossover workstation on Line 1, one regular workstation on Line 2, one crossover workstations on Line 2, and one multi-line workstation.

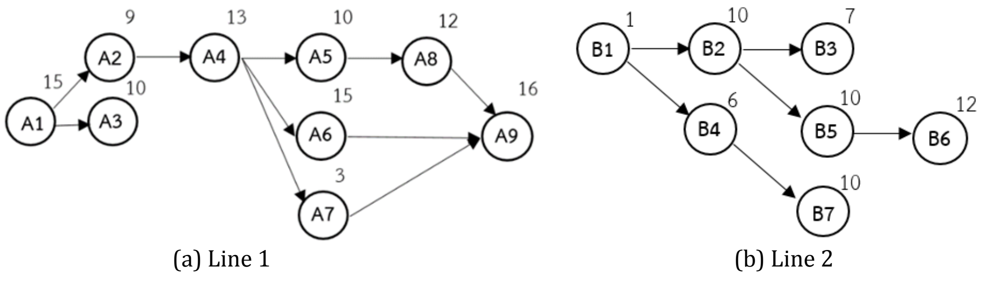

Fig. 5 An example of solution strings

Table 1 Task-to-workstation assignment (cycle time = 30)

Work-station	Assignable Task				Select	Task Time	Idle Time
	Line 1		Line 2				
	S1	S2	S3	S4			
1	A1	A9	B1	B3, B6, B7	B6	12	18
	A1	A9	B1	B3, B5, B7	B7	10	8
	A1	A9	B1	B3, B5, B4	B4	6	2
	A1	A9	B1	B3, B5	B1	1	1
2	A1	A9	B2	B3, B5	B2	10	20
	A1	A9	B3, B5	B3, B5	A9	16	4
	A1	A8, A6, A7	B3, B5	B3, B5	A7	3	1
3	A1	A8, A6	B3, B5	B3, B5	B3	7	23
	A1	A6, A8	B5	B5	B5	10	13
4	A1	A8, A6	-	-	A1	15	15
	A2, A3	A8, A6	-	-	A6	15	0
5	A2, A3	A8	-	-	A3	10	20
	A2	A8	-	-	A2	9	11
6	A4	A8	-	-	A4	13	17
	A5	A8	-	-	A5	10	7
7	A8	A8	-	-	A8	12	18

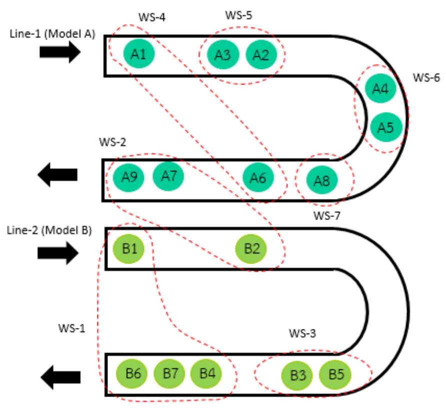

Fig. 6 The resultant PAUL

4. Results and discussion

4.1 Experimental design

Problem set

Twelve problems were used to test the performances of MOEA/D-PSO. The problems were modified from previously published research to fit in the PAUL's environment and they were classified into three sizes, i.e. small, medium and large. The number of tasks ranged from 30-50, 50-100 and 100-170 for small, medium and large, respectively. In each problem, the cycle times of the lines were assumed unequal and varied in three levels; hence, the common cycle time of the system was determined according to [2]. In addition, mixed-models of the product were produced by each U-line. Table 2 shows the detail of each problem.

Parameter settings of the algorithms

Three algorithms were tested in this research, i.e. MOEA/D, MOPSO and MOEA/D-PSO. MOPSO is the conventional PSO algorithm but applying the Pareto-based fitness scheme to guide its search trajectory. In order to provide a fair-play contest, all algorithms were carefully coded and tuned so that they could execute at their best performances. The parameter tuning of the algorithms was based on the result obtained from statistical analyses of the experimental designs, particularly the general full factorial design. All programs were coded in MathLab on a notebook computer using Intel® Core™ i7-7700HQ CPU@2.8GHz 8.00 GB RAM 64-bit operating system operated under Microsoft Windows 10 Pro.

Two parameters of MOEA/D including the number of weight vectors in the neighbourhood and the maximum number of solutions replaced by each offspring were set at 10% and 20%, respectively. The values of inertia weight, cognitive learning parameter and social learning parameter (learning) for MOPSO were set at 1, 1.5 and 1, respectively. For MOEA/D-PSO, the values of the number of weight vectors in the neighbourhood, the maximum number of solutions replaced by each offspring, inertia weight, cognitive learning parameter and social learning parameter were set at 10 %, 20 %, 1, 1.5 and 1, respectively. The number of populations in each algorithm was 133. The number of generations was 1000, 1500 and 2000 for small, medium and large problems, respectively.

Pareto-based metrics

To evaluate the relative performances of the algorithms in a Pareto sense, several metrics were employed in this research. The metrics which are related to the convergence performance of the algorithms including generational distance (GD), inverted generational distance (IGD), ratio of non-dominated solutions (self-comparison, R_{NDS1}) and ratio of non-dominated solutions (Pareto-optimum comparison, R_{NDS2}). $Spread$ is a metric to indicate the diversity of non-dominated solutions. The detailed formulations of all Pareto-based metrics were discussed by Coello Coello and Cortés [8] and Chutima and Olarnvanitchai [9].

Table 2 Problems used in the experiments

No.	Line1				Line2				Total Task	Common Cycle Time
	Problem	Models	MPS	Cycle Time	Problem	Models	MPS	Cycle Time		
1	Mitchell	3	2:1:2	15	Jackson	2	3:1	10	32	30
				18				12		36
				21				14		42
2	Jackson	3	1:2:1	9	Rozieg	2	3:1	18	36	18
				11				22		22
				13				26		26
3	Mitchell	2	2:1:2	17	Rozieg	3	1:2:1	34	46	34
				21				18		126
				21				21		21
4	Rozieg	3	1:2:1	21	Rozieg	3	2:1:2	42	50	42
				25				50		50
				16				32		32
5	Heskiaoff	3	1:1	138	Heskiaoff	2	1:1	205	56	28290
				205				216		44280
				216				324		648
6	Gunther	3	1:1	41	Sawyer	2	2:3	41	65	41
				54				54		54
				81				81		81
7	Killbridge	3	1:1	79	Heskiaoff	2	2:3	138	73	10902
				110				205		4510
				110				216		11880
8	Killbridge	2	1:1	57	Kilbridge	2	1:3	79	90	4503
				92				110		5060
				110				110		110
9	Killbridge	3	2:1:2	79	Tonge	3	1:2:1	410	115	32390
				110				468		25740
				110				527		57970
10	Tonge	3	1:1:1	320	Tonge	2	3:1	320	140	320
				207				270		6210
				293				220		64460
11	Tonge	2	3:1	220	Wee-mag	3	1:1:1	270	145	5940
				252				84		252
				303				101		303
12	Arcus1	3	1:1:2	6842	Acrus1	3	1:2:1	6842	166	6842
				7571				7571		7571
				6309				6309		6309

4.2 Experimental results

Table 3 in Appendix A shows the relative performances of all algorithms in tackling the PAULBP. As mentioned earlier, two performance aspects can be evaluated when facing MOPs, i.e. convergence and spread. The first convergence-related metric is GD which indicates the distances between the non-dominated solutions (NDSs) on the PF obtained by the algorithm and the closest NDSs on the approximated true Pareto front (ATPF). Note that ATPF is constructed by applying non-dominated sorting to the combined PFs of all algorithms obtained after the algorithm is terminated. If GD is 0 (the best value of GD), the PF of the algorithm is perfectly overlapped with the ATPF. It is obvious that GDs of MOEA/D-PSO are always lowest, followed by MOEA/D and MOPSO, regardless of the problems' sizes. In addition, GDs obtained by MOEA/D-PSO are always very close to 0 meaning that most of its obtained NDSs are on the ATPF.

IGD is similar to GD, but it measures the distances between the NDSs on the ATPF and the closest NDSs on the PF obtained by the algorithm. As a result, IGD also implies the coverage of the extreme points on ATPF by the algorithm. The algorithm with a lower IGD is the better one (the best value of IGD is 0). It is obvious that MOEA/D-PSO always has the lowest values of IGD comparing with MOEA/D and MOPSO, regardless of problem's sizes and cycle times.

R_{NDS1} and R_{NDS2} are another important convergence-related metrics. They specify how many of the NDSs on the PF of the algorithm belongs to the ATPF. R_{NDS1} compares this number with its owned number of NDSs on the PF; whereas, R_{NDS2} compares this number with the NDSs on the ATPF. The higher the values of R_{NDS1} and R_{NDS2} are the better algorithm. The results show clearly that MOEA/D-PSO often outperforms MOEA/D and MOPSO. When the results from R_{NDS1} and R_{NDS2} are interpreted along with GD and IGD, it conveys about the PF of MOEA/D-PSO that it is located pretty close to the ATPF and all of its NDSs are almost on the ATPF. In addition, the PF of MOEA/D-PSO covers the extreme point on the ATPF.

$Spread$ is used to assess the distribution of NDSs produced by the algorithm which is related to the diversity of the solutions. This metric determines how much difference between the distance between two adjacent NDSs and the average distance. The lower $Spread$ (i.e. more uniform distribution) is the better algorithm. Although MOPSO often provides the best $Spread$, particularly in large sizes' problems, its value is marginally lower than MOEA/D and MOEA/D-PSO. This result comes with unsurprising since there is no diversity preservation mechanism embedded in any algorithm tested in this research.

Another aspect observed during the experiments, but does not present in this paper because of page limitation, is the number of workstations (N_w) and stations (N_S) created by each algorithm. It is found that the best value of N_w created by all algorithms is almost the same as the ideal number which calculates from dividing the total task time by the cycle time. MOEA/D-PSO always obtains the best N_w comparing with the other algorithms. In addition, the best value of N_S generated by each algorithm is the same. This reflects the effectiveness of the decoding method that is proposed in this research, as well as the PAUL's configuration.

In theory, the PAUL's concept is viable and could be extended to integrate more than two or even all U-lines located in close vicinity of each other to form a multi-PAUL facility. However, in practice, a number of issues that should be carefully addressed are as follows. For some factories that do not plan to utilise the PAUL in advance, it may be necessary to re-arrange their existing U-lines to be aligned with the PAUL layout.

The distance between the adjacent U-lines which will be used to form the PAUL should not be too long. In addition, multi-line workers should be more appropriate to work in a standing posture since they will have to travel back and forth between two legs of different U-lines. The need to walk while working could cause fatigue in the workplace with workers.

The features of tasks assigned to two legs of a multi-line workstation of the PAUL must not be too much diverse, e.g. soldering IC's pins (expert hands) and visual inspection (expert eyes). If possible, they should be in the same category, i.e. requiring the same skill. In fact, multi-skilled workers should not be bombarded with too many skill trainings since it could prevent them from being an expert in the field. In addition, higher wages should be paid to multi-line workers since their responsibilities are much higher than workers in regular workstations.

5. Conclusion

A novel assembly line configuration widely found in many multiple U-lines facilities but no one has ever utilised it, namely a PAUL, is proposed in this paper. This configuration is developed to increase the productivity of two or more U-lines placed adjacently and some of their workstations are underutilised (i.e. high idle time) while they are operated independently. To increase the utilisation rate of the whole system, paralleling these U-lines is done by allowing the formation of multi-line workstations located between their adjacent legs (Front of one U-line and Back of the other adjacent U-line). Multi-skilled labours are allocated such workstations to ensure the smoothness of mixed-model production flow on the PAUL. Several conflicting objectives of the PAULBP are optimised simultaneously and evaluated under a Pareto sense. Since the problem is NP-hard, the decomposition-based algorithm to generate good layouts of the PAUL is proposed namely MOEA/D-PSO which is the hybridisation between MOEA/D and PSO. The algorithm is tested against MOEA/D and MOPSO to assess their relative performances. Several problems with different sizes, number of tasks, product mixes and cycle times are employed as test-bed cases. The results reveal clearly that MOEA/D-PSO outperforms MOEA/D and MOPSO in terms of convergence-related metrics while their performances are indifferent in diversity-related metric. In addition, the decoding algorithm is quite effective in generating good PAUL layouts. The further research directions could be extended to the PAULBP Type II, considering the walking time of the worker, various skilled labour [10], asynchronous U-lines [11], or simultaneous balancing and sequencing [12].

References

[1] Cheng, C.H., Miltenburg, J., Motwani, J. (2000). The effect of straight- and U-shaped lines on quality, *IEEE Transactions on Engineering Management*, Vol. 47, No. 3, 321-334, doi: 10.1109/17.865901.

[2] Gökçen, H., Ağpak, K., Benzer, R. (2006). Balancing of parallel assembly lines, *International Journal of Production Economics*, Vol. 103, No. 2, 600-609, doi: 10.1016/j.ijpe.2005.12.001.

[3] Küçükkoç, I., Zhang, D.Z. (2015). Balancing of parallel U-shaped assembly lines, *Computers & Operations Research*, Vol. 64, 233-244, doi: 10.1016/j.cor.2015.05.014.

[4] Chutima, P., Chimklai, P. (2012). Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge, *Computers & Industrial Engineering*, Vol. 62, No. 1, 39-55, doi: 10.1016/j.cie.2011.08.015.

[5] Zhang, Q., Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Transactions on Evolutionary Computation,* Vol. 11, No. 6, 712-731, doi: 10.1109/TEVC.2007.892759.

[6] Hassan, R., Cohanim, B., de Weck, O., Venter, G. (2005). A comparison of particle swarm optimisation and the genetic algorithm, In*: Proceedings of 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference,* Austin, Texas, USA, doi: 10.2514/6.2005-1897.

[7] Kennedy, J., Eberhart, R. (1995). Particle swarm optimization, In: *Proceedings of ICNN'95 – International Conference on Neural Networks,* Perth, WA, Australia, Vol. 4, 1942-1945, doi: 10.1109/ICNN.1995.488968.

[8] Coello, C.A.C., Cortés, N.C. (2005). Solving multiobjective optimisation problems using an artificial immune system, *Genetic Programming and Evolvable Machines,* Vol. 6, No. 2, 163-190, doi: 10.1007/s10710-005-6164-x.

[9] Chutima, P., Olarnviwatchai, S. (2016). A multi-objective car sequencing problem on two-sided assembly lines, *Journal of Intelligent Manufacturing,* Vol. 29, No. 7, 1617-1636, doi: 10.1007/s10845-016-1201-6.

[10] Corominas, A., Pastor, R., Plans, J. (2008). Balancing assembly line with skilled and unskilled workers, *Omega,* Vol. 36, No. 6, 1126-1132, doi: 10.1016/j.omega.2006.03.003.

[11] Tiacci, L. (2017). Mixed-model U-shaped assembly lines: Balancing and comparing with straight lines with buffers and parallel workstations, *Journal of Manufacturing Systems,* Vol. 45, 286-305, doi: 10.1016/j.jmsy.2017.07.005.

[12] Defersha, F.M., Mohebalizadehgashti, F. (2018). Simultaneous balancing, sequencing, and workstation planning for a mixed model manual assembly line using hybrid genetic algorithm, *Computers & Industrial Engineering,* Vol. 119, 370-387, doi: 10.1016/j.cie.2018.04.014.

Appendix A

Table 3 Experimental results

Problem	1 (S1)			2 (S2)			3 (S3)			4 (S4)			5 (M1)			6 (M2)		
No. of Tasks (Line1-Line2)	32 (21-11)			36 (11-25)			46 (21-25)			50 (25-25)			56 (28-28)			65 (35-30)		
Cycle Time	30	36	42	18	22	26	34	126	21	42	50	52	28290	44280	648	41	54	81
Generational Distance (GD)																		
MOPSO	0.0980	0.1508	0.1379	0.1349	0.1862	0.1057	0.1110	0.1079	0.1366	0.2265	0.1294	0.1565	0.1928	0.2203	0.1171	0.3239	0.1206	0.1717
MOEA/D	0.0699	0.0749	0.0648	0.0907	0.1156	0.0842	0.0498	0.0631	0.1061	0.1471	0.0616	0.1032	0.0914	0.1538	0.0793	0.1386	0.0732	0.1066
MOEA/D-PSO	**0.0412**	**0.0749**	**0.0616**	**0.0342**	**0.1027**	**0.0651**	**0.0343**	**0.0546**	**0.0180**	**0.1317**	**0.0520**	**0.0313**	**0.0877**	**0.0752**	**0.0508**	**0.0422**	**0.0510**	**0.0732**
Inverted Generational Distance (IGD)																		
MOPSO	0.1031	0.1379	0.1176	0.1467	0.1482	0.1350	0.0959	0.0935	0.1680	0.2183	0.1240	0.1683	0.1968	0.1362	0.1380	0.2882	0.1730	0.1489
MOEA/D	0.0870	0.1072	0.1105	0.1468	0.1480	0.1791	0.1764	0.0972	0.1362	0.2774	0.0714	0.0930	0.1904	0.0894	0.0982	0.1531	0.1389	0.1797
MOEA/D-PSO	**0.0621**	**0.0812**	**0.0727**	**0.1055**	**0.0730**	**0.0788**	**0.0415**	**0.0800**	**0.0892**	**0.0838**	**0.0618**	**0.0926**	**0.1901**	**0.0873**	**0.0956**	**0.1086**	**0.0864**	**0.1204**
R_{NDS1}																		
MOPSO	0.1470	0.0527	0.0689	**0.5562**	0.2514	0.1472	0.0667	0.0513	0.0426	0.1176	0.1647	0.0763	0.0959	0.0291	0.1189	0.0055	0.1480	0.0603
MOEA/D	0.3883	**0.4133**	0.4028	0.0550	0.2906	0.3504	0.4005	0.2628	0.0736	0.2414	0.2859	0.0775	0.3473	0.1923	0.3488	0.1430	0.3194	0.3571
MOEA/D-PSO	**0.4745**	0.4032	**0.4101**	0.2341	**0.3454**	**0.4116**	**0.5016**	**0.4773**	**0.7027**	**0.3095**	**0.4732**	**0.6759**	**0.4769**	**0.5253**	**0.3955**	**0.6063**	**0.3346**	**0.4567**
R_{NDS2}																		
MOPSO	0.0820	0.0283	0.0536	0.0316	0.0952	0.0727	0.0370	0.0379	0.0225	0.0476	0.0825	0.0402	0.0584	0.0227	0.0714	0.0057	0.0750	0.0443
MOEA/D	0.1914	0.2311	0.1905	0.1263	0.1619	0.1273	0.1543	0.1742	0.0449	0.1786	0.1495	0.0632	0.1948	0.1477	0.2071	0.1092	0.1375	0.2089
MOEA/D-PSO	**0.2695**	**0.2453**	**0.2857**	**0.3421**	**0.2762**	**0.3000**	**0.3148**	**0.2879**	**0.4326**	**0.2738**	**0.2732**	**0.4023**	**0.2468**	**0.3295**	**0.2214**	**0.3851**	**0.2875**	**0.2468**
Spread																		
MOPSO	**0.7448**	0.8388	**0.7758**	0.0550	0.8677	0.8937	**0.8989**	**0.8550**	0.9516	0.9487	0.8691	0.9537	**0.7304**	**0.7500**	0.8835	**0.7142**	**0.7956**	**0.8383**
MOEA/D	0.8937	0.8575	0.8745	0.2341	0.9021	0.9441	0.9871	0.9014	**0.8701**	0.8998	**0.8486**	**0.6809**	0.9766	0.7544	0.9103	0.7955	0.9608	0.9190
MOEA/D-PSO	0.8175	**0.7379**	0.8663	**0.5562**	**0.7948**	**0.8827**	0.9182	0.8836	0.9156	**0.7955**	0.8716	0.8231	0.8670	0.9083	**0.8497**	0.8920	0.8929	0.9720

Table 3 Experimental results (continuation)

Problem	7 (M3)			8 (M4)			9 (L1)			10 (L2)			11 (L3)			12 (L4)		
No. of Tasks (Line1-Line2)	73 (45-28)			90 (45-45)			115 (45-70)			140 (70-70)			145 (70-75)			166 (83-83)		
Cycle Time	10902	4510	11880	4503	5060	110	32390	25740	57970	320	6210	64460	5940	252	303	6842	7571	6309
Generational Distance (GD)																		
MOPSO	0.1953	0.1106	0.1576	0.1129	0.0756	0.1232	0.0872	0.1194	0.0977	0.3657	0.2195	0.1568	0.2382	0.2210	0.2321	0.2329	0.3424	0.1838
MOEA/D	0.1629	0.0516	0.0726	0.0633	**0.0358**	0.0817	0.0556	0.0752	0.0823	0.0630	0.1443	0.1296	0.1029	0.1541	0.1394	0.1687	0.1962	0.1239
MOEA/D-PSO	**0.0699**	**0.0455**	**0.0440**	**0.0484**	0.0389	**0.0546**	**0.0530**	**0.0751**	**0.0561**	**0.0565**	**0.0249**	**0.0245**	**0.0951**	**0.1002**	**0.0424**	**0.0758**	**0.0910**	**0.0777**
Inverted Generational Distance (IGD)																		
MOPSO	0.2113	0.1304	0.2155	0.1296	0.0922	0.1203	0.1034	0.1466	0.1862	0.2495	0.2186	0.1537	0.2324	0.1902	0.2140	0.1638	0.1386	0.1349
MOEA/D	0.2096	0.0946	0.1458	0.1944	0.0996	0.1144	0.1639	0.1375	0.1541	0.1156	0.1940	0.1535	0.1648	0.1910	0.1189	0.1297	0.1369	0.1459
MOEA/D-PSO	**0.1227**	**0.0647**	**0.0899**	**0.0888**	**0.0773**	**0.0992**	**0.1001**	**0.1070**	**0.1280**	**0.0885**	**0.1223**	**0.1056**	**0.1031**	**0.1743**	**0.1171**	**0.1220**	**0.1268**	**0.0693**
R_{NDS1}																		
MOPSO	0.0259	0.0626	0.0526	0.2068	0.1495	0.0554	0.2564	0.0250	0.0663	0.0133	0.0450	0.0328	0.1699	0.1050	0.0473	0.0756	0.0175	0.0504
MOEA/D	0.0693	0.3656	0.4114	0.2205	0.3697	0.2094	**0.4559**	0.3247	0.2107	0.4583	0.0775	0.2364	0.1632	0.1327	0.2001	0.1579	0.1134	0.2835
MOEA/D-PSO	**0.5857**	**0.4936**	**0.5781**	**0.4430**	**0.4330**	**0.3859**	0.2722	**0.5234**	**0.3968**	**0.6401**	**0.8400**	**0.6529**	**0.3818**	**0.5104**	**0.5962**	**0.5454**	**0.7155**	**0.3983**
R_{NDS2}																		
MOPSO	0.0161	0.0324	0.0174	0.1115	0.0940	0.0640	**0.1875**	0.0192	0.0739	0.0053	0.0234	0.0152	0.0427	0.0972	0.0160	0.0577	0.0159	0.0265
MOEA/D	0.0484	0.1574	0.2087	0.0808	0.1745	0.1628	0.1602	0.1154	0.1420	0.2090	0.0391	0.1091	0.1325	0.1157	0.1440	0.1538	0.0476	0.1549
MOEA/D-PSO	**0.4355**	**0.3102**	**0.2739**	**0.3077**	**0.2315**	**0.2733**	0.1523	**0.3654**	**0.2841**	**0.2857**	**0.4375**	**0.3758**	**0.3248**	**0.2870**	**0.3400**	**0.2885**	**0.4365**	**0.3186**
Spread																		
MOPSO	**0.8129**	**0.8611**	**0.7586**	**0.8241**	**0.7392**	**0.7719**	0.7754	**0.9236**	**0.8413**	**0.5435**	**0.7135**	**0.8260**	0.9029	**0.6685**	**0.7624**	**0.7755**	**1.0852**	1.0123
MOEA/D	0.9164	0.8741	0.9333	0.8840	0.8809	0.9766	0.9743	1.1529	0.9164	0.8592	0.8289	0.8856	**0.8968**	0.9949	0.8711	0.9063	1.0978	**0.9552**
MOEA/D-PSO	0.8671	0.8996	0.9009	0.9164	0.8371	0.9252	**0.7566**	1.0200	1.0618	0.9090	0.8975	0.9622	0.9378	0.8844	1.0241	0.9837	1.1209	1.0739