Received November 29, 2020, accepted December 18, 2020, date of publication December 22, 2020,
date of current version January 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3046495

Printed Circuit Board Assembly Planning for Multi-Head Gantry SMT Machine Using Multi-Swarm and Discrete Firefly Algorithm

HSIEN-PIN HSU
Department of Supply Chain Management, National Kaohsiung University of Science and Technology, Kaohsiung 81157, Taiwan, R.O.C.
e-mail: hphsu@nkust.edu.tw

This work was supported by the Ministry of Science and Technology (MOST) Taiwan under Grant MOST 109-2410-H-992-020.

ABSTRACT This research proposes a Multi-swarm and Discrete Firefly Algorithm (MDFA) to deal with the component sequencing problem (CSP) and feeder assignment problem (FAP) simultaneously for a multi-head gantry surface-mounted technology (SMT) machine. The MDFA is an improved version of standard Firefly Algorithm (FA) which is a kind of nature-inspired and population-based metaheuristic. To our best knowledge, the FA has never been used to deal with the CSP and FAP simultaneously for the multi-head gantry SMT machine. Empowered by novel features such as multiple swarms as well as adaptive and discrete moving step, in addition to using an exploration-to-exploitation strategy, the MDFA is found capable of searching a solution space effectively. To investigate its effectiveness, the MDFA has been compared to the standard FA, particle swarm optimization (PSO), and genetic algorithm (GA). The experimental results show that the MDFA outperforms the others in terms of assembly time.

INDEX TERMS Firefly algorithm, print circuit board assembly, multi-head gantry SMT machine.

I. INTRODUCTION

Printed Circuit Board Assembly (PCBA) is a process connecting components such as integrated circuit, capacitor, resistor with a bare PCB. It is an essential operation for producing final electronic products.

The PCBA depends on two kinds of operations, insertion and onsertion, which can be conducted by operator manually or machine automatically. The insertion is an operation connecting a component with a PCB through pin holes. In contrast, the onseration is an operation which can attach a component on a PCB directly. The latter appears to be more effective. Usually, a small component uses the onsertion operation performed by machine automatically and some bigger or special components employ the insertion operation performed by operator manually. Due to the large amount of components to be mounted on PCBs, the PCBA tends to become the bottleneck in an assembly line. Expediting this process is essential and automatic machines can play an essential role.

Surface-mounted technology (SMT) machines are capable of onsertion operation. They have been widely used in assembly firms to expedite the PCBA [1]. How to best utilize these machines has become an important issue. Improving the assembly time of one PCB for a SMT machine is critical as a significant benefit can be obtained from a large batch. There are different kinds of SMT machines currently used in industry, such as sequential pick-and-place, multi-head, dual-delivery, turret type, multi-station, and concurrent pick-and-place machines [2], [3]. In this research, the kind of multi-head gantry SMT machine (multi-head and sequential pick-and-place) is focused as they have been widely used in industry [4], [5]. However, optimizing PCBA productivity is a very complicated task due to involving with many problems, such as PCB grouping, PCB-to-machine assignment, component-to-machine assignment, component-to-feeder assignment (FAP), nozzle-to-assembly head assignment (NAP), and component sequencing problem (CSP) [6]. As these problems are impossible to be simultaneously included in one research, this research only takes the CSP and FAP into consideration.

Various approaches have been proposed for dealing with PCBA problems. Integer programming (IP) and Mixed Integer Linear Programming (MILP) are mathematical models which have been often used to find the optimal (exact) solution to a PCBA problem. However, these mathematical models tend to become computationally intractable when the problem is of big size due to NP-complete [6]–[9]. This limit

The associate editor coordinating the review of this manuscript and approving it for publication was Arturo Conde.

the applicability of mathematical models. Other alternative methods are thus required. As an alternative, simple heuristics are found popular in practice due to the advantages: ease of use and computational efficiency. However, simple heuristics are found hard in terms of finding a high-quality solution also owning to their simplicity. Besides the simple heuristics, metaheuristics have been increasingly used to dealing with relevant PCBA problems. This kind of advanced metaheuristics have the capability of driving solutions toward optimality iteratively. While improving the simplicity problem faced by simple heuristics, metaheuristics can avoid computationally intractability problem faced by exact approaches by controlling their parameters. Thus, metaheuristics are focused in this research.

Based on incomplete or imperfect information, metaheuristics can find, generate or select a heuristic to find the optimal/near-optimal solution [10]. Metaheuristics can basically be classified into the four categories: local search-based, evolution-based, swarm intelligence (SI)-based and hybrid. The local search-based metaheuristics finds a better solution by exploring neighborhoods. The evolution-based metaheuristics evolves solutions toward optimality through an evolutionary process. The SI-based metaheuristics depend on a population of entities and their intelligence to find the best solution. This kind of metaheuristics is regarded as one discipline of artificial intelligence which has attracted much attention in recent research. Metaheuristics such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony Algorithm (ABC), Bacterial Foraging Optimization (BFO), Firefly Algorithm (FA), Fish Swarm Optimization (FWO), and Shuffled Frog-Leaping Algorithm (SFLA), Cuckoo Search Algorithm (CS) are examples of this kind of metaheuristic [11]. A hybrid approach hybridizes different approaches and takes advantages from these components.

Evolutionary-based metaheuristics have been used to deal with the PCBA problems. For examples, the [3], [12], [13] used SA whereas the [6], [8], [14]–[16] employed GA/Hybrid GA. The GAs have been used extensively in past research. However, some novel evolutionary and SI-based metaheuristics have been increasingly used to solve PCBA problems in recent years. For instances, for the multi-head gantry machine, ABC has been used to deal with the Automatic Nozzle Change (ANC), Nozzle Assignment Problem (NAP), and Component Sequencing Problem (CSP) [17]; SFLA has been used to deal with the CSP [18]; APSO (adaptive PSO) has been used to deal with the CSP [19]. For the chip shooter machine, PSO2 (an improved PSO) has been used to deal with the FAP and CSP simultaneously [1]; Improved SFLA has been used to deal with the FAP and CSP [20]. However, to our best knowledge, there is still a lack of using FA to deal with the PCBA problems for a multi-head gantry SMT machine.

Proposed by Yang [21], FA is a SI-based metaheuristic which mimics the brightness and attractiveness characteristics of fireflies. The objective is to find the optimal solution in a solution space. The FA compares each pair of fireflies with the brighter firefly attracting the less bright one. The position of a firefly corresponds to a solution in the solution space. The FA has been widely for with various applications, including optimization of loading pattern for nuclear reactor core [22], capacitated facility location problem [23], dynamic multidimensional knapsack problems [24, job shop scheduling problem [25], mechanical design optimization problems [26] big data optimization [27], project scheduling [28], production-distribution network design [29], capacitated vehicle routing problem [30], computer-aided process planning [31], stock market prediction [32], and feature selection [33], [34], etc. The [35] provides details of FA applications. However, the conventional FA tends to fall into the local optima [34], which highlights the necessity of a further improvement. In addition, in terms of application, it is found that the FA has never been employed for dealing with PCBA problems, which provides a room for such research. In this research, an improved FA is proposed with the aim to enhance the searching capability for the conventional FA and apply the improve FA to deal with two PCBA problems for SMT machines.

The main contents and contributions of this work are summarized as follows.

1) An improved FA, termed as Multi-swarm and Discrete Firefly Algorithm (MDFA), is proposed for dealing with the CSP and FAP simultaneously for a multi-head gantry SMT machine. The MDFA includes some novel features such as multi-swarm, adaptive step, and the use of an exploration-to-exploitation strategy. The multi-swarm helps diversify fireflies. The adaptive step enables a firefly to approach an elite effectively. The exploration-to-exploitation strategy gives higher momentum for fireflies to explore the solution space at the earlier iterations. But, at latter iterations higher momentum is given to fireflies to exploit fewer elites. These features enable fireflies to search a solution space effectively. These novel features differ the MDFA from the standard FA.
2) The MDFA has been successfully applied to a new application area, i.e., to deal with the CSP and FAP simultaneously for a multi-head gantry SMT machine. To our best knowledge, FA has never been used for this purpose. To investigate its effectiveness, the MDFA have been compared with other metaheuristics, including standard FA, PSO, and GA. The results showed that the MDFA outperforms the others. In addition, the MDFA has been compared to a lower bound of a MILP. The MILP is specific for the CSP which can be regarded as a traveling sales problem. The results show that the MDFA can find the shortest path (assembly sequence of components) under some small-sized experimental instances, which demonstrates good capability of the MDFA.

The rest of this paper is organized as follows. Section II includes a literature review on the multi-head gantry SMT machine, relevant studies, approaches. Section III includes problems definition and mathematical model formulation. Section IV develops the MDFA. Section V shows the experimental results and discussion. Finally, Section VI has a conclusion and future research directions.

II. LITERATURE REVIEW

A. THE MULTI-HEAD GANTRY SMT MACHINE

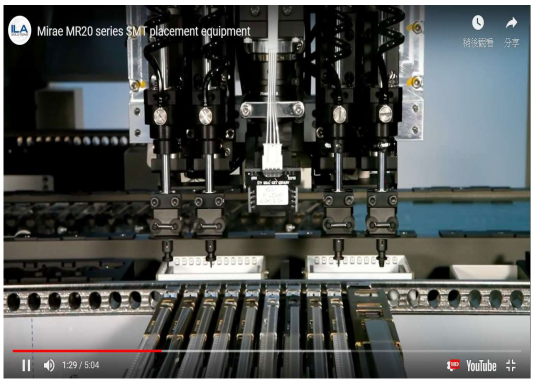
Figure 1(a) shows a physical multi-head gantry SMT machine. The Figure 1(b) shows a schematic drawing of its top view. Typically, a gantry SMT machine includes a feeder carrier for accommodating feeders, a table for fixing the PCB, a gantry for moving robot arm in the horizontal and vertical directions. The robot arm is equipped with a number of assembly heads for picking up and placing components through grippers or nozzles. The components are stored in different feeder positions or on different tray positions. The package and size of a component can determine the use of feeder or tray. To start assembling, assembly heads have to pick up components firstly. The movement of these assembly to pick up components from the feeder or tray forms a fetch cycle. These assembly head picks up components one by one according to planned sequence. After this, the gantry moves the robot arm back to the PCB where the assembly heads start placing components on the PCB. When one placement cycle is completed, the robot arm starts the next fetch cycle of components and then back to the PCB for placement. Above procedure repeats until all components for one PCB are completed.

Typically, a multi-head gantry SMT machine supports different kinds of feeders such as stick feeder, tape feeder and tray carrier to ensure component availability [2]. The stick feeders are used for "sticks" (or "tubes") while the tape feeders are used for reels. Each feeder contains a specific type of component. The tray is used to carry some components not kept in stick, tube or tape and components of small pitch (high precision) are often carried by tray. The physical dimensions of the sticks and reels can determine the type of feeder and the machine used.

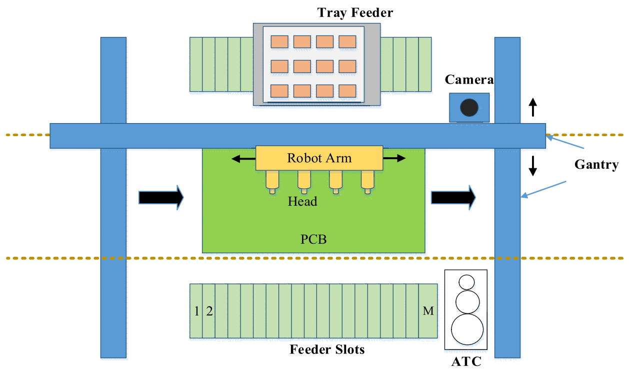
The total assembly time required for one PCB are affected by the assembly sequence of components and the storage positions of these components, corresponding to the CSP and FAP, respectively. To achieve a good result, it needs to deal with the two problems carefully.

B. OPERATIONAL FLOW OF MULTI-HEAD GANTRY SMT MACHINE

A multi-head gantry SMT machine appears to be more effective than a single-head gantry SMT machine as multiple assembly heads can grasp more components in one fetch cycle. The components within a same fetch cycle is defined as a "group." The formation of component groups become critical as it can affect the total assembly time for the completion of one PCB. For example, if the distances of components of a same group are far from each other, it will definitely lead to longer assembly time. In addition, the storage positions



(a)



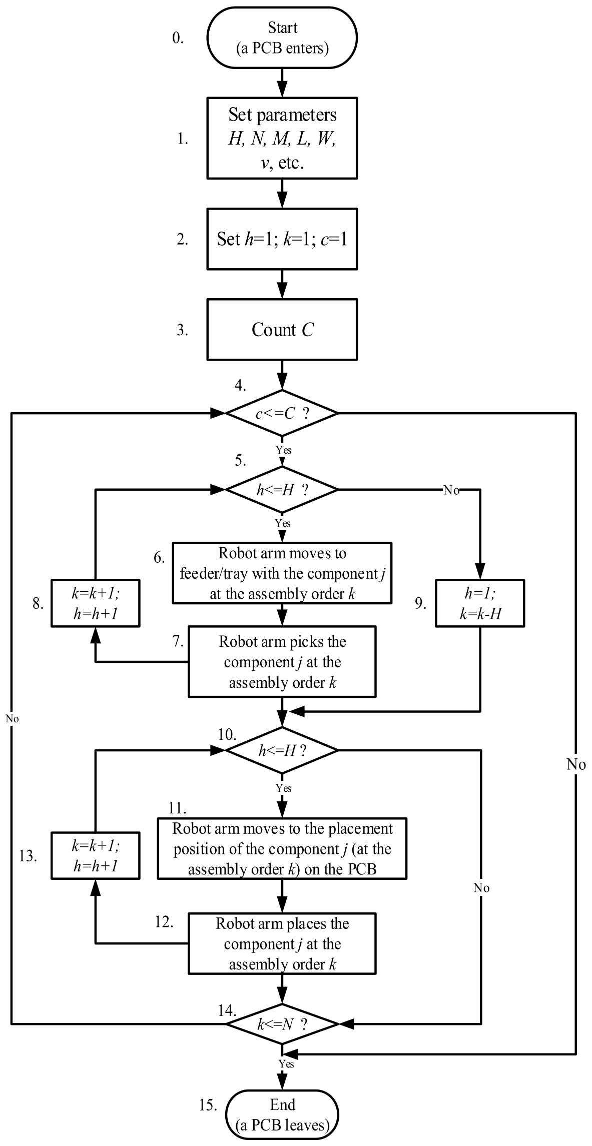
(b)

FIGURE 1. (a) A real multi-head gantry SMT machine. (b) A schematic drawing of top view of multi-head gantry SMT machine.

of the components to be fetched are another factor affecting the assembly time. Obviously, the optimization of the PCBA problem indeed is very complicated.

Fig. 2 shows the operational flow for a multi-head gantry SMT machine (with H assembly heads) to assemble one PCB. Relevant steps are detailed as follows.

1) Given H and N, where H is the total number of assembly heads on the robot arm and N is the number of components to be assembled for one PCB
2) Set $h = 1$, where the h indicates the assembly head No. Set $k = 1$, where k indicates the assembly order of a component to be pick-up. Set $c = 1$, where c indicates the current cycle No. of assembly.
3) Count the total assembly cycle C required for completion N components by using $C = \text{QUOT}(N/H) + 1$, the QUOT function derives the quotient of the division. For example, $C = \text{QUOT}(22/4) + 1 = 6$.
4) If ($c \leq C$) then go to the step 5) else go to step 15)
5) If ($h <= H$) go to step 6) else go to step 9)
6) The robot arm moves to the feeder slot m ($1 < m < M$) containing the component j at the assembly order k and pick one component j. The moving time is denoted as *T1*.
7) The robot arm picks up the component j at the assembly order k. The pick-up time is denoted as *T2*.



0. Start (a PCB enters)
1. Set parameters H, N, M, L, W, v, etc.
2. Set h=1; k=1; c=1
3. Count C
4. c<=C ?
5. h<=H ?
6. Robot arm moves to feeder/tray with the component j at the assembly order k
7. Robot arm picks the component j at the assembly order k
8. k=k+1; h=h+1
9. h=1; k=k-H
10. h<=H ?
11. Robot arm moves to the placement position of the component j (at the assembly order k) on the PCB
12. Robot arm places the component j at the assembly order k
13. k=k+1; h=h+1
14. k<=N ?
15. End (a PCB leaves)
Yes
No

FIGURE 2. The operational flow a multi-head gantry SMT machine.

8) $k = k + 1$; $h = h + 1$; go to step 5)
9) Set $h = 1$; set $k = k - H$; go to step 10)
10) If ($h <= H$) go to step 11) else go to step 13)
11) The robot arm moves to the PCB to the placement position of the component j at the assembly order k. The moving time is denoted as *T3*.
12) The robot arm places the component j at the assembly order k. The placement time is denoted as *T4*.
13) $k = k + 1$; $h = h + 1$; go to step 10)
14) If ($k < N$) then go to step 4) else go to step 15)
15) Stop; the PCB is completed and moves to the next station

The actual operational time required for a component j is dependent on its assembly order and its storage positions in either feeder or on tray. A mathematical model for dealing with the FAP and CSP of the multi-head gantry SMT machine is formulated in Section III.*B*.

C. RELEVANT STUDIES

Fig.3 shows approaches for dealing with PCBA problems, including mathematical model, heuristic, and metaheuristic. The mathematical models are traditional approach for dealing with the PCBA problems. The [7] uses a MILP model to deal with PCBA problems, including component-to-machine allocation problem and CSP. Two strategies were proposed. One is C strategy and another is P strategy. The C strategy aims to minimize the total number of changeovers whereas the P strategy aims to minimize the assembly time. Real company data were used to investigate the two strategies and the results showed the applicability of the two proposed strategies. The [8] developed MILPs to solve the NAP, FAP, and CSP simultaneously for a multi-head SMT machine. A two-stage procedure was proposed. A MILP is firstly used to solve the NAP in the first stage. Then, in the second stage, another MILP was used to find the solution to the FAP and CSP simultaneously. The main objective was to minimum the total assembly time. However, the proposed MILPs can only deal with small-size problems due to NP-hard.

Heuristics have been used in some other PCBA studies. The [4] proposed a heuristic to deal with the CSP and FAP. With a grouping concept, the heuristic appears to take advantage of the nature structure of a PCB. That study considered physical constraints of PCB. Simulation experiments were performed to validate the effectiveness of this heuristic. The [36] used constructive heuristics to deal with the FAP and CSP simultaneously. In that heuristic components of similar speed and turret rotation speed tare grouped together. Penalties were applied in cases of feeder carrier movements, changes in turret rotation speed and changes in PCB table speed. An initial solution to the FAP and CSP was first derived, then a two-opt heuristic was used to find a better solution. This heuristic can find a final solution that closes to a lower bound. However, the required computational time increases along with the problem size. The [37] formulated a MILP model to deal with the FAP and CSP simultaneously for a sequential pick-and-place machine. The MILP determines an assembly sequence for components and their storage slots. Some heuristics were also investigated for solving the two problems. Heuristics such as nearest neighbor, nearest insertion, furthest insertion, and random generation were used to initialize an assembly sequence. On the other hand, heuristics such as 2- opt, 3-opt, and Or-opt were used to improve the initial assembly sequence. Simulation results showed the heuristic approach was able to improve the current heuristic used in a factory by 25%.

Metaheuristics have been increasingly used to deal with PCBA problems. The [3] proposed an iterative approach to deal with the CSP and FAP simultaneously for a pick-and-place SMT machine with 4 assembly heads. The iterative approach includes SA and heuristic. The CSP was treated as sequence dependent traveling salesman problem while the FAP was treated as the Quadratic Assignment Problem. Having compared to the SA and ABC, the experimental

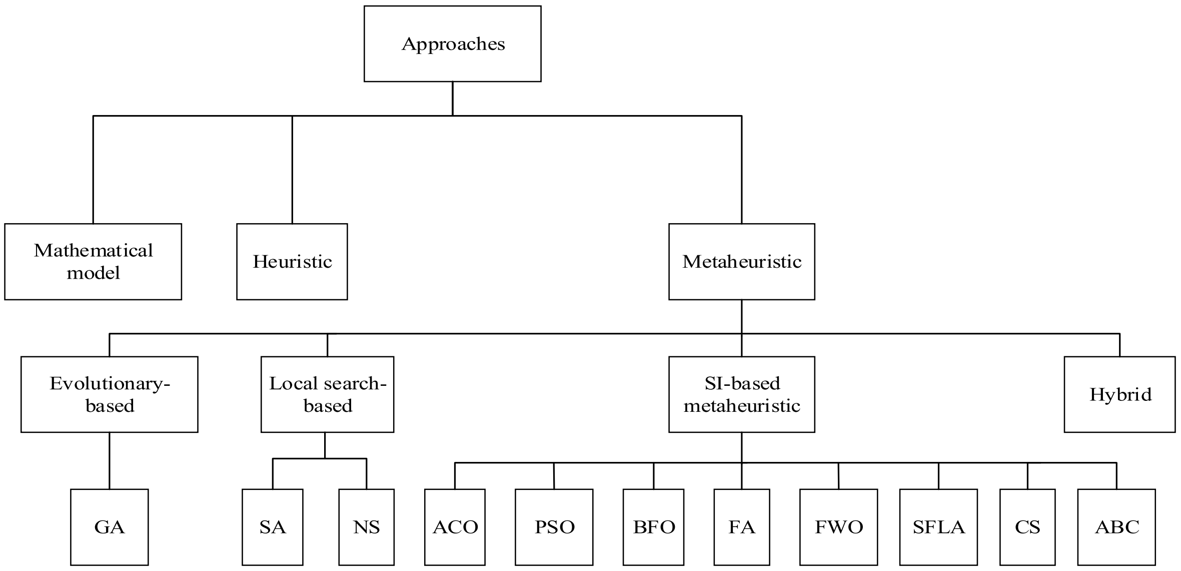


FIGURE 3. **Approaches available for solving PCBA problems.**

results are in favor of the iterative approach. The [6] employed a GA to determine the assembly time for PCBA. The [13] proposed a hybrid genetic algorithm (HGA) for solving the CSP and FAP simultaneously for a chip shooter machine. The minimization of total assembly time was focused. Different heuristics including nearest-neighbor heuristic and 2-opt heuristic were used. The experimental results showed that the HGA performed good in terms of in terms of assembly time. The [15] proposed another HGA for solving the CSP and FAP for a chip shooter machine. In addition, different heuristics such as nearest neighbor heuristic, 2-opt heuristic and an iterative swap procedure were used. The authors concluded the HGA outperformed the standard GA. The [19] optimized the CSP and FAP simultaneously for a pick-and-place machine by using an adaptive PSO. The objective was to minimize the traveling time of assembly heads and change time of nozzles. The adaptive PSO hybridizes PSO with some heuristics for assembly head assignment and reel grouping. The experimental results showed the adaptive PSO outperformed GA. The [14] also developed HGA to deal with the FAP and CSP simultaneously. In that study, components were first into "key" or "free" type, with their relationships being stored in a matrix. Then, the HGA was applied to solve the two problems. The HGA was found effective. The [18] proposed an improved Shuffled Frog-leaping Algorithm (SFLA) to solve the CSP for a gantry multi-head SMT machine. The SFLA allows all frogs to take part in memetic evolution as well as variation. The experiment results showed that the improved SFLA outperforms SFLA and GA in terms of convergence accuracy at a higher computational cost. The [1] proposed an improved PSO to solve the FAP and CSP simultaneously for a chip shooter machine. The PSO2 was found to outperform PSO and two GAs.

New metaheuristics such as ABC, BFO, SFLA, and FWO have been rarely used to deal with the PCBA problems, especially in dealing with the CSP and FAP simultaneously for a chip shooter machine. The [17] proposed a modified ABC algorithm (MABCA) to optimize PCB production. The objective was to maximize throughput and minimize total assembly time. That study considered the automatic nozzle changer (ANC) assignment problem, NAP, and CSP which were solved by a three-phase procedure. First, a proportional distribution method was used for the optimization of ANC assignment. Then, the NAP was solved by taking factors including component height, component thick, and component shape restrictions into consideration. Finally, the CSP was solved by using a 2-opt algorithm with constraints including placement and predetermined nozzle assignment taking into account. Experiments results showed good results obtained from the MABCA in terms of assembly time. The [20] proposed an improved SFLA (I-SFLA2) to deal with the FAP and CSP simultaneously for a chip shooter. The experiments showed that the I-SFLA2 outperformed the PSO2 proposed in [1] as well as the improved SFLA proposed in [18].

III. MATHEMATICAL MODEL

A. PROBLEM DEFINITIONS

The CSP and FAP of a multi-head gantry SMT machine are defined as follows.

Definition 1 (Component Sequencing Problem (CSP)): Given a set of components ($j = 1$ to N, where N is the total number of components) and their placement locations on a PCB, the CSP is a problem of finding a placement sequence of components, with each component j being assigned with a placement order k ($k \in [1, \ldots, N]$).

Definition 2 (Feeder Assignment Problem (FAP)): Given a set of feeder positions (M1 $= [1, \ldots, m1]$, where $m1$ is the total number of feeder positions) and a set of tray positions (M2 $= [1, \ldots, m2]$, where $m2$ is the total number of tray positions on a multi-head gantry SMT machine, the FAP is a problem of assigning each type of component to a feeder slot f ($f \in$ M1) or a tray position p ($p \in$ M2) to this machine. Each feeder/tray position contains one type of component only.

Definition 3 (Simultaneous CSP and FAP): Given a set of components ($j = 1$ to N, where N is the total number of components), the placement location of each component, a set of feeder positions (M1) and a set of tray positions (M2) of a multi-head gantry SMT machine, the simultaneous CSP and FAP is a problem of determining a placement sequence for these components. Each component is assigned with an assembly order k ($k \in [1, \ldots, N]$) and each component is assigned with a feeder/tray position f ($f \in$ M1 $\cup$ M2). Each feeder/ tray position contains one type of component only.

B. MATHEMATICAL MODEL FORMULATION

Referring to [1], [20], and [38] this section formulates a mathematical model of simultaneous CSP and FAP for a multi-head gantry SMT machine. To this end, relevant assumptions and notations are made firstly.

1) ASSUMPTIONS

1) The type of each component is known before assembly.
2) Each component type is only assigned to one feeder and vice versa.
3) The X–Y table moves at a constant speed in the x and y direction.
4) The assembly starts from the original point (0,0).

2) NOTATIONS

a: INDICES

i,j	a component number ($1 \leq i, j \leq N$)
f	a feeder slot position number
t	the component type of component j; $t \in \{1, M\}$
k	an assembly order of a component ($1 \leq k \leq N$)

b: SETS

M	The set of feeder numbers and tray position; M = M1 $\cup$ M2.
M1	the set of feeder numbers; M1 = $\{1, \ldots, m1\}$ where $m1$ is total number of feeder slots in the feeder carrier
M2	the set of tray position numbers; M2 = $\{1, \ldots, m2\}$ where $m2$ is total number of tray positions in the tray

c: PARAMETER

N	total number of components to be mounted
M	total number of component types, i.e., the total number of feeder slots plus tray positions available for the machine.
H	total number of assembly heads
v	the average moving speed of robot arm
t_f	The time for a robot arm moving to the next feeder/tray position

d: OPERATION TIMES

$T0_j$	the average round trip time for robot arm moving to feeder/tray and return
$T1_j$	The time for the robot arm to move from the storage position of component i to the storage position the component j (variable time). The distance is denoted as $D1(i, j)$.
$T2_j$	The time for an assembly head to pick up a component (fixed time)
$T3_j$	The travel time for the robot arm to move from the placement location of component i to the placement position of component j (variable time). The distance is denoted as $D3(i, j)$.
$T4_j$	The total time for an assembly head to mount a component (fixed time).
$T5_j$	The additional average time switch between M1 (feeder slots) and M2 (tray).

e: DECISION VARIABLES

X_{ij}	=1 if component j is assigned with the k assembly order and component i is assigned with assembly order k-1; =0, otherwise
Y_{tf}	=1 if component type t is assigned to the feeder f; =0, otherwise
Z_{ij}	the number of components remains to assign after leaving node i and before entering node j in terms of network analysis.
T_j	the assembly time of component j.

The entire placement process can be regarded as a series of pick-and-place cycles. The time for the robot arm moving to the feeder containing the component j at the assembly order k (i.e. the $T1_j$) is estimated by (1).

$$T1_j = \begin{cases} T0 + D1\,(i,j) \cdot t_f, & \text{if Mod } (k+H, H) = 1 \\ D1\,(i,j) \cdot t_f, & \text{if Mod } (k+H, H) \neq 1 \\ & \quad \text{and } (i, j \in \text{M1}) \\ & \quad \textit{or } (i, j \in M2) \\ T5 + 1\,(i,j) \cdot t_f, & \text{if Mod } (k+H, H) \neq 1 \text{ and} \\ & \quad (i \in \text{M1} \textit{ and } j \in \text{M2}) \\ & \quad \textit{or } (i \in \text{M2} \textit{ and } j \in \text{M1}) \end{cases} \tag{1}$$

There are three cases for determining the $T1_j$. In the first case, the Mod is a function acquiring the remainder of a division operation. If Mod $(k + H, H) = 1$ it indicates that the component j (at the assembly k) is the 1st component, thus an average round trip time ($T0$) is added additionally. For example, given $H = 4$ and $k = 1$ for the component j, the Mod $(k + H, H) =$ Mod $(1 + 4, 4) = 1$ so that the component j is identified as a first component. The $D1(i, j)$ is the distance between the feeder slots of component i and j, which is estimated by (2).

$$D1\,(i,j) = |p\,(i) - p\,(j)| \tag{2}$$

where $p(i)$ and $p(j)$ are feeder/tray positions of component i and j, respectively. The $p(0)$ is set to 0.

In the second case, the $\mathrm{Mod}(k+H, H) \neq 1$ which indicates that the component j is not the 1st component in a component fetch cycle and the condition, $i, j \in \mathrm{M1}$ *or* $i, j \in \mathrm{M2}$, means both components i and j are in either feeder or tray zone. In the third case, the condition, $(i \in M1$ and $j \in \mathrm{M2})$ *or* $(i \in \mathrm{M2}$ and $j \in \mathrm{M1})$, means that component i and and j are in different storage zones, thus an average switch time ($T5$) is required additionally. The $T3_j$ is a variable time estimated by (3).

$$T3_j = D3(i,j)/v \tag{3}$$

As the robot arm of a gantry SMT placement machine can move along X and Y direction simultaneously, we use chebychev measurement to calculate the distance of any two placement positions of components. Given $\mathrm{P}_i = (\mathrm{x1}, \mathrm{y1})$, $\mathrm{P}_{j'} = (\mathrm{x2}, \mathrm{y2})$ as two placement positions for components i (at the assembly order k-1) and j (at the assembly order k), respectively. The distance between the two components is determined by (4).

$$D3(i,j) = \sqrt{(\mathrm{x1}-\mathrm{x2})^2 + (\mathrm{y1}-\mathrm{y2})^2} \tag{4}$$

The total assembly time required for the component j is determined by (5).

$$T_j = T0_j + T1_j + T2_j + T3_j + T4_j + T5_j \tag{5}$$

The mathematical model for the problems is formulated as follows.

$$Min\ Z = \sum_{j=1}^{N} \sum_{k=1}^{N} T_j X_{ij} \tag{6}$$

$$s.t.\ T_j \geq 0 \quad \forall 1 \leq j \leq N \tag{7}$$

$$\sum_{i=1}^{N} X_{ij} = 1 \quad \forall 1 \leq j \leq N \tag{8}$$

$$\sum_{j=1}^{N} X_{ij} = 1 \quad \forall 1 \leq i \leq N \tag{9}$$

$$\sum_{t=1}^{TP} Y_{tf} = 1 \quad \forall 1 \leq f \leq M \tag{10}$$

$$\sum_{f=1}^{M} Y_{tf} = 1 \quad \forall 1 \leq t \leq M \tag{11}$$

$$Z_{ij} \geq 0 \quad \forall 1 \leq i,j \leq N \tag{12}$$

$$Z_{ij} \leq (N-1)X_{ij} \quad \forall 1 \leq i,j \leq N \tag{13}$$

$$\sum_{j=1}^{N} Z_{ji} = \sum_{j=1}^{N} Z_{ji} + 1 \quad \forall 1 \leq i \leq N \tag{14}$$

$$X_{ij}, Y_{tf}, \quad Z_{ij} \in \{0,1\}\, \forall 1 \leq i,j,k \leq N; \quad \forall 1 \leq t,f \leq M \tag{15}$$

Equation (6) is the objective function minimizing the total assembly time for one PCB. Constraint (7) defines the condition for the variable T_j. Constraints (8) and (9) relate to the variables X_{ij}. Constraint (8) assures that each component is assigned with one order only. Constraint (9) ensures each component is mounted once only. Constraint (10) and (11) associate with the variable Y_{tf}. Constraint (10) requires that each feeder slot stores one type of component only. Constraint (11) stipulates that each component type is assigned to exactly one feeder slot. Constraints (12), (13) and (14) associate with the variable Z_{ij} which assigns sequence of components through a network perspective. Inequality (12) defines that unassigned number of components ≥ 0 when leaving the node i and entering the node j. Inequality (13) states that on leaving a node i it is sufficient to have not more than $N-1$ component to be assigned. Equation (14) is a conservation constraint for node i. Constraint (15) stipulates that X_{ij}, Y_{tf}, and Z_{ij} are binary variables.

IV. DEVELOPMENT OF THE MULTI-SWARM AND DISCRETE FIREFLY ALGORITHM (MDFA)

A. STANDARD FA

Proposed by Yang [21], FA mimics the behavior of fireflies. Each position of a firefly corresponds to a solution in the solution space. Through signal radiation, a brighter firefly can attract a less-bright one. The FA defines the three basic features.

- **Brightness**

Equation (16) defines the brightness of a firefly, which is a value depends on the distance between two fireflies and the atmospheric absorption coefficient. Each firefly has a brightness.

$$I(r) = I_0 e^{-\gamma r^2} \tag{16}$$

where

I:	is the intensity of the light source, γ is the absorption coefficient.
r:	is the distance between the two considered fireflies
γ:	is a given medium with a fixed light absorption coefficient.
I_0:	is the intensity of the light source when $r = 0$.

- **Attractiveness**

Equation (17) is used to calculate the attractiveness of a firefly.

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{17}$$

where

β_0: is the attractiveness of the firefly when $r = 0$.

- **The next position**

For each pair of fireflies a less-bright firefly is attracted by a brighter one and (18) is used to determine the next position for an attracted one.

$$X_j(t+1) = X_j(t) + \beta_0 e^{-\gamma r^2}\left(X_i(t) - X_j(t)\right) + \alpha R(-0.5, 0.5) \tag{18}$$

where

α:	is the mutation coefficient which is generally a self-adaptive parameter decreasing through iterations
R:	is a normal randomized number between $(-0.5, 0.5)$.

In the right hand side of (18), the second term is due to attraction and the third term is a randomness.

TABLE 1. The pseudo code of the standard FA.

```
Define the problem // for a minimize problem
Define the objective function F(X)
Initialize a population of fireflies ($F_j$, $j$=1,…,$N$)
Define the light absorption coefficients, $I_0$, $\gamma$, and $\beta_0$
REPEAT (termination condition is not satisfied)
For $i$=1 to $N$
  For $j$=1 to $N$
    If (F($X_i$) < F($X_j$))
      {
       Move firefly $F_j$ towards $F_i$ using (18)
      }
    End IF
     Varying attractiveness using (17)
     Assess new solutions and light intensity (16)
     Memorize the best solution found so far
   End for $j$
End for $i$
Rank the fireflies and find the best one denoted as $g$
End REPEAT (if termination condition is satisfied)
```

The pseudo code of the FA standard is shown in Table 1 [39].

- **Multiple swarms scheme**

Unlike the standard FA that contains all fireflies within one single group, the MDFA separates fireflies into multiple swarms so that a firefly can be attracted and affected more elites. One is the best one of a same swarm and another us the global one in this population. Equation (19) is used to determine the number of swarms at the iteration t.

$$S(t) = \mathrm{R}(\sqrt{P} + \sqrt{P}/2 - t \cdot \sqrt{P}/T) \tag{19}$$

where R is a function rounding a real value to a nearest integer. For example, R(5.8) = 6 and R(3.2) = 3.

- **The use of exploration-to-exploitation strategy**

The FA is found likely to face the problem of oscillatory behavior when dealing with a combinatorial optimization problem, which means that the FA can find the solution faster in some cases but this is not true in other cases [35]. The balance of exploration and exploitation during the search process for FA is still an open issue [35].

To this issue, the MDFA uses an exploration-to-exploitation strategy which is implemented by (19). This formula determines the number of swarms at each iteration. In fact, it decreases the number of swarms when increasing the iteration. For example, given $P = 100$ (the total number of fireflies) and $T = 100$ (the total number of iterations), the number of swarms at the 1st iteration obtained from the (1) is $S(t = 1) = 14$, but at the last iteration, $t = 100$, the derived number of swarms is $S(t = 100) = 2$. This means that fireflies are deployed to search 14 areas initially but finally they are deployed to search 2 areas only. This also means that at the 1st iteration fireflies have the maximum momentum to explore the solution space but at the last iteration the maximum momentum are used to exploit fewer elite fireflies. At the half iteration, $t = 50$, the momentum for exploration and exploitation is balance.

- **Transformation from real values to discrete values**

Using FA to deal with combinatorial optimization problems requires a mapping between the discrete variables (usually permutation of integer) in the problem space and the continuous variables in the search space where the firefly operators (like moving fireflies) act [35]. In this research, the ROV technique is used to transform real values in the continuous variables to discrete integers in the solution space.

- **The development of next moving step for fireflies**

The adaptation of FA parameters to changing environment is expected to improve the solutions found quickly [35]. This is also an open issue. To this issue, the MDFA uses adaptive step to move fireflies.

Unlike the standard FA, the MDFA uses a different moving step for a firefly. Assume that $X_i(t) = [X_{i,k}(t); k = 1, \ldots, D]$ and $X_j(t) = [X_{j,k}(t); k = 1, \ldots, D]$ are two positions vectors of the fireflies i and j with the firefly i attracting the firefly j, the adaptive moving step of the firefly fly j is then defined by (20).

$$\Delta D_{i,j}(t) = D_{i,j}(t) \oplus AV_j(t), \tag{20}$$

The $D_{i,j}(t)$ is termed as *total distance vector* and the $AV_j(t)$ is termed as *adaptive binary vector*. The $D_{ij}(t)$ is a vector indicating the total distance between the fireflies i and j. It is determined by (21).

$$D_{i,j}(t) = [X_{i,k}(t) \sim X_{j,k}(t); k = 1, \ldots, D] \tag{21}$$

where the operator "$\sim$" works as (22).

$$X_{i,k}(t) \sim X_{j,k}(t) = \begin{cases} 0, & \text{if } X_{i,k}(t) = X_{j,k}(t) \\ X_{i,k}(t), & \text{if } X_{i,k}(t) \neq X_{j,k}(t) \end{cases} \quad \text{for the } k\text{-th element,} \tag{22}$$

The $AV_j(t)$ is determined by Equation (23).

$$AV_{j,k}(t) = \begin{cases} 0, & \text{if } R() \geq R1_{i,j}(t); \\ 1, & \text{if } R() < R1_{i,j}(t); \end{cases} \tag{23}$$

the $R()$ is a random number and the $R1_{i,j}(t)$ is a value controlling the probability for the generation of binary value 1. Both $R(\,)$ and $R1_{i,j}(t)$ are within the range [0,1]. The $R1_{i,j}(t)$ is determined by (24).

$$R1_{i,j}(t) = \begin{cases} \dfrac{\Delta D_{i,j}(t) - 2}{D}, & \text{if } \Delta D_{i,j}(t) > 2; \\ 0, & \text{if } \Delta D_{i,j}(t) \leq 2; \end{cases} \tag{24}$$

The $\Delta D_{i,j}(t)$ is defined in (24). Finally, the operator "$\oplus$" works as (25).

$$D_{i,j}(t) \oplus AV_{j,k}(t) = \begin{cases} D_{i,j}(t), & \text{if } AV_{j,k}(t) = 1 \\ 0, & \text{if } AV_{j,k}(t) = 0 \end{cases} \quad \text{for the } k\text{-th element,} \tag{25}$$

TABLE 2. The logic flow of the MDFA.

```
Define the problem // for a minimize problem
Define the objective function Z(X)
Initialize parameter values for T, N
Initialize a population of fireflies ($F_j$, j=1,…,N)
Define the light absorption coefficients, $I_0$, $\gamma$, and $\beta_0$
For t=1 to T
    Calculate the total number of swarms S(t) at the iteration t using (19)
    Assign firefly to swarms based on fireflies' rankings
    For g=1 to S(t) // parallel and decreasing groups
        For j=1 to N(S(t)) // N(S(t)) is the number of fireflies in the swarm
              S(t) at iteration t
         Move firefly $F_j$ toward the best firefly $F^*_{S(t)}$ of a same swarm
         If $(Z(X(t)_j) < Z(X(t+1)_j))$ // not improved{
            Move firefly $F_j$ towards the global best firefly $F_g^*$ using (26)
            If $(Z(X(t+1)_j) < Z(X(t)_j))$ // improved
            {
             Accept this position $X(t+1)_j$
            }
            Else // not improved
            {
              Give a random fly for firefly $F_j$
            }
           End IF
        }
        ELSE // improved
        {
         Accept this position $X(t+1)_j$
        }
        End IF
        Varying attractiveness
        Assess new solutions and light intensity
        Memorize the best solution found so far
       End for j
    End for g
    Rank the fireflies and find the global best $F_g^*$
End for t (if termination condition is satisfied)
```

- **The next position**

To determine the next position of the particle j, (26) is used.

$$X_j(t+1) = X_j(t) \odot \Delta D_{i,j}(t), \tag{26}$$

The operator "$\odot$" works with the following steps:

1) Take the first non-zero value out of the $\Delta D_{i,j}(t)$ and replace the value at the same position in the vector $X_j(t)$. Following this, the replaced value takes the position of the non-zero value in the $X_j(t)$.
2) Repeat above step until there is no non-zero value in the $\Delta D_{i,j}(t)$.
3) Copy the current $X_j(t)$ as the $X_j(t+1)$.

B. THE MAIN LOGIC FLOW OF MDFA

The main logic flow of MDFA is different from the FA's logic flow (Table 1). The logic flow of the MDFA is shown in Table 2.

V. EXPERIMENTS

More experiments have been conducted to compare the MDFA with FA, GA, PSO as well as a lower bound of mathematical model (MILP) under different problem sizes including $10 \times 5 \times 5$, $20 \times 5 \times 5$, $40 \times 5 \times 5$, and $60 \times 5 \times 5$.

TABLE 3. The parameter settings for different approaches.

Method	FA	GA	PSO	MDFA
Parameter values				
N	10,20,40,60	10,20,40,80	10,20,40,60	10,20,40,60
M1	5	5	5	5
M2	5	5	5	5
P	120	120	120	120
n=P/m	-	-	10	*P/m(t)*
n_ls	-	-	-	2
T (iterations)	1	1,000	1,000	500
S(t)	-	-	-	Eq. (19)

-: not used

A. PARAMETER SETTING

Parameter values involving with PCB, multi-head SMT machine are set as follows. For the PCB, $W = 40$ cm (the width) and $L = 90$ (the length) are used. For the multi-head gantry SMT machine, $T0 = 1$, $T2 = 0.2$, $T4 = 0.2$, and $T5 = 2$ (s); $v = 5$ cm/s $H = 4$, $M1 = 5$; $M2 = 5$; $t_f = 1$ (s) (moving speed between feeder slots/tray positions) are used.

Table 3 shows the parameter values setting for these comparing approaches. The N indicates the total number of containers; the P indicates the total number of particles; the *n_ls* indicates the total number of local searches for particles; T indicates the total number of iterations). In the SGPSO, the (19) is used to determine the $S(t)$ which indicates the total number of swarms; $P/S(t)$ is the total number of particles in a swarm; the number 0.5 is set for θ (the probability of mutation). A problem size is defined as $N \times m1 \times m2$.

B. AN ILLUSTRATION OF SMALL-SIZE EXAMPLE

In this section we demonstrate the best result obtained from the MDFA for a small size of problem $10 \times 5 \times 5$ (Table 4).

In Table 4, the k in the 1st row indicates the assembly order of component j; the j in the 2nd row is component id, the $t(j)$ in the 3rd row indicates the type of component j; the x_j in the 4th row indicates the x coordinate of component j; the y_j in the 5th row indicates the y coordinate of component j; the $T1$ in 6th row indicates the pick-up times of components from feeders/tray before placing the component j; the $T3$ in the 7th indicates the of component j from feeder/tray; The T_j in the 8th row is the total process time including $T1$ and $T2$; the B_j in the 9th row is the start assembling time of component j; the E_j in the 10th row is the end assembly time of component j.

The MDFA finds the best solution with the sequence 5-10-2-1-3-6-8-7-9-4 of components. The component 1 is assigned to tray position 9; the components 2,3 and 8 are assigned to feeder 2; the component 4 is assigned to tray position 6; the component 5 is assigned to feeder 4; the component 6 is assigned to tray position 7; the component 7 is assigned to feeder 1; the components 9 and 10 are assigned to feeder 3.

C. EXPERIMENTAL RESULTS OF BIG EXPERIMENTAL INSTANCES

Table 5 shows the experimental results of big experimental instances. The findings are summarized as follows.

TABLE 4. The best result found by different approaches.

FA

k	1	2	3	4	5	6	7	8	9	10
j	9	10	8	7	6	4	5	3	2	1
$t(j)$	3	3	2	1	7	6	4	2	2	9
x_j	6	7	21	24	33	3	1	35	15	20
y_j	9	69	24	14	41	37	67	74	79	86
$T1$	15.7	12	9.4	2.1	5.7	6.1	6	6.9	4.1	1.7
$T3$	1.5	0	0	0	6.3	0	0	0	3	0
T_j	16.1	12.4	9.8	2.5	6.1	6.5	6.4	7.3	4.5	2.1
B_j	1.5	16.1	28.4	38.3	47.1	46.9	53.3	59.7	70.1	71.6
E_j	16.1	28.4	38.3	40.8	46.9	53.3	59.7	67.1	71.6	73.7

MDFA

k	1	2	3	4	5	6	7	8	9	10
j	5	10	2	1	3	6	8	7	9	4
$t(j)$	4	3	2	9	2	7	2	1	3	6
x_j	1	7	15	20	35	33	21	24	6	3
y_j	67	69	79	86	74	41	24	14	9	37
$T1$	6	1.3	2.6	1.7	3.8	6.6	4.2	2.1	3.7	5.6
$T3$	3.8	0	0	0	7.6	0	0	0	3.3	0
T_j	6.4	1.7	3	2.1	4.2	7	4.6	2.5	4.1	6
B_j	3.8	6.4	8.1	11	20.8	17.4	24.4	29	34.8	35.6
E_j	6.4	8.1	11	13.2	17.4	24.4	29	31.4	35.6	41.6

GA

k	1	2	3	4	5	6	7	8	9	10
j	5	1	2	10	6	3	4	9	8	7
$t(j)$	4	9	2	3	7	2	6	3	2	1
x_j	1	20	15	7	33	35	3	6	21	24
y_j	67	86	79	69	41	74	37	9	24	14
$T1$	11.6	5.4	1.7	2.6	7.6	6.6	9.8	5.6	4.2	2.1
$T3$	5.2	0	0	0	9	0	0	0	1.7	0
T_j	12	5.8	2.1	3	8	7	10.2	6	4.6	2.5
B_j	5.2	12	17.7	19.9	31.8	30.9	37.9	48	55.8	58.7
E_j	12	17.7	19.9	22.8	30.9	37.9	48	54.1	58.7	61.2

PSO

k	1	2	3	4	5	6	7	8	9	10
j	5	2	10	1	3	6	8	9	7	4
$t(j)$	4	2	3	9	2	7	2	3	1	6
x_j	1	15	7	20	35	33	21	6	24	3
y_j	67	79	69	86	74	41	24	9	14	37
$T1$	6	3.7	2.6	4.3	3.8	6.6	4.2	4.2	3.7	6.2
$T3$	3.9	0	0	0	7.3	0	0	0	3.1	0
T_j	6.4	4.1	3	4.7	4.2	7	4.6	4.6	4.1	6.6
B_j	3.9	6.4	10.5	13.5	25.4	22.4	29.4	34	41.7	42.7
E_j	6.4	10.5	13.5	18.1	22.4	29.4	34	38.6	42.7	49.4

- At the problem size $10 \times 5 \times 5$, the MDFA has achieved the best performance in terms of Z. The average gap over the PSO is 26.5%; 22.6% over GA, and 44.3 % over FA.
- At the problem size $20 \times 5 \times 5$, the MDFA extends its advantage. The average gap over the PSO is 90.2%; 91.9% over GA; and 102.3 % over FA.
- At the problem size $40 \times 5 \times 5$, the MDFA continues to extend its advantage. The average gap increases up to 137% over the PSO; 138.9% over GA; and 152.0 % over FA.
- At the problem size $60 \times 5 \times 5$, the MDFA continues to extend its advantage. The average gap increases up to 163.8% over the PSO; 164.3% over GA; and 176.3 % over FA.

D. STATISTICALLY TEST

Referring to [40], [41], the experimental results in Table 6 are investigated by using t-test at the significance level $\alpha = 0.05$.

For t-test, Ho is set as the null hypothesis assuming that the average Z values of these comparing approaches are equal in terms of assembly time; H1 is the hypothesis assuming that the average Z values of these comparing approaches are not equal. Table 6 shows the t-test statistical testing results of these comparing approaches against the MDFA. In this table, the Y+ means that the MDFA significantly outperforms the comparing algorithm at this significance level; Y- indicates that the MDFA is significantly inferior to the comparing algorithm; N denotes there is no difference between the MDFA and the comparing algorithm. The p-values obtained from the t-test are found all less than 0.005. For example, the p-values of the FA, GA and PSO at the problem size $10 \times 5 \times 5$ are 0.0000118444, 0.0004929538, and 0.0000036408, respectively. As a result, the H0 is rejected and H1 is accepted. Therefore, it is concluded that the proposed MDPA is statistically superior to the other three approaches. The statistical results demonstrate the robustness of the proposed MDFA in the statistical sense.

TABLE 5. The experimental results of different problem sizes.

	FA			GA			PSO			MDFA	
10x5x5	Z	T	G	Z	T	G	Z	T	G	Z	T
1	68.5	0.07	67.5	51.5	12	25.9	52.9	10	29.3	40.9	25
2	56.8	0.07	56.0	40.4	12	11.0	39.9	10	9.6	36.4	25
3	47.4	0.07	21.5	44.4	12	13.8	45.1	10	15.6	39	25
4	61.4	0.07	50.5	47.6	12	16.7	53.2	10	30.4	40.8	25
5	59.7	0.07	50.4	56.5	12	42.3	52.4	9	32.0	39.7	26
6	53.2	0.07	20.9	51.2	12	16.4	56.3	10	28.0	44	25
7	63.1	0.07	67.4	54.2	12	43.8	46.4	10	23.1	37.7	25
8	53.6	0.07	30.7	54.7	12	33.4	52.9	10	29.0	41	25
9	54.6	0.07	44.4	38.5	12	1.9	51.6	10	36.5	37.8	24
10	54	0.07	37.8	47.2	12	20.4	50.9	10	29.8	39.2	25
Avg.	57.2	0.1	44.3	48.6	12	22.6	50.2	10	26.5	39.7	25
20x5x5											
1	112.6	37	79.6	102	19	62.7	126.5	20	101.8	62.7	40
2	134	37	157.2	113.1	18	117.1	100.7	20	93.3	52.1	41
3	104.3	37	81.4	118	19	105.2	109.4	20	90.3	57.5	40
4	113.1	37	107.9	102.4	19	88.2	100.8	20	85.3	54.4	41
5	128.5	37	142.5	111.8	19	110.9	109.4	20	106.4	53	41
6	110.9	37	114.1	95	19	83.4	94.7	20	82.8	51.8	41
7	115.5	37	87.2	124.5	19	101.8	114.6	20	85.7	61.7	42
8	109.8	37	94.0	106.2	17	87.6	110.7	20	95.6	56.6	41
9	99.7	37	78.7	97.1	19	74.0	93.4	20	67.4	55.8	41
10	111	37	92.4	111	19	92.4	111	20	92.4	57.7	41
Avg.	113.9	37.0	102.3	108.1	19	91.9	107.1	20	90.2	56.3	41
40x5x5											
1	265.1	0.3	124.1	248.4	37	110.0	267.5	36	126.1	118.3	77
2	248.9	0.3	162.6	243.9	37	157.3	231.6	36	144.3	94.8	76
3	247.4	0.3	201.7	259.1	36	216.0	258.7	36	215.5	82	77
4	259.5	0.3	185.8	242.9	37	167.5	250.6	36	176.0	90.8	78
5	231.4	0.3	146.2	214.1	37	127.8	221	35	135.1	94	77
6	211.5	0.3	151.2	199	37	136.3	205.2	36	143.7	84.2	77
7	240	0.3	124.7	240.3	37	125.0	219.8	35	105.8	106.8	77
8	271.4	0.3	160.7	237	35	127.7	232.1	36	123.0	104.1	77
9	269.2	0.3	141.0	247.8	37	121.8	237.8	36	112.9	111.7	79
10	266	0.3	142.9	247.2	37	125.8	236.7	36	116.2	109.5	77
Avg.	251.0	0.3	152.0	238.0	37	138.9	236.1	36	137.0	99.6	77
60x5x5											
1	387.8	0.4	166.7	383.9	58	164.0	392	55	169.6	145.4	112
2	419.6	0.4	214.1	364.3	58	172.7	387.3	55	189.9	133.6	110
3	387.7	0.4	191.9	372	58	180.1	361.2	55	172.0	132.8	112
4	378.7	0.4	154.7	351.1	57	136.1	365.6	55	145.9	148.7	112
5	385.8	0.4	167.9	378.2	58	162.6	364	57	152.8	144	111
6	387.2	0.4	197.4	352.3	58	170.6	324.4	55	149.2	130.2	112
7	326.7	0.4	135.9	333.5	58	140.8	351.2	55	153.6	138.5	112
8	417.9	0.4	193.3	386.3	58	171.1	380.9	55	167.3	142.5	112
9	385.3	0.4	163.2	387.5	58	164.7	384.2	55	162.4	146.4	113
10	362.2	0.4	184.5	362.7	58	184.9	354.9	55	178.8	127.3	112
Avg.	383.9	0.4	176.3	367.2	58	164.3	366.6	55	163.8	138.9	112

TABLE 6. Average Z values and their T-test results obtained from the four approaches under different problem sizes.

Problem size	FA		GA		PSO		MDFA
	Avg. Z	*t*-test	Avg. Z	*t*-test	Avg. Z	*t*-test	Avg. Z
10x5x5	57.2	Y+	48.6	Y+	50.2	10	39.7
20x5x5	113.9	Y+	108.1	Y+	107.1	20	56.3
20x5x5	251.0	Y+	238.0	Y+	236.1	36	99.6
60x5x5	383.9	Y+	367.2	Y+	366.6	55	138.9

E. COMPARISION TO A LOWER BOUND

In this section, the MDFA is compared to a lower bound based on the MILP developed in section III. *B*.

To find a lower bound, the constraints (10) and (11) in the original MILP are removed. As a result, the MILP is downgraded to the traveling salesman problems (TSP) which aims to find the minimum path for a salesman to travel all cities once. A city's location now corresponds to a component's position on a PCB. To solve the downgraded MILP model, the open source software GLPK (https://www.gnu.org/software/glpk/) is used. The GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library which can solve a problem to find the optimal solution Z∗.

To compare to the Z∗ found by the MILP, the MDFA is set with these parameter values $T0 = T1 = T2 = T4 = T5 = t_f = 0$, and $v = 5$ (*cm/*s). These settings make the MDFA to focus on finding the minimum path for assembly heads to assemble all components on the PCB. As a result the minimum assembly time (Z or *T3*) can be derived and compared to the Z∗.

Table 7 shows the results obtained from the MDFA and downgraded MILP. It is found that the MDFA is capable of finding the optimal solutions at the problem 10 × 5 × 5 as it hits the optimal Z∗ for 9 experiments out of 10 with the average gape being only about 0.3%.

Takes the 1st experiment of the problem size 10 × 5 × 5 as an example, Table 8 shows that the best assembly sequence of components is 10-2-6-1-9-3-7-8-4-5 which takes 39.7 (s) to complete assembling these components. The MILP obtains the following result:

```
INTEGER OPTIMAL SOLUTION FOUND
Time used: 0.2 secs
Memory used: 0.8 Mb (843202 bytes)
Optimal tour has length 198
  From node   To node   Distance
  1           6         21.2
  2           10        13.6
  3           9         24.1
  4           8         11
  5           4         33
  6           2         12
  7           3         20.9
  8           7         26.5
  9           1         23.1
  10          5         12.8
Model has been successfully processed
```

The result shows that the 1-6-2-10-5-4-8-7-3-9 is the minimum path which has a same loop as that found by the MDFA. The path length is 198 cm which requires 39.7 (198/5) (s) of traveling time, same to that found by MDFA. However, the MILP becomes computationally intractable at the problem size 20 × 5 × 5.

TABLE 7. The comparisons between MDFA and MILP.

Problem size	MDFA			MILP		
10x5x5	Z	T	G	Z*	T	Sequence
1	39.7	27	0.0	39.7	0.2	1-6-2-10-5-4-8-7-3-9
2	37.4	26	0.0	37.4	0.2	1-2-8- 6-9-7-10-3-5-4,
3	43.7	26	3.1	42.4	0.2	1-8-4-10-2-7-5-3-9-6
4	30.4	26	0.0	30.4	0.2	1-5-6-2-8-4-7-10-9-3
5	30.6	26	0.0	30.6	0.2	1-4-7-8-5-9-2-3-6-10
6	36.9	26	0.0	36.9	0.2	1-10-2-5-3-7-4-8-9-6
7	40.6	26	0.0	40.6	0.2	1-9-4-5-10-3-7-8-6-2
8	39.8	26	0.0	39.8	0.2	1-7-5-9-10-8-4-3-6-1
9	30.1	26	0.0	30.1	0.2	1-5-9-8-7-4-6-2-3-10
10	34.7	26	0.0	34.7	0.2	1-5-6-2-7-9-8-3-10-1
Avg.	36.39	26.1	0.3	36.26	0.2	
20x5x5						
1	47.3	78	0.4	44	468	1-6-17-13-19-2-5-10-16-20-11-18-12-14-9 8-15-4-7-3
2	62.5	78	-	-	-	abort >3600s
3	57.4	41	-	-	-	Abort> 4561s
4	44.2	41	-	-	-	Abort> 14552s

TABLE 8. The best result obtained from MDFA.

k	1	2	3	4	5	6	7	8	9	10
j	10	2	6	1	9	3	7	8	4	5
t(j)	1	6	8	5	2	5	8	8	8	9
x_j	22	14	2	9	2	18	24	19	30	30
y_j	76	65	66	46	24	6	26	52	53	86
T1	0	0	0	0	0	0	0	0	0	0
T3	2.6	2.7	2.4	4.2	4.6	4.8	4.2	5.3	2.2	6.6
T_j	2.6	2.7	2.4	4.2	4.6	4.8	4.2	5.3	2.2	6.6
B_j	0	2.6	5.3	7.7	11.9	16.6	21.4	25.6	30.9	33.1
E_j	2.6	5.3	7.7	11.9	16.6	21.4	25.6	30.9	33.1	39.7

F. DISCUSSION

1) Our experiments show that the standard FA underperforms other approaches in solving the simultaneous CSP and FAP for a multi-head gantry SMT machine. In the standard FA, each pair of fireflies are compared and the brighter one attracts the other. However, such comparisons may lead to an ineffective search as the brighter one is not certainly being an elite in the whole population. This tends to result in invalid searches. In contrast, the MDFA lets a firefly to be always attracted by an elite, either the best of a group or the whole swarm, which is expected to result in more effective searches for fireflies. The experiments showed a good result from this change.
2) The [35] highlights the need of balancing exploration and exploitation for FA due to its oscillatory behavior. This means the FA finds the best solutions faster in some cases but slower in others when dealing with a combinational optimization problem. The balancing of exploration and exploitation during firefly search is still an open issue [35]. To this issue, the MDFA uses the exploration-to-exploitation strategy. This strategy creates the maximum exploration momentum at the initial run so that the most number of areas in the solution space are searched by fireflies. This gives the best chance to find the possible optimal areas. However, at the latter search iterations, the MDFA transits more momentum to the exploitation of fewer elites which have been proven to be outstanding in the performance. The exploitation on these elites is expected to have a

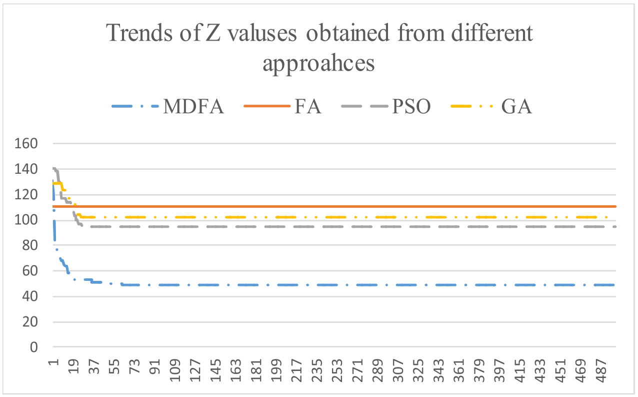


FIGURE 4. Trend of Z values of different approaches under different iterations.

high chance to find a better solution than the random search in the solution space given the few remaining iterations. At the last iteration, the maximum momentum is used to exploit fewer outstanding elites. This is a new strategy for FA.

3) One novel feature of the MDFA is that it employs discrete operators (operation) to deal with the CSP and FAP which are combinational problems with discrete domains. The discrete operators can operate the position vectors of a firefly precisely without the need of transforming values within continue domain. The experimental results show that these discrete operators used in this research can handle the two problems well and result in a better solution.
4) The [35] highlights that multi-swarm population scheme may diversify fireflies so as to better respond to changing environment. In this research, the multi-swarm scheme is adopted and the experiments show a good result.
5) In [42] the PSO was found better than the FA when used to extract features from fingerprint. In this research, the experimental results also show the PSO is better than FA in dealing with the CSP and FAP simultaneously for a multi-head gantry machine. However, the PSO is inferior to the MDFA which is an improved version of standard FA.
6) Fig. 4 shows an example of the trends of Z values found by different approaches under different iterations at the problem size $20 \times 5 \times 5$. The FA has a fixed value (the minimum of all pair comparisons) due to the only one iteration. The MDFA, GA and PSO were set to run 500 iterations. It is found that these approaches improve he best Z along with the increase of iterations. The MDFA finds the best solution at the 66th iteration ($Z = 48.70056301$). The GA finds the best solution at the 33th iteration ($Z = 101.9563891$). The PSO finds the best solution at the 28th iteration ($Z = 95.1373374$). This figure shows the MDFA has a better capability to improve the solution. In this case, the PSO performs the 2nd best, followed by the GA. The FA, though finds the best solution initially, it lacks the capability to improve the solution.

VI. CONCLUSION

The SMT machines are prevail in industry and they can affect the productivity of PCBA considerably. The SI and evolutionary-based metaheuristics is a branch of AI which can help the achievement of ''intelligent manufacturing'' in the context of Industry 4.0.

Though many metaheuristics are available for dealing with PCBA problems, to our best knowledge, the FA has never been used to deal with the CSP and FAP simultaneously for a multi-head gantry SMT machine. The main contribution of this research is the development of an improved FA, i.e., the MDFA, for dealing with the two problems for a multi-head gantry SMT machine. In addition to using the exploration-to-exploitation strategy, the MDFA is empowered by some novel features, such as multi-swarm scheme, adaptive moving step, and discrete operators. To investigate its effectiveness, the MDFA has been compared with the standard FA, GA and PSO by using different problem sizes, including $10 \times 5 \times 5$, $20 \times 5 \times 5$, $40 \times 5 \times 5$ and $60 \times 5 \times 5$. The experimental results show that the MDFA outperforms the others significantly in terms of total assembly time. The MDFA has been further compared to a downgraded mathematical model (MILP). The experimental results show that the MDFA is capable of finding an approximate solution.

The future research directions are suggested as follows. In this research, the MDFA focuses on dealing with the CSP and FAP for the multi-head gantry SMT machine. In future research, the extension of the research scope to include the NAP can be considered. In addition, the MDFA can be applied to deal with other kinds of SMT machines, such as the chip shooter machine. Finally, further improvements on the MDFA can be conducted to further strengthen its searching capability.

REFERENCES

[1] H.-P. Hsu, ''Solving feeder assignment and component sequencing problems for printed circuit board assembly using particle swarm optimization,'' *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 881–893, Apr. 2017.
[2] M. Ayob and G. Kendall, ''A survey of surface mount device placement machine optimisation: Machine classification,'' *Eur. J. Oper. Res.*, vol. 186, no. 3, pp. 893–914, May 2008.
[3] A. F. Alkaya and E. Duman, ''Combining and solving sequence dependent traveling salesman and quadratic assignment problems in PCB assembly,'' *Discrete Appl. Math.*, vol. 192, pp. 2–16, Sep. 2015.
[4] M. K. Ng, ''Heuristics approach to printed circuit board insertion problem,'' *J. Oper. Res. Soc.*, vol. 49, no. 10, pp. 1051–1059, Sep. 1998.
[5] E. K. Burke, P. I. Cowling, and R. Keuthen, ''Effective heuristic and Metaheuristic approaches to optimize component placement in printed circuit board assembly,'' in *Proc. Congr. Evol. Comput. (CEC)*, 2000, pp. 301–308.
[6] P. Ji, M. T. Sze, and W. B. Lee, ''A genetic algorithm of determining cycle time for printed circuit board assembly lines,'' *Eur. J. Oper. Res.*, vol. 128, no. 1, pp. 175–184, Jan. 2001.
[7] J. Ashayeri and W. Selen, ''A planning and scheduling model for insertion in printed circuit board assembly,'' *Eur. J. Oper. Res.*, vol. 183, no. 2, pp. 909–925, 2007.
[8] J. Luo, X. Zhang, and H. Liu, ''Modelling the operation optimization of multi-head surface mounting machines with over-head gantry in printed circuit board assembly,'' in *Proc. 33rd Chin. Control Conf.*, Jul. 2014, pp. 7537–7542.

[9] K. M. Nelson and L. T. Wille, "Comparative study of heuristics for optimal printed circuit board assembly," in *Proc. Southcon*, 1995, pp. 322–327, doi: 10.1109/SOUTHC.1995.516124.
[10] R. Balamurugan, A. M. Natarajan, and K. Premalatha, "Stellar-mass black hole optimization for biclustering microarray gene expression data," *Appl. Artif. Intell.*, vol. 29, no. 4, pp. 353–381, Apr. 2015.
[11] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1–17, Apr. 2017.
[12] M. Alssager and Z. A. Othma, "Simulated annealing algorithm using iterative component scheduling approach for chip shooter machines," *J. Theor. Appl. Inf. Technol.*, vol. 65, no. 2, pp. 480–490, Jul. 2014.
[13] W. Ho and P. Ji, "Component scheduling for chip shooter machines: A hybrid genetic algorithm approach," *Comput. Oper. Res.*, vol. 30, no. 14, pp. 2175–2189, Dec. 2003.
[14] Z. Guohui, L. Zongbin, and D. Xuan, "A hybrid genetic algorithm to optimize the printed circuit board assembly process," in *Proc. Int. Conf. Logistics Syst. Intell. Manage. (ICLSIM)*, Jan. 2010, pp. 563–567.
[15] W. Ho and P. Ji, "Integrated component scheduling models for chip shooter machines," *Int. J. Prod. Econ.*, vol. 123, no. 1, pp. 31–41, Jan. 2010.
[16] C.-J. Lin and M.-L. Huang, "Modified artificial bee colony algorithm for scheduling optimization for printed circuit board production," *J. Manuf. Syst.*, vol. 44, pp. 1–11, Jul. 2017.
[17] C.-J. Lin and M.-L. Huang, "Modified artificial bee colony algorithm for scheduling optimization for printed circuit board production," *J. Manuf. Syst.*, vol. 44, pp. 1–11, Jul. 2017.
[18] G.-Y. Zhu and W.-B. Zhang, "An improved shuffled frog-leaping algorithm to optimize component pick-and-place sequencing optimization problem," *Expert Syst. Appl.*, vol. 41, no. 15, pp. 6818–6829, Nov. 2014.
[19] Y.-M. Chen and C.-T. Lin, "A particle swarm optimization approach to optimize component placement in printed circuit board assembly," *Int. J. Adv. Manuf. Technol.*, vol. 35, nos. 5–6, pp. 610–620, Nov. 2007.
[20] H.-P. Hsu and S.-W. Yang, "Optimization of component sequencing and feeder assignment for a chip shooter machine using shuffled frog-leaping algorithm," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 56–71, Jan. 2020, doi: 10.1109/TASE.2019.2916925.
[21] X. S. Yang, "Firefly algorithm," in *Nature-inspired Meta Heuristic Algorithms*, X.-S. Yang, Ed. Hoboken, NJ, USA: Wiley, 2008, pp. 79–90.
[22] N. Poursalehi, A. Zolfaghari, and A. Minuchehr, "Multi-objective loading pattern enhancement of PWR based on the discrete firefly algorithm," *Ann. Nucl. Energy*, vol. 57, pp. 151–163, Jul. 2013.
[23] A. Rahmani and S. A. MirHassani, "A hybrid firefly-genetic algorithm for the capacitated facility location problem," *Inf. Sci.*, vol. 283, pp. 70–78, Nov. 2014.
[24] A. Baykasoğlu and F. B. Ozsoydan, "An improved firefly algorithm for solving dynamic multidimensional knapsack problems," *Expert Syst. Appl.*, vol. 41, no. 8, pp. 3712–3725, Jun. 2014.
[25] K. C. Udaiyakumar and M. Chandrasekaran, "Application of firefly algorithm in job shop scheduling problem for minimization of makespan," *Procedia Eng.*, vol. 97, pp. 1798–1807, Jan. 2014.
[26] A. Baykasoğlu and F. B. Ozsoydan, "Adaptive firefly algorithm with chaos for mechanical design optimization problems," *Appl. Soft Comput.*, vol. 36, pp. 152–164, Nov. 2015.
[27] H. Wang, W. Wang, L. Cui, H. Sun, J. Zhao, Y. Wang, and Y. Xue, "A hybrid multi-objective firefly algorithm for big data optimization," *Appl. Soft Comput.*, vol. 69, pp. 806–815, Aug. 2018.
[28] T. Kassandra, Rojali, and D. Suhartono, "Resource-constrained project scheduling problem using firefly algorithm," *Procedia Comput. Sci.*, vol. 135, pp. 534–543, Jan. 2018.
[29] S. Khalifehzadeh and M. B. Fakhrzad, "A modified firefly algorithm for optimizing a multi stage supply chain network with stochastic demand and fuzzy production capacity," *Comput. Ind. Eng.*, vol. 133, pp. 42–56, Jul. 2019.
[30] A. M. Altabeeb, A. M. Mohsen, and A. Ghallab, "An improved hybrid firefly algorithm for capacitated vehicle routing problem," *Appl. Soft Comput.*, vol. 84, Nov. 2019, Art. no. 105728.
[31] A. F. Zubair and M. S. Abu Mansor, "Embedding firefly algorithm in optimization of CAPP turning machining parameters for cutting tool selections," *Comput. Ind. Eng.*, vol. 135, pp. 317–325, Sep. 2019.
[32] S. R. Das, D. Mishra, and M. Rout, "Stock market prediction using firefly algorithm with evolutionary framework optimized feature reduction for OSELM method," *Expert Syst. Appl., X*, vol. 4, Nov. 2019, Art. no. 100016.
[33] Y. Zhang, X.-F. Song, and D.-W. Gong, "A return-cost-based binary firefly algorithm for feature selection," *Inf. Sci.*, vols. 418–419, pp. 561–574, Dec. 2017.
[34] H. Xu, S. Yu, J. Chen, and X. Zuo, "An improved firefly algorithm for feature selection in classification," *Wireless Pers. Commun.*, vol. 102, no. 4, pp. 2823–2834, Oct. 2018, doi: 10.1007/s11277-018-5309-1.
[35] I. Fister, I. Fister, Jr., X. S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm Evol. Comput.*, vol. 13, pp. 34–46, Dec. 2013.
[36] R. Kumar and H. Li, "Integer programming approach to printed circuit board assembly time optimization," *IEEE Trans. Compon., Packag., Manuf. Technol., B*, vol. 18, no. 4, pp. 720–727, Nov. 1995.
[37] M. C. Leu, H. Wong, and Z. Ji, "Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm," *J. Electron. Packag.*, vol. 115, no. 4, pp. 424–432, Dec. 1993.
[38] S. Chen and Y. Shen, "An integrated mathematical model for optimizing the component placement process with a multi-head placement," in *Proc. 29th Chin. Control Conf.*, Beijing, China, Jul. 2010, pp. 1839–1842.
[39] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications* (Lecture Notes in Computer Science), vol. 5792, O. Watanabe and T. Zeugmann, Eds. Berlin, Germany: Springer, 2009, doi: 10.1007/978-3-642-04944-6_14.
[40] Y. Zhang, D.-W. Gong, and J. Cheng, "Multi-objective particle swarm optimization approach for cost-based feature selection in classification," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 1, pp. 64–75, Jan. 2017.
[41] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective particle swarm optimization for feature selection with fuzzy cost," *IEEE Trans. Cybern.*, early access, Sep. 14, 2020, doi: 10.1109/TCYB.2020.3015756.
[42] Z. T. M. Al-Ta'i and O. Y. A. Al-Hameed, "Comparison between PSO and firefly algorithms in fingerprint authentication," *Int. J. Eng. Innov. Technol.*, vol. 3, no. 1, pp. 421–425, Jul. 2013.

HSIEN-PIN HSU received the Ph.D. degree from the Department of Industrial Engineering and Management, National Chiao-Tung University, Hsinchu, Taiwan, in 2004. He is currently a full-time Professor with the Department of Supply Chain Management, National Kaohsiung University of Science and Technology, Taiwan. He is familiar with Manufacturing Execution Systems (MESs), such as WORKSTREAM and PROMIS. He had ever served as a MIS manager in some semiconductor assembly and testing firms. He has published many articles in various international journals, including the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, the IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING, *IJPR*, *SAEC*, *ESWA*, and *IJAMT*. His research interests include MES, simulation, system modeling using Petri net or Predicate/Transition net, soft computing, and operations research in the areas of factory and container terminal.