Received May 22, 2019, accepted June 10, 2019, date of publication June 26, 2019, date of current version August 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2925088

PMT: Opposition-Based Learning Technique for Enhancing Meta-Heuristic Performance

HAMMOUDEH S. ALAMRI AND KAMAL Z. ZAMLI, (Member, IEEE)
Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Gambang 26300, Malaysia

Corresponding author: Kamal Z. Zamli (kamalz@ump.edu.my)

This work was supported in part by the Trans-Disciplinary Research Grant Scheme–An Artificial Neural Network-Sine Cosine Algorithm-Based Hybrid Prediction Model for the Production of Cellulose Nanocrystals from Oil Palm Empty Fruit Bunch under Grant RDU1918014, and in part by the Universiti Malaysia Pahang (UMP) Internal Grant–Optimization using Artificial Bee Colony Algorithm under Grant RDU182301.

ABSTRACT Meta-heuristic algorithms have shown promising performance in solving sophisticated real-world optimization problems. Nevertheless, many meta-heuristic algorithms are still suffering from a low convergence rate because of the poor balance between exploration (i.e., roaming new potential search areas) and exploitation (i.e., exploiting the existing neighbors). In some complex problems, the convergence rate can still be poor owing to becoming trapped in local optima. Addressing these issues, this research proposes a new general opposition-based learning (OBL) technique inspired by a natural phenomenon of parallel mirrors systems called the parallel mirrors technique (PMT). Like existing OBL-based approaches, the PMT generates new potential solutions based on the currently selected candidate. Unlike existing OBL-based techniques, the PMT generates more than one candidate in multiple solution-space directions. To evaluate the PMT's performance and adaptability, the PMT has been applied to four contemporary meta-heuristic algorithms, differential evolution (DE), particle swarm optimization (PSO), simulated annealing (SA), and whale optimization algorithm (WOA), to solve 15 well-known benchmark functions. The experimentally, the PMT shows promising results by accelerating the convergence rate against the original algorithms with the same number of fitness evaluations.

INDEX TERMS Optimisation, meta-heuristic, algorithms, opposition-based learning, OBL.

I. INTRODUCTION

Optimisation relates to the process of finding the best solution from all possible solutions. Specifically, optimisation involves maximising/minimising one or more defined objective functions by systematically choosing input values within allowable set values. Owing to their effectiveness in obtaining optimal solutions within reasonable computing power, meta-heuristic algorithms have often been adopted as a suitable methodology for addressing optimisation problems. To date, many meta-heuristic algorithms have been developed in the literature mimicking nature and physical phenomena such as Differential Evolution (DE) [1], Particle Swarm Optimisation (PSO) [2], Whale Optimisation Algorithm (WOA) [3], Genetic Algorithm (GA) [4], Simulated Annealing (SA)[5], Gravitational Search Algorithm (GSA) [6], and Harmony Search (HS) [7] to name a few.

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

To ensure good convergence, meta-heuristic algorithms often provide some form of parameter controls to balance between exploration (i.e., explore a new possible solution area within the search space boundaries) and exploitation (i.e., learn from the surrounding knowledge of the search neighbourhood). However, using parameter controls for balancing exploration and exploitation can still be problematic. In fact, there could be a situation when the current chosen candidate is converging to the wrong and opposite value or the best solution just a step away from the current candidate. Dealing with these situations, an opposition-based learning (OBL) scheme [8] has been proposed to consider both the candidate and its opposition, leading towards better solutions and accelerated convergence. As a result, OBL-based techniques have been adopted in many meta-heuristic algorithms (e.g., DE, GA, PSO, and HS [9]–[12]).

Although showing promising results in meta-heuristic algorithms (e.g., in terms of accelerating the convergence rate [13]–[18]), existing work on OBL has not sufficiently dealt with opposite searching from more than one direction.

Despite many useful works on OBL, the selection of the suitable opposite candidates remains a challenging problem. In particular, most existing work limits the oppositional value to a single directional search. In this research, a new technique has been proposed to exploit multi-directional searching allowing the adoption of more than one candidate from all profitable directions. Allowing such diverse search strategies can potentially be the key to obtain an optimal solution and improve the convergence rate further.

Owing to its prospects, a new OBL-based technique called the Parallel Mirrors Technique (PMT) is proposed. Inspired by a natural phenomenon when two mirrors are facing each other, the PMT can generate an infinite number of images for any object between them (i.e., allowing a diverse pool of candidate selections). As such, our contributions in this research are summarised as follows:

- A novel OBL technique based on the parallel mirrors analogy called the PMT that potentially increases the diversity of candidate solutions for consideration via multi-directional searching.
- The PMT is adaptable for general meta-heuristic algorithms. Experiments on DE, PSO, SA, and WOA, demonstrate an improved convergence rate against the original algorithms.

This paper is organised as follows: Section II illustrates the theoretical OBL framework along with related work. The proposed technique and algorithms are explained in section III. Section IV elaborates on validation experiments. The research conclusion and future work are summarised in section V.

II. THEORETICAL FRAMEWORK AND RELATED WORK

The basic concept of OBL was first proposed by Tizhoosh [8] in 2005. In short, OBL considers candidates from the opposite direction of the current solution, within the same solution space. OBL stipulates that the opposite value of the current candidate might also be closer to the vicinity of the optimal solution as well [9], hence, potentially enhancing the search convergence.

Consider a candidate solution of x, on the interval [a, b] where x is a real number ($x \in \mathbb{R}$). The opposite value x' is defined as:

$$x' = a + bx \qquad \text{(EQ1)}$$

Exploiting the basic oppositional EQ1, many existing works on exploring OBL exist. For example, Jabeen *et al.* [10] have employed the basic concept of OBL for the initialisation of population in standard PSO. The developed opposition-based PSO (O-PSO) outperforms standard PSO based on the results of selected benchmark functions. Similarly, Rahnamayan *et al.* [9] reported an enhanced convergence rate for oppositional differential evolution (ODE). In the work, ODE outperforms related work, the OBL-based HS has also been employed to solve the power compensation of an autonomous power system problem with the commendable performance [11]. Recently, Alamri *et al.* [19], [20] adopted the OBL approach for the Whale Optimisation Algorithm (OWOA) resulting in an improved convergence rate. Experimentally, the OWOA outperforms the original WOA in many benchmark functions.

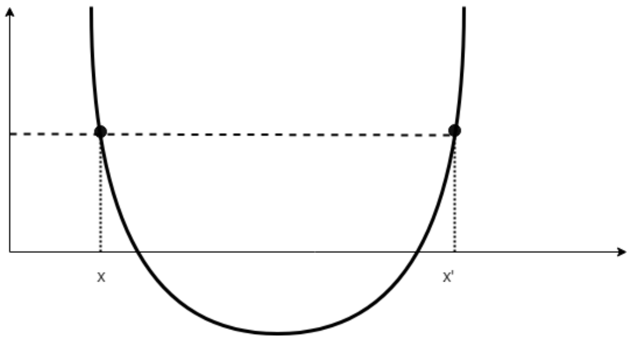
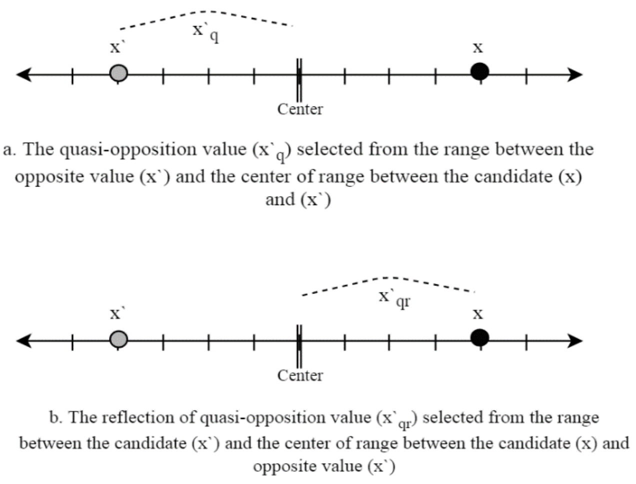


FIGURE 1. The problem of symmetrical opposition value whereby the candidate value x and its opposite value (x′) may potentially yield the same solution.



a. The quasi-opposition value ($x`_q$) selected from the range between the opposite value (x`) and the center of range between the candidate (x) and (x`)

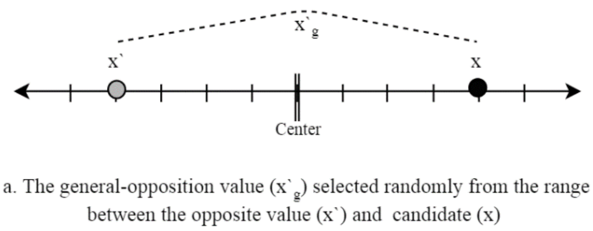
b. The reflection of quasi-opposition value ($x`_{qr}$) selected from the range between the candidate (x`) and the center of range between the candidate (x) and opposite value (x`)

FIGURE 2. Quasi-OBL and Reflection Quasi-OBL search regions in one-dimensional space.

Although giving good performance in some scenarios, the generic OBL approach is not without limitations. In cases involving symmetric problems, the use of an OBL approach may be counter-productive. Consider a minimisation problem in Figure 1. If the particular search is converging towards global optima, using OBL may generate another opposite solution but with the same fitness (see Figure 2). Here, the net effect is that the global optima will likely be missed.

Addressing the aforementioned issue, many new variants of OBL have been proposed in the literature to generate asymmetric opposition values such as Quasi-opposition (QOBL), Reflection of Quasi-opposition (RQOBL), Generalised Opposition (GOBL), Super Opposition (SOBL), and centre-based sampling (COBL).

Quasi-opposition (QOBL) dictates that the opposition solution must be close to the centre of the interval rather than the opposite value. Referring to Figure 2(a), the value of QOBL is actually selected in the space between the centre of the interval and the opposite value. Rahnamayan *et al.* [21] have demonstrated that the proposed algorithm (QODE) has outperformed the standard DE algorithm performance and the opposition-based DE (ODE) on 22 functions out of 30 benchmark functions. In similar work, Zhang *et al.* [22] employed the Quasi-OBL Particle Swarm Opposition (QOCLPSO) to improve the Oppositional Comprehensive Learning Particle Swarm Optimisation (OCLPSO) algorithm and evaluated its performance against the original algorithms (CLPSO and OCLPSO). The QOCLPSO demonstrated superior performance in term of convergence rate against CLPSO and OCLPSO. Complementing Zhang *et al.*, Sultana and Roy [23] adopted a QOBL-based algorithm called QOTLBO based on the standard teaching learning-based optimisation (TLBO). The QOTLBO has been successfully adopted to optimise the power loss within a distributed generator. The proposed algorithm has outperformed PSO, GA, and the standard TLBO for both single- and multi-objective problems. Riding on the success of the QOBL technique, Ergezer *et al.* [24] propose Quasi-Reflection opposition (QROBL). As the name suggests, QROBL introduces the opposite solution to the QOBL solution. As shown in Figure 2(b), the value of QROBL is selected in between the candidate x and the centre of the interval. Application of QROBL to Biogeography-based optimisation (BBO) [25] demonstrated good performance as compared to the standard BBO.


x`g
x`
x
Center
a. The general-opposition value (x`g) selected randomly from the range between the opposite value (x`) and candidate (x)

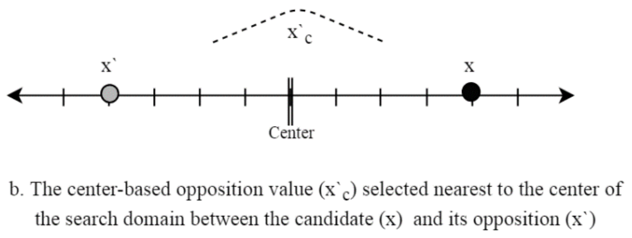

x`c
x`
x
Center
b. The center-based opposition value (x`c) selected nearest to the center of the search domain between the candidate (x) and its opposition (x`)

FIGURE 3. General OBL and centre-based OBL search regions in one-dimensional space.

Using a slightly different selection of opposition value, Rahnamayan and Wang [26] experimented with the centre between the current value and its opposite within which the opposition value (called centre-based opposition [COBL]) is selected, as shown in Figure 3(b), where the centre-based opposite candidate (x′c) is closer to the centre of the interval between the candidate (x) and its opposition (x′). Making the search space wider, Wang *et al.* [16] introduced Generalised Opposition-based Learning (GOBL) where the opposition value is selected between the current candidate and its opposite as shown in Figure 3(a). The generalised opposition candidate (x′g) is selected randomly from the interval between the candidate (x) and its opposition (x′). Applied to PSO, Wang *et al.* demonstrated the Generalised Opposition PSO (GOPSO) outperforms the standard PSO. A similar application to Artificial Bee Colony (ABC) by El-Abd [27] also yields good performance against the original ABC algorithm using 14 selected benchmark functions. Other similar works but on a different oppositional selection mechanism include Comprehensive Opposition (CO), Extended Opposition (EO), Reflected Extended Opposition [28], Super Opposition (SOBL) [14], and Fitness-based Opposition (FOBL) [24].

At a glance, the applications of the aforementioned OBL techniques appear useful in terms of improving the quality of the solution as well as facilitating good convergence. A closer look, however, reveals three main limitations. Firstly, many works on OBL are similar to each other with the exception of how the opposition value is selected. As such, the improvement of an OBL technique against its original algorithm could be highly problem dependent (i.e., based on the specific search space contour) and cannot be generalised to other optimisation problems (with different a search contour). Secondly, while existing OBL techniques have demonstrated their performance against a particular meta-heuristic algorithm, their scalability to other meta-heuristic counterparts has not been well proven. Finally, most existing OBL techniques consider a single opposite candidate selected from a particular direction. In some situations, having only two values (i.e., actual and opposite ones) might not be sufficient enough to exploit the current best-known solution(s).

Subscribing to the idea of a cluster bomb (whereby an air-dropped or ground-launched explosive weapon releases smaller sub-ammunitions to ensure total casualties and zero survivors), our work proposes an OBL generation technique based on the parallel mirrors techniques (i.e., PMT) to generate multiple (and oppositional) candidate solutions from all profitable directions. In this manner, more sufficient exploitation can be guaranteed as compared to existing works based on single oppositional OBL. It is the hypothesis that suggests generating more than one oppositional candidate solution through the parallel mirrors analogy is useful to improve the search performance of any given meta-heuristic algorithms that will be the main focus of this work.

III. PARALLEL MIRRORS TECHNIQUE (PMT)

A. PARALLEL MIRRORS SYSTEM

Historically, the first mirrors, which have been dated to around 6000 BC and found in Anatolia, Turkey, were made from well-polished stones such as obsidian stones. Originally, humans were inspired by nature to produce mirrors, since humans noticed the reflections of objects on the surface of water [29]. Since that time, mirror systems have been developed and studied to be used for different purposes and technologies as each human generation has needed, such as for making fire, for sighting aids, televisions, and solar power. The most common usage is still for grooming and tending.

In geometric optics, when two or more mirrors are placed together and the angle between them is (θ) degrees, this setup is called a multiple mirrors system. In a multiple mirrors system, the N number of images produced into the mirrors depends on the mirror type (either plane or concave), the angle between the mirrors, and the sight angle. As a special case, when the angle between two mirrors equals zero, the mirrors generate an infinite number of virtual images into each mirror. In geometric optics, this system is called a parallel mirrors system. Each generated image has the exact same shape of the original object, except for the visual size because of the virtual location from the original object. This similarity with the original object and the organised distribution of the generated image into the virtual space inspired this research to generate a different number of opposition candidates to be evaluated as new possible candidates to increase the probability of reaching a better solution.

B. PARALLEL MIRRORS TECHNIQUE (PMT)

In a parallel mirrors system, an infinite number of images is produced into each mirror. In fact, these images produce each other sequentially because of the angle between both mirrors and the sight angle. After the first image, each produced image continuously produces another image into the opposite mirror, which causes an infinite number of similar images in the virtual space. The PMT mimics this phenomenon to produce new candidates by assuming two mirrors are beside each candidate, where each mirror is located on different sides of the candidate. As such, the PMT allows distributing candidates into the searching space, which increases the probability of reaching the global optimum solution and avoiding local optimum traps.

Like OBL, the PMT tries to reproduce an opposite possible solution based on the current solution candidates. Different from the previous OBL variants, the PMT uses wider concepts to search in more than one opposite direction and produce more than one new candidate solution.

In fact, there are three types of mirrors, plane, convex, and concave, with different characteristics and behaviours. This research mimics the characteristics of plane mirrors to produce the set of virtual images from the current candidate solutions. Before going into the PMT details, some facts about the mirrors need to be highlighted to address the parallel plane mirror characteristic to produce an infinite number of images: 1) when a new image produces into a mirror, the distance between the image and the mirror equals to the distance between the real image and the mirror, 2) the angle between the object and sight equals zero, and 3) the angles between both mirrors is equal to zero as well.

Consider the parallel mirrors example in Figure 4. Here, a candidate value v is surrounded by two mirrors (M1, M2) and the distance between each mirror and v equals (d1, d2) respectively. Initially, let us assume the first mirror M1 produces the first image v1 from the original value v, at the same time M2 produces another image v2 from the original value v as well. Consequently, each image of the new images (v1, v2) will produce more new images into the opposite mirrors (v12, v21), respectively. Then each value of these new values (v12, v21) will produce an uncountable number of images into these two mirrors.

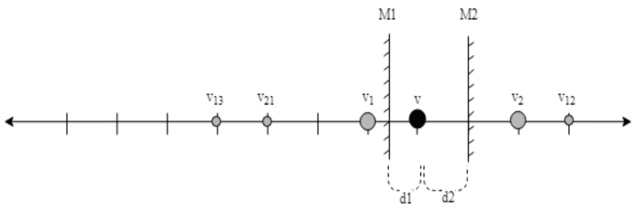


FIGURE 4. Parallel mirrors example.

Thus, to calculate the position of each generated image mathematically, the location of each mirror defined is as shown in EQ2 and EQ3.

Definition (Mirror Position): Let $v \in \mathbb{R}$ real number defined on interval $v \in [lb, ub]$. Hence, the location of the mirrors M1 and M2 are defined as follows:

$$\text{Position}\,(M1) = \text{v} - \text{d1}, \quad \text{d1} > 0 \qquad \text{(EQ2)}$$

$$\text{Position}\,(M2) = \text{v} + \text{d2}, \quad \text{d2} > 0 \qquad \text{(EQ3)}$$

where lb is the lower bound and ub is the upper bound of the solution space.

Thus, the generated images value can be defined.

Definition (PMT): Let us assume the current candidate value is v_0. Then the next generated image $v_1, \ldots v_i$ is defined as follows:

$$v_i = v_{i-1} \pm 2\,(i * d1 + (i-1)\,d2) \qquad \text{(EQ4)}$$

where $(i = 1, 2, 3 \ldots N)$ *is the image sort and* $v_i \in [lb, ub]$.

As shown in EQ4, the positions of the PMT images depend on the distance between the value v and the positions of the mirrors, which also affect the distribution of the image in the search space. To put it simply, images will be distributed in the space based on the distance between the value and each mirror, which can be managed to reproduce near or far images at the same time.

As a special case in the PMT, if d1 is equal to d2, all mirrors will generate images in the same exact positions in the search space, but this will be excluded in this research.

Generally, the PMT includes a few simple steps to generate the new candidates. Whenever a new candidate is generated (i.e., either in the initialisation step or during population), the first step is to allocate mirrors around the candidate, one mirror on the left and the second one on the right, as explained in EQ2 and EQ3. Then, the new set of generated images is defined using equation EQ4. Meanwhile, each generated image is evaluated as a new candidate and compared with the current best solution. If the current generated image reaches a better solution, the current best solution will be updated based on the generated-image value and position. To allocate the mirrors, we need to define the d1 and d2 values, which will be selected randomly in this research.

Theoretically, the PMT can be used to generate an infinite number of images, which means an infinite loop of generated images and evaluation. Thus, to make the PMT more practical for meta-heuristic algorithms, two stopping factors can be used, the maximum number of images (MI) and the maximum number of failure (MF) images. The MI is the maximum number of images that can be generated for each candidate before stopping the PMT from generating more images, and the MF is the maximum number of failure-generated images (i.e., failed to reach a better solution than the current candidate solution).

```
Input: the population $X = \{X_1, X_2 \ldots X_n\}$
Output: $X_{best}$ and the updated population $X' = \{X_1', X_2' \ldots X_n'\}$
[1].  Begin
[2].    Set $MI$ the maximum number of generated images
[3].    Set $MF$ the maximum failure images
[4].    Set initial $S_{best} = f(X)$
[5].    Set initial $X_{best} = X_0$
[6].    Set initial mirror $d_1 = random()$
[7].    Set initial mirror $d_2 = random()$
[8].    For $i = 0$; $i <$ MI; $i$ ++
[9].      $X_i = generateImage(X)$
[10].     If $f(X_i) < f(X_{best})$ then
[11].          $S_{best} = f(X_i)$
[12].          $X_{best} = X_i$
[13].     Else
[14].          Update $d_1 = random()$
[15].          Update $d_2 = random()$
[16].        $MF = MF - 1$
[17].     EndIf
[18].     If MF == 0
[19].       Break; //Stop the for loop
[20].     EndIf
[21].   EndFor
[22].     Return the best population, $X$ and the best result ($X_{best}$)
[23].   End
```

FIGURE 5. PMT pseudo code.

As shown in the PMT pseudo code in Figure 5, the PMT has the current candidate as input to generate new images as candidates. For as many meta-heuristic algorithms based on a randomly generated population, the PMT uses that population as input in each iteration in the algorithm. To initialise the PMT, four values need to be set, MI, MF, d1, and d2, as shown in Figure 5, lines 2–7. Based on the maximum number of images (MI), the image generator calls for the loop to generate and evaluate a new candidate based on the input population, as shown in Figure 5, lines 8–21.

For as many meta-heuristic algorithms based on randomly generated values, the PMT aims to use the generated values to mirror them from two sides to generate many images into the mirrors based on the distance between the mirrors and the candidate. The distance between each mirror and the candidate can be effectively used to maintain a balance between exploration and exploitation by generating images close to the current candidate, while continuously generating and evaluating faraway images.

In this research, a random-based PMT was adopted to verify the concept of the PMT. Figure 5 explains the pseudo code of the basic PMT. In line 2, the number of the targeted image (N) assigned to control the maximum number of images can be generated from the candidate. Lines 3 and 4 set and evaluate the current population. In lines 5 and 6, the distance parameters (d1 and d2) are selected randomly to start generating and evaluating the images as the lines between 7 and 14 show. Lines 11, 12, and 13 show the criteria of updating the best solution based on the cost function. If the generated image shows a better result, the global best cost needs to be updated. The current population can be updated in this step, as shown in line 13. Otherwise, the mirror location needs to be changed.

To prove that the PMT is a generic technique to enhance meta-heuristic algorithms, the basic PMT adopted was applied in four well-known algorithms (DE, PSO, SA, and WOA), which represent an evolutionary-based algorithm, a swarm-based algorithm, a single solution-based algorithm, and a bio-based algorithm, respectively. All these algorithms were extended using the PMT and tested using 15 benchmark functions to evaluate the performance in terms of convergence rate.

C. PMT-DE

Differential Evolution (DE) is a population-based evolutionary algorithm that uses a parallel direct search method [1], [30]. The DE algorithm starts the population vector using randomly generated values from the search space, and then it uses three steps to search in the search space (exploration) and enhance its search within local values (exploitation).

In the original DE (DE/rand/1/bin), we assume that $X_{n,G}$ (n = 1, 2, 3 . . . NP) are a population for each generation (G). All populations should be chosen randomly from the search space with uniform distribution. Then the initialised population is used by the next three operations, mutation, crossover, and selection, to find the best candidate solution, which is defined as follows:

1) MUTATION

For each vector $X_{n,G}$ (n = 1, 2, . . . NP), a mutant vector is generated according to EQ5:

$$v_{n,G+1} = X_{c1,G} + \mathrm{F}\left(X_{c2,G} - X_{c3,G}\right), \quad c1 \neq c2 \neq c3 \tag{EQ5}$$

where c1, c2, and c2 are random integers ∈ [1, NP] and the scaling factor F is a positive real number between 0 and 2. For each vector c1, c2, and c3 chosen randomly to be different from the running index n, the scaling factor F is a constant positive real number chosen from the interval [0,2] to control the amplification of the difference vectors F $\left(X_{c2,G} - X_{c3,G}\right)$.

2) CROSSOVER

The crossover step aims to increase the diversity of perturbed parameter vectors using the following scheme based on EQ6:

$$u_{n,G+1} = \left(u_{1n,G+1}, u_{2n,G+1} \ldots, u_{Dn,G+1}\right) \qquad \text{(EQ6)}$$

where

$$u_{jn,G+1} = \begin{cases} v_{jn,G+1}, & if\ randi(n) \le CR\ or\ j = ri(n) \\ x_{ij,G}, & if\ randi(n) > CR\ and\ j \ne ri(n) \end{cases}$$

for j = 1, 2, 3 . . . D; randi(j) ∈ [0,1] is the evaluation uniform random number; CR is the crossover constant ∈ [0,1], which is configured by the user; and ri(n) is a random number ∈ 1, 2, 3 . . . D.

3) SELECTION

In this step, a greedy selection criterion decides if the new population $u_{jn,G+1}$ yields a better result than $u_{jn,G}$ using the cost function. If the new generation gives a better result, then the $X_{jn,G+1}$ is set to $u_{jn,G+1}$. Otherwise, $X_{jn,G}$ will not change.

In the DE algorithm, some control parameters can affect the algorithm performance, according to Storn and Price [1]. The suggested values by Storn and Price [1] are:

- F ∈ [0.5,1]
- CR ∈ [0.8,1]
- NP = 10* D, where D is the dimensionality of the problem.

Although DE has been used in the last decade to solve many real-world problems [31]–[33], DE still suffers from an unstable and low convergence rate and is easily trapped into local best solutions. Hence, in this research, the PMT is employed to enhance the DE algorithm to keep a balance between exploration and exploitation to achieve a better convergence rate. The DE algorithm generates a new population in each iteration, to explore new possible solutions in the search space, by using the PMT. Each time a new population is generated, a set of images from this population will be produced using the PMT. Then, during the selection criteria, selected images will be evaluated to rival the original population to find the best solution. As shown in Figure 6, the PMT-DE algorithm initialises and starts as with the original algorithm steps, but as shown in lines 12–25, the PMT-DE algorithm generates and evaluates images based on the original population using the PMT. During the PMT, if the generated population is not showing a better result, the position of the mirror will be updated randomly, as shown in lines 16–22. On the other hand, if the PMT keeps failing to generate a better solution than the original solution, the PMT will stop generating more images and move forward to the next step of the DE algorithm, as shown in the lines 23–25. In other words, using the

```
Input: the population X = {X1, X2... Xn}
Output: Xbest and the updated population X ′= {X1′, X2′... Xn′}
[1].  Begin
[2].    Initialize random populations X
[3].    Set MI the maximum number of generated images
[4].    Set MF the maximum failure images
[5].    Set initial $S_{best} = f(X)$
[6].    Set initial $X_{best} = X_0$
[7].    While (DE Stop Criteria)
[8].    Mutation Step
[9].    Crossover Step
[10].   Selection Step
[11].   Set initial mirror d_1= random ()
[12].   Set initial mirror d_2= random ()
[13].   For i = 0; i < N; i ++
[14].     $X_i = generateImage(X)$
[15].     If $f(X_i) < f(X_{best})$ then
[16].          $S_{best} = f(X_i)$
[17].          $X_{best} = X_i$
[18].     Else
[19].          Update $d_1 = random()$
[20].          Update $d_2 = random()$
[21].          $MF = MF - 1$
[22].     EndIf
[23].     If MF == 0
[24].       Break; //Stop the for loop
[25].     EndIf
[26].   EndFor
[27].   Return the best population, X and the best result (Xbest)
[28].   EndWhile
[29].  End
```

FIGURE 6. PMT-DE pseudo code.

PMT to enhance the DE algorithm will help the DE steps to explore more local candidates and explore further candidates as well, which could positively help to overcome the DE algorithm's weaknesses.

D. PMT-PSO

Particle Swarm Optimisation (PSO) has become a popular meta-heuristic algorithm owing to its performance in solving many optimisation problems. The PSO algorithm simulates the social behaviour in the physical movements of birds and fish, where each individual particle shares its own movement experience in the space. The PSO algorithm's simple searching steps are as follows:

- Each particle (fish, bird) is treated as a volume-less point in D-dimensional space
- The ith particle is presented as follows: $X_i = (x_{i1}, x_{i2}, x_{i3} \ldots x_{iD})$
- The best particle position is represented as $P_i = (p_{i1}, p_{i2}, p_{i3} \ldots p_{iD})$
- The index of the best particle is represented by b

- The velocity (rate of change) of each particle is represented as $V_i = (v_{i1}, v_{i2}, v_{i3}, \ldots v_{iD})$
- All particles are manipulated using (EQ7) and (EQ8):

$$v_{id} = v_{id} + c_1 * rand\,1() * (p_{id} - x_{id}) + c_2 * rand2() * (p_{bd} - x_{id}), \quad \text{(EQ7)}$$

where rand1() and rand2() are two random functions between [0,1], and c1, c2 are two positive constants

○ $$x_{id} = x_{id} + v_{id}, \quad \text{(EQ8)}$$

- Finally, the particle is evaluated to find the best value

As with many meta-heuristic algorithms, PSO suffers from a low convergence rate as well as a weakness of a lack of dynamic velocity adjustment. Hence, employing the PMT to enhance the PSO should give an opportunity for the PSO to overcome some of its shortcomings. To put it simply, the PMT can be employed to enhance the PSO by mirroring each particle to generate new virtual particles and assume the new images have a better opportunity to be closer to the best solution. If the generated particles reached better results, it will be treated as a new best particle. As shown in Figure 7, the PMT-PSO pseudo code modifies the original particle by generating a new set of virtual images to be evaluated, as shown in lines 12–20. The PMT-PSO starts by initialising the PMT and the PSO parameters as well as the initial population, as shown in lines 2–6. Then, using the original PSO steps as shown in lines 7–9, the PMT is injected after calculating the latest best solution. Then for each generated population, the PMT is applied to generate parallel images and evaluate them using the targeted-cost function, as shown in lines 10–124. In lines 14–20, the PMT updates the best population and best solution values if the new generated images reach a better solution than the latest updated solution. But if it does not enhance the result, mirrors are relocated randomly within the space and continue generating more new images. Finally, the PMT stops generating images if there has been either a failure to generate a better solution than the MF or the limit of the allowed generated images has been exceeded.

```
Input: the population $X = \{X_1, X_2 \ldots X_n\}$
Output: $X_{best}$ and the updated population $X' = \{X_1', X_2' \ldots X_n'\}$
[1].  Begin
[2].    Initialize random populations $X$
[3].    Set $MI$ the maximum number of generated images
[4].    Set $MF$ the maximum failure images
[5].    Set initial $S_{best} = f(X)$
[6].    Set initial $X_{best} = X_0$
[7].    While (PSO Stop Criteria)
[8].    GenerateParticalesPositions(X)
[9].    CalculateBestParticals
[10].   Set initial mirror d_1= random ()
[11].   Set initial mirror d_2= random ()
[12].   For $i = 0$; $i < N$; $i$++
[13].     $X_i = generateImage(X)$
[14].     If $f(X_i) < f(X_{best})$ then
[15].         $S_{best} = f(X_i)$
[16].         $X_{best} = X_i$
[17].     Else
[18].         Update $d_1 = random()$
[19].         Update $d_2 = random()$
[20].         $MF = MF - 1$
[21].     EndIf
[22].     If MF == 0
[23].       Break; //Stop the for loop
[24].     EndIf
[25].   EndFor
[26].   Return the best population, $X$ and the best result ($X_{best}$)
[27].   EndWhile
[28].   End
```

FIGURE 7. PMT-PSO pseudo code.

E. PMT-SA

The Simulated Annealing (SA) algorithm is a random-based algorithm that uses a single candidate to find the optimum value [5]. The SA algorithm was inspired by the annealing process in metalwork, where solid metal is heated until it becomes liquid and then cooled slowly to reach thermal equilibrium. The slow cooling process increases the size of the substance's metal crystals and reduces defects. Based on the physical analogy, the SA algorithm involves a few steps to search for the optimum solutions:

- Initialisation: The SA algorithm starts by initialising the temperature and selecting a random initial solution.
- Move: A random neighbour is selected.
- Evaluate: The fitness function is calculated.
- Choose: Depending on the evaluation, the current solution can be updated or the previous solution kept.
- Update and loop: The temperature is updated and the steps (move, evaluate, and choose) are repeated again until reaching the freezing point.

Although SA is an easily adopted algorithm because of the simple steps, its ability to find the global optima is weak [34]. Hence, using opposition-based SA shows a noticeable improvement in the convergence rate [35]. Thus, exploiting the PMT concept can lead the SA algorithm to new possible solutions and avoid being trapped in the local optima.

Regarding the concept of the PMT cloning the original solution candidate, the PMT-SA pseudo code in Figure 8 shows the modified SA algorithm employing the concept of the PMT to generate a new candidate each time the SA searches for a new random solution.

Figure 8 explains the PMT-SA pseudo code, which is formed by injecting the PMT steps into the SA algorithm steps. In the beginning, the SA and the PMT parameters need to be initialised, as shown in lines 1–10. Then, while the SA temperature changes in each iteration, the PMT keeps trying

```
Input:  $X = \{X_1, X_2 \ldots X_n\}$
Output: $X_{best}$ and the updated solution $X' = \{X_1', X_2' \ldots X_n'\}$
[1].  Begin
[2].    Initialize random solution $X$
[3].    $T_{max}$ = final temperature
[4].    T= initial temperature
[5].    Set max iteration iter
[6].    Set initial $S_{best} = f(X)$
[7].    Set initial $X_{best} = X_0$
[8].    Initialize random populations $X$
[9].    Set $MI$ the maximum number of generated images
[10].   Set $MF$ the maximum failure images
[11].   While ($T > T_{max}$)
[12].     While ($iter > 0$)
[13].   s = select new random $X$
[14].   For $i = 0$; $i < N$; $i$ ++
[15].     $X_i = generateImage(s)$
[16].     If $f(X_i) < f(X_{best})$ then
[17].          $S_{best} = f(X_i)$
[18].          $X_{best} = X_i$
[19].     Else
[20].         Update $d_1 = random()$
[21].         Update $d_2 = random()$
[22].         $MF = MF - 1$
[23].         If random (0,1) < $e^{(\frac{f(X_i)-f(s)}{T})}$ )
[24].           $X_i = s$
[25].         EndIf
[26].     EndIf
[27].     If MF == 0
[28].       Break; //Stop the for loop
[29].     EndIf
[30].   EndFor
[31].   Update temperature T
[32].   EndWhile
[33].   EndWhile
[34].   Return the best population, $X$ and the best result ($X_{best}$)
[35].   End
```

FIGURE 8. PMT-SA pseudo code.

to generate images, as explained between lines 11 and 30. In the meantime, if in any case the number of failures in images exceeds the limit of the failures (MF), the PMT relocates the mirrors, as shown in lines 19–22. But if the MF reaches its maximum, the PMT should be stopped immediately.

F. PMT-WOA

The Whale Optimisation Algorithm (WOA) [3] is a new swarming intelligence optimisation algorithm that mimics the hunting mechanism of humpback whales. The WOA was inspired by the unique shrinking encircling and spiral updating behaviours of humpback whales when they are hunting a herd of their targeted prey. Based on humpback whale behaviour, the WOA has three stages of searching in the problem space: encircling, bubble-net attacking, and searching for new prey.

In the first step, the encircling WOA assumes that the targeted prey is the best solution, or that it is close to the optimum solution. Then, based on the current other individual agents, the algorithm tries to update its positions towards the best solution. The second step is the bubble-net attaching, which represents the exploitation phase. In this step, individual whales update their positions to be closer to the current best solution. Finally, in the search for prey, or the exploration phase, the whales are forced to update their positions randomly far away from the reference whale to search for new and better prey.

```
Input:  $X = \{X_1, X_2 \ldots X_n\}$
Output: $X_{best}$ and the updated solution $X' = \{X_1', X_2' \ldots X_n'\}$
[36].  Begin
[37].   Initialize random solution $X$
[38].   Calculate the fitness of each search agent
[39].   Set max iteration iter
[40].   Set initial $S_{best} = f(X)$
[41].   Set initial $X_{best} = X_0$
[42].   Initialize random populations $X$
[43].   Set $MI$ the maximum number of generated images
[44].   Set $MF$ the maximum failure images
[45].    While ($iter > 0$)
[46].   s = select new random $X$
[47].   Update Each Search agent ()
[48].   For $i = 0$; $i < MI$; $i$ ++
[49].     $X_i = generateImage(s)$
[50].     If $f(X_i) < f(X_{best})$ then
[51].          $S_{best} = f(X_i)$
[52].          $X_{best} = X_i$
[53].     Else
[54].         Update $d_1 = random()$
[55].         Update $d_2 = random()$
[56].         $MF = MF - 1$
[57].         EndIf
[58].     EndIf
[59].     If MF == 0
[60].       Break; //Stop the for loop
[61].     EndIf
[62].   EndFor
[63].   Update WOA values (a, A and c)
[64].   EndWhile
[65].     Return the best population, $X$ and the best result ($X_{best}$)
End
```

FIGURE 9. PMT-WOA pseudo code.

Figure 9 shows the pseudo code of the WOA modified by the PMT. In the first few lines, the WOA generates a random population and evaluates it using the fitness function and initialising the PMT limitation values MI and M. Then, after the WOA generates and evaluates the population, the PMT employs the generated population to generate new a possible solution, as shown in lines 11–23. As the PMT limits the generated images by the MI and MF, the for loop in line 13, it limits the number of the maximum number of images, while

TABLE 1. All algorithms' parameters settings and function calls (FC), and PMT setup parameters.

Algorithm	Parameter	Value
DE	d-30 FC	100000
	d-60 FC	200000
	F	[0.5, 0.9]
	CR	0.9
PSO	d-30 FC	200000
	d-60 FC	500000
	w	1
	wdamp	0.99
	C1	1.5
	C2	2
SA	d-30 FC	10000
	d-60 FC	20000
	T_0	0.1
WOA	d-30 FC	50000
	d-60 FC	100000
	Search agents	30
PMT	d1	(0,1]
	d2	(0,1]
	MF	5

the code in lines 21 and 24–26 keeps the PMT tracking the image failures to stop it when it reaches the maximum number of failures. In each iteration of the algorithm, the fitness function will be evaluated by the originally generated population and evaluated again in each generated image by the PMT steps.

IV. EXPERIMENTAL RESULTS

To validate the PMT performance, the experiments in this study focus on three related goals: 1) assess PMT performance in terms of enhancing the original version of the adopted algorithms, 2) determine if the PMT is a generic technique for meta-heuristic algorithms, and 3) verify the PMT using statistical analysis. In line with these objectives, we raised three research questions:

RQ1: Can the PMT improve existing meta-heuristic algorithms in terms of speed and convergence rate?

RQ2: Is the PMT a generic technique for meta-heuristic algorithms?

RQ3: Is the PMT depends on the problem?

To answer these questions, this study applied the PMT on four meta-heuristic algorithms, each representing a different type of meta-heuristic algorithm, and tested them on 15 benchmark functions to compare the PMT performance using the modified algorithms' results with the original algorithms' results. Furthermore, four versions of the PMT were applied in all meta-heuristic algorithms and benchmarked independently.

Hence, the set of 15 well-known benchmark functions as explained in Table 2 have been used to verify the performance of all tested algorithms after they were modified by the PMT. As shown in Table 2, the benchmark functions include eight multimodal functions and seven unimodal functions.

For the PMT setup parameters, each algorithm was modified by four versions of the PMT (each version uses a different number of the maximum number of images [MI], namely, PMT4, PMT12, PMT20, and PMT28) and five maximum failure (MF) images. The original algorithms (DE, PSO, SA, and WOA) were compared with all versions of the PMT algorithms using two problem sizes, 30 and 60 dimensions.

To make a fair comparison, all the PMT versions (PMT4, PMT12, PMT20, and PMT28) used the same number of function calls (FC), (i.e., the FC is based on an algorithm and problem size, as shown in Table 1). In addition, we set the parameter to the most common parameter setting in the literature for all algorithms, as Table 1 shows. For PMT parameter MF aims to prevent wasting computing time loop searching in weak solutions. Hence, for all PMT versions MF sets to equal 5, which will be ignored for small PMT versions like PMT4 or lower, while cut-off generating weak images for the large PMT versions like PMT12, PMT 20 and PMT28. All the algorithms were implemented in MATLAB (version 2018b) and executed on a Dell machine (Intel Core i7-7700, 3.60 GHz CPU, Windows 10 Pro, 64-bit, 16 GB RAM).

To reduce the randomness effect in the experiment, each version of each algorithm ran 750 times independently (50 runs for each benchmark function). The results of running all algorithms and their PMT versions are tabulated in Tables 3–10. As this experiment aims to measure the performance of the PMT versions against the original version of the same algorithm, experiment results measure the average of the most optimum result (R.Avg) and the average execution time to reach the optimum results (T.Avg) as well as the standard deviation (Std) to point out the distribution of the results around the average, as shown in the results tables (Table 3–10).

As the results in Tables 3–10 illustrate, the PMT shows a promising result in most benchmark functions for any problem size and benchmark modal. Moreover, the PMT results show an algorithm's independency, where all the adopted algorithms have been improved by at least one of the PMT versions. The evolutionary-based algorithm (DE), shows a noticeable improvement in all benchmark functions in 30-dimensional problems and 60-dimensional problems, and with multimodal and unimodal benchmark functions. Furthermore, the execution times were reduced by increasing the PMT images as shown in Tables 3 and 7. Likewise, the PSO results prove that the weakness in the PSO algorithm was overcome, and its performance was improved in terms of the most optimum results and time. The T.Std and R.Std show more stable results in a large number of images (i.e., PMT20 and PMT28) as shown in Tables 4 and 8. The results of the SA algorithms, which used a single search agent, show that it is best to use the PMT to improve its results. Tables 5 and 9 show that most of the better results are from the PMT20 and PMT28. In the last algorithm, the WOA, which is the most recently developed algorithm in this research, the results improve in all 30-dimensional benchmark functions,

TABLE 2. Benchmark function details (Function name, functions' formula, global minimum value, searching range, and function type – unimodal or multimodal-).

Function	Equation	Min	Range	Modal
Ackley	$f_1(x_1,\dots,x_n) = -a.exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)\right) + a + exp(1)$	0	[-32, 32]	Multimodal
Schwefel	$f_2(x_1,x_2,\dots,x_n) = 418.9829d - \sum_{i=1}^{n} x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$	0	[-500,500]	Unimodal
Schwefel 2.20	$f_3(x_1,\dots,x_n) = \sum_{i=1}^{n} \lvert x_i\rvert$	0	[-100,100]	Unimodal
Powell Sum	$f_4(x_1,\dots,x_n) = \sum_{i=1}^{n} \lvert x_i\rvert^{i+1}$	0	[-1,1]	Unimodal
Rastrigin	$f_5(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n}\left(x_i^2 - 10cos(2\pi x_i)\right)$	0	[-5.12, 5.12]	Multimodal
Rosenbrock	$f_6(x_1 \cdots x_n) = \sum_{i=1}^{n-1}\left(100\left(x_i^2 - x_{i+1}\right)^2 + (1-x_i)^2\right)$	0	[-5,10]	Multimodal
Sphere	$f_7(x_1 \cdots x_n) = \sum_{i=1}^{n} x_i^2$	0	[-5.12,5.12]	Unimodal
Sum Squares	$f_8(x_1,\dots,x_n) = \sum_{i=1}^{n} ix_i^2$	0	[-10,10]	Unimodal
Zakharov	$f_9(x_1,\dots,x_n) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$	0	[-5,10]	Unimodal
Xin-She Yang N. 2	$f_{10}(x_1,\dots,x_n) = \left(\sum_{i=1}^{n}\lvert x_i\rvert\right) exp\left(-\sum_{i=1}^{n} sin(x_i^2)\right)$	0	[-2π, 2π]	Multimodal
Xin-She Yang N. 4	$f_{11}(x_1,\dots,x_n) = \left(\sum_{i=1}^{n} sin^2(x_i) - e^{-\sum_{i=1}^{n} x_i^2}\right) e^{-\sum_{i=1}^{n} sin^2\sqrt{\lvert x_i\rvert}}$	-1	[-10,10]	Multimodal
Salomon	$f_{12}(x_1,\dots,x_n) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	0	[-100,100]	Multimodal
Qing	$f_{13}(x_1,\dots,x_n) = \sum_{i=1}^{n}(x^2 - i)^2$	0	[-500,500]	Multimodal
Alpine N. 1	$f_{14}(x_1,\dots,x_n) = \sum i = 1^n \lvert x_i sin(x_i) + 0.1x_i\rvert$	0	[0,10]	Multimodal
Schwefel 2.23	$f_{15}(x_1,\dots,x_n) = \sum_{i=1}^{n} x_i^{10}$	0	[-10,10]	Unimodal

while the WOA loses some performance in the largest benchmark functions, as explained in Tables 6 and 10.

V. DISCUSSION

As experimental results show a promising performance by using the PMT, the question is raised as to how the PMT improved the meta-heuristics. The main reason for the improvements is that most meta-heuristic algorithms are based on a random population, which make the algorithms depend on the random-generation algorithm. Yet, however the population is distributed in the search space, the results could be better. Based on the PMT calculations, the generated population will be manipulated to build a new, more distributed population. Thus, the first images can be considered a local search population and the next images can be a further exploration of new farther solutions. In other words, the PMT guides the current generated population to new local and global solutions.

Our experiment tries to answer three questions, RQ1: Can the PMT improve existing meta-heuristic algorithms in terms

TABLE 3. **DE 30 dimensions, original DE comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.**

		Original	PMT4-DE	PMT12-DE	PMT20-DE	PMT28-DE
f1	R.Avg	19.894	**18.893**	19.208	19.055	19.594
	R.Std	**0.0123**	3.6711	2.489	2.992	1.6918
	T.Avg	26.5939	21.998	27.059	29.148	**14.555**
	T.Std	9.815	8.343	11.878	9.204	**5.8115**
f2	R.Avg	5829.276	4466.88	4409.78	**4097.94**	4836.50
	R.Std	**314.929**	1402.202	1286.98	1619.55	1112.59
	T.Avg	31.760	**20.707**	23.551	23.291	29.174
	T.Std	13.836	**9.078**	12.141	13.136	12.709
f3	R.Avg	75.432	52.68	53.944	**52.316**	56.402
	R.Std	13.992	8.264	**7.644**	8.183	9.441
	T.Avg	34.018	42.88	36.22	**33.61**	35.15
	T.Std	14.787	15.037	18.912	15.61	**13.466**
f4	R.Avg	6.5E-05	1.8E-06	**9.6E-07**	1.4E-06	1.5E-06
	R.Std	6.64E-05	1.76E-06	**8.5E-07**	1.4E-06	1.5E-06
	T.Avg	28.135	26.814	28.933	34.198	**19.219**
	T.Std	11.608	10.541	12.975	11.935	**7.750**
f5	R.Avg	260.238	231.68	**230.613**	232.108	232.408
	R.Std	**11.925**	14.617	14.849	14.631	12.411
	T.Avg	39.708	38.347	46.307	40.896	**38.069**
	T.Std	17.195	19.723	18.073	**10.849**	14.854
f6	R.Avg	584.900	307.779	301.813	325.790	**293.521**
	R.Std	227.379	74.888	**57.097**	92.622	72.007
	T.Avg	35.344	26.408	**22.267**	24.363	28.759
	T.Std	12.339	**9.027**	9.867	10.411	10.856
f7	R.Avg	1.386	**0.372**	0.377	0.379	0.396
	R.Std	0.302	0.110	0.109	**0.107**	0.128
	T.Avg	**27.854**	37.215	37.140	28.778	39.263
	T.Std	16.541	19.955	16.259	15.219	**13.602**
f8	R.Avg	47.933	**13.571**	14.437	16.166	14.670
	R.Std	10.883	3.396	3.722	3.899	**3.380**
	T.Avg	33.070	42.932	**26.046**	27.380	30.000
	T.Std	17.432	15.588	10.134	**9.907**	11.852
f9	R.Avg	244.799	211.460	209.408	**203.802**	213.175
	R.Std	**24.974**	33.643	28.491	26.465	25.552
	T.Avg	30.990	22.012	33.064	32.667	**29.512**
	T.Std	**12.208**	13.046	16.442	18.601	13.696
f10	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	37.982	43.441	43.874	**33.117**	43.816
	T.Std	17.481	19.404	18.149	**9.536**	20.186
f11	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	44.854	37.336	**34.326**	35.044	37.540
	T.Std	18.905	17.691	19.107	17.273	**13.107**
f12	R.Avg	4.297	3.041	3.024	2.988	**2.987**
	R.Std	0.343	**0.326**	0.382	0.372	0.356
	T.Avg	35.823	34.908	37.403	37.945	**25.338**
	T.Std	18.166	17.721	19.196	**14.137**	14.501
f13	R.Avg	108439303	11067267	**10967164**	12361967	11855859
	R.Std	47145823	**5141629**	5678475	6693138	5557817
	T.Avg	39.676	42.766	41.254	**29.972**	40.122
	T.Std	20.165	20.147	19.745	**14.716**	15.762
f14	R.Avg	6.14E-06	**0**	**0**	**0**	**0**
	R.Std	5.87E-06	**0**	**0**	**0**	**0**
	T.Avg	40.487	34.397	**22.186**	39.908	43.279
	T.Std	17.371	17.305	16.705	**15.236**	16.823
f15	R.Avg	146508.47	939.251	1025.202	**839.055**	999.35
	R.Std	208544.584	899.321	1116.360	**833.536**	1047.339
	T.Avg	44.399	**30.795**	36.886	39.609	40.062
	T.Std	17.965	**10.871**	17.347	14.341	18.647

TABLE 4. PSO 30 dimensions, original PSO comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.

	Original PSO		PMT4-PSO		PMT12- PSO		PMT20- PSO		PMT28- PSO	
f1	R.Avg	18.939	R.Avg	16.930	R.Avg	9.605	R.Avg	**9.228**	R.Avg	9.239
	R.Std	0.462	R.Std	1.276	R.Std	0.280	R.Std	**0.144**	R.Std	0.146
	T.Avg	38.668	T.Avg	35.896	T.Avg	35.553	T.Avg	35.469	T.Avg	**35.394**
	T.Std	0.721	T.Std	0.698	T.Std	**0.263**	T.Std	0.268	T.Std	0.296
f2	R.Avg	6391.68	R.Avg	**5119.97**	R.Avg	5350.85	R.Avg	5377.94	R.Avg	5135.95
	R.Std	**414.15**	R.Std	428.07	R.Std	422.15	R.Std	472.37	R.Std	514.57
	T.Avg	40.257	T.Avg	36.799	T.Avg	36.451	T.Avg	36.331	T.Avg	**36.244**
	T.Std	0.821	T.Std	0.616	T.Std	0.319	T.Std	0.311	T.Std	**0.275**
f3	R.Avg	0.0043	R.Avg	1.94E-06	R.Avg	2.58E-06	R.Avg	**1.84E-06**	R.Avg	2.06E-06
	R.Std	0.021	R.Std	3.09E-06	R.Std	3.57E-06	R.Std	**2.37E-06**	R.Std	2.82E-06
	T.Avg	43.075	T.Avg	40.127	T.Avg	39.763	T.Avg	39.699	T.Avg	**39.579**
	T.Std	0.908	T.Std	0.609	T.Std	0.305	T.Std	0.345	T.Std	**0.295**
f4	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	51.702	T.Avg	48.278	T.Avg	47.914	T.Avg	48.016	T.Avg	**44.163**
	T.Std	0.922	T.Std	0.702	T.Std	**0.343**	T.Std	0.350	T.Std	6.716
f5	R.Avg	53.389	R.Avg	37.769	R.Avg	35.162	R.Avg	**32.575**	R.Avg	35.500
	R.Std	8.477	R.Std	6.010	R.Std	5.544	R.Std	5.283	R.Std	**4.609**
	T.Avg	39.681	T.Avg	36.635	T.Avg	**36.210**	T.Avg	36.288	T.Avg	36.274
	T.Std	0.437	T.Std	0.456	T.Std	0.339	T.Std	0.332	T.Std	**0.320**
f6	R.Avg	53.265	R.Avg	21.374	R.Avg	**19.263**	R.Avg	20.390	R.Avg	21.527
	R.Std	28.530	R.Std	4.439	R.Std	5.362	R.Std	4.283	R.Std	**2.837**
	T.Avg	38.382	T.Avg	35.260	T.Avg	34.899	T.Avg	**34.888**	T.Avg	34.896
	T.Std	0.433	T.Std	0.672	T.Std	0.320	T.Std	**0.253**	T.Std	0.308
f7	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	42.192	T.Avg	39.168	T.Avg	38.543	T.Avg	38.566	T.Avg	**38.550**
	T.Std	0.826	T.Std	0.796	T.Std	0.407	T.Std	0.336	T.Std	**0.335**
f8	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	44.172	T.Avg	41.198	T.Avg	**40.584**	T.Avg	40.586	T.Avg	40.589
	T.Std	0.643	T.Std	0.649	T.Std	0.367	T.Std	**0.287**	T.Std	0.311
f9	R.Avg	0.120	R.Avg	0.063	R.Avg	0.055	R.Avg	0.054	R.Avg	**0.052**
	R.Std	0.040	R.Std	0.027	R.Std	**0.017**	R.Std	0.021	R.Std	**0.017**
	T.Avg	38.244	T.Avg	35.310	T.Avg	**34.782**	T.Avg	34.847	T.Avg	34.808
	T.Std	0.369	T.Std	0.684	T.Std	0.127	T.Std	0.140	T.Std	**0.111**
f10	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	44.241	T.Avg	40.683	T.Avg	40.264	T.Avg	40.240	T.Avg	**40.187**
	T.Std	0.915	T.Std	0.558	T.Std	**0.297**	T.Std	0.377	T.Std	0.306
f11	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	47.760	T.Avg	44.284	T.Avg	43.777	T.Avg	**43.667**	T.Avg	43.750
	T.Std	1.070	T.Std	0.670	T.Std	**0.385**	T.Std	0.490	T.Std	0.446
f12	R.Avg	0.434	R.Avg	0.376	R.Avg	0.372	R.Avg	0.378	R.Avg	**0.370**
	R.Std	0.052	R.Std	0.043	R.Std	0.044	R.Std	**0.041**	R.Std	0.046
	T.Avg	44.766	T.Avg	40.489	T.Avg	39.646	T.Avg	**39.527**	T.Avg	39.549
	T.Std	0.738	T.Std	0.683	T.Std	**0.297**	T.Std	0.310	T.Std	0.299
f13	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	38.978	T.Avg	35.268	T.Avg	34.804	T.Avg	**34.788**	T.Avg	34.843
	T.Std	0.474	T.Std	0.737	T.Std	0.437	T.Std	**0.330**	T.Std	0.528
f14	R.Avg	0.001	R.Avg	**0**	R.Avg	**0**	R.Avg	**0**	R.Avg	**0**
	R.Std	0.005	R.Std	**0**	R.Std	**0**	R.Std	**0**	R.Std	**0**
	T.Avg	40.462	T.Avg	37.821	T.Avg	**37.468**	T.Avg	37.691	T.Avg	37.538
	T.Std	0.478	T.Std	0.709	T.Std	**0.434**	T.Std	0.498	T.Std	0.492
f15	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	49.252	T.Avg	45.144	T.Avg	44.958	T.Avg	**44.849**	T.Avg	44.878
	T.Std	1.006	T.Std	0.635	T.Std	0.309	T.Std	**0.285**	T.Std	0.389

TABLE 5. SA 30 dimensions, original SA comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.

		Original SA	PMT4-SA	PMT12-SA	PMT20-SA	PMT28-SA
f1	R.Avg	21.41	20.79	20.83	20.84	**20.78**
	R.Std	**0.10**	0.15	0.19	0.17	0.20
	T.Avg	0.405	0.379	0.401	0.366	**0.325**
	T.Std	0.125	**0.040**	0.064	0.041	**0.040**
f2	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.519	0.500	0.487	0.397	**0.373**
	T.Std	0.171	0.053	0.046	0.036	**0.024**
f3	R.Avg	1636.119	1335.760	1340.556	**1290.251**	1298.609
	R.Std	88.536	111.582	98.152	108.074	**77.499**
	T.Avg	0.495	0.456	0.454	0.362	**0.346**
	T.Std	0.049	0.050	0.454	0.031	**0.025**
f4	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.442	0.441	0.460	0.418	**0.361**
	T.Std	0.159	0.050	0.074	0.044	**0.038**
f5	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.437	0.413	0.426	0.358	**0.315**
	T.Std	0.136	0.045	0.076	0.036	**0.026**
f6	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.458	0.428	0.443	0.350	**0.323**
	T.Std	0.124	0.048	0.090	0.036	**0.025**
f7	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.423	0.418	0.436	0.397	**0.349**
	T.Std	0.121	0.055	0.067	0.050	**0.029**
f8	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.446	0.438	0.458	0.409	**0.369**
	T.Std	0.117	0.047	0.069	0.050	**0.034**
f9	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.452	0.430	0.452	0.356	**0.335**
	T.Std	0.163	0.040	0.088	0.036	**0.023**
f10	R.Avg	928.058	**0.001**	**0.001**	**0.001**	0.002
	R.Std	3521.355	**0.001**	**0.001**	**0.001**	**0.001**
	T.Avg	0.583	0.536	**0.531**	0.417	0.374
	T.Std	0.064	0.057	0.065	**0.027**	0.068
f11	R.Avg	5.5E-05	0	0	0	0
	R.Std	8.12E-5	0	0	0	0
	T.Avg	0.713	0.685	0.636	0.579	**0.212**
	T.Std	0.075	0.073	0.058	0.074	**0.026**
f12	R.Avg	34.77	29.33	**28.70**	29.01	28.97
	R.Std	1.57	1.83	1.64	1.57	**1.56**
	T.Avg	0.603	0.556	0.562	0.503	**0.270**
	T.Std	0.044	0.058	0.049	**0.042**	0.071
f13	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	0.473	0.436	0.451	0.356	**0.338**
	T.Std	0.125	0.042	0.097	0.031	**0.029**
f14	R.Avg	107.413	**35.217**	38.090	37.037	37.036
	R.Std	11.972	3.666	**3.219**	3.904	3.240
	T.Avg	0.570	0.488	0.499	0.398	**0.392**
	T.Std	0.181	0.044	0.058	**0.030**	0.033
f15	R.Avg	3.56E+10	**1.46E+10**	1.56E+10	1.64E+10	1.5E+10
	R.Std	7.3E+09	**4.91E+09**	6.0E+09	5.5E+09	5.5E+09
	T.Avg	0.529	0.465	0.481	0.431	**0.422**
	T.Std	0.175	0.043	0.053	0.048	**0.039**

TABLE 6. WOA 30 dimensions, original WOA comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.

	Original WOA		PMT4-WOA		PMT12- WOA		PMT20- WOA		PMT28- WOA	
f1	R.Avg	5.16E-06	R.Avg	1.88E-06	R.Avg	**1.86E-06**	R.Avg	2.06E-06	R.Avg	1.88E-06
	R.Std	2.74E-06	R.Std	**8.24E-07**	R.Std	1.03E-06	R.Std	1.14E-06	R.Std	1.11E-06
	T.Avg	0.621	T.Avg	0.564	T.Avg	0.543	T.Avg	0.544	T.Avg	**0.539**
	T.Std	0.244	T.Std	0.045	T.Std	0.021	T.Std	**0.017**	T.Std	0.019
f2	R.Avg	673.51	R.Avg	1.88	R.Avg	1.30	R.Avg	0.98	R.Avg	**0.95**
	R.Std	1084.71	R.Std	0.85	R.Std	0.63	R.Std	2.18	R.Std	**0.61**
	T.Avg	0.818	T.Avg	0.778	T.Avg	0.761	T.Avg	0.764	T.Avg	**0.465**
	T.Std	0.066	T.Std	0.049	T.Std	**0.025**	T.Std	0.032	T.Std	0.057
f3	R.Avg	2.25E-05	R.Avg	**6.98E-06**	R.Avg	8.3E-06	R.Avg	7.98E-06	R.Avg	8.1E-06
	R.Std	1.45E-05	R.Std	3.62E-06	R.Std	4.37E-06	R.Std	4.57E-06	R.Std	**3.29E-06**
	T.Avg	0.593	T.Avg	0.573	T.Avg	0.545	T.Avg	0.541	T.Avg	**0.538**
	T.Std	0.057	T.Std	0.047	T.Std	**0.017**	T.Std	0.019	T.Std	0.030
f4	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.984	T.Avg	0.951	T.Avg	0.931	T.Avg	0.665	T.Avg	**0.478**
	T.Std	0.056	T.Std	0.047	T.Std	**0.018**	T.Std	0.080	T.Std	0.047
f5	R.Avg	0.138	R.Avg	**0**	R.Avg	**0**	R.Avg	**0**	R.Avg	**0**
	R.Std	0.97	R.Std	**0**	R.Std	**0**	R.Std	**0**	R.Std	**0**
	T.Avg	0.639	T.Avg	0.597	T.Avg	**0.584**	T.Avg	0.586	T.Avg	0.590
	T.Std	0.052	T.Std	0.027	T.Std	**0.018**	T.Std	0.020	T.Std	0.026
f6	R.Avg	26.78	R.Avg	**26.61**	R.Avg	26.62	R.Avg	26.63	R.Avg	**26.61**
	R.Std	0.15	R.Std	**0.14**	R.Std	0.16	R.Std	**0.14**	R.Std	0.17
	T.Avg	0.583	T.Avg	0.537	T.Avg	**0.510**	T.Avg	0.515	T.Avg	0.511
	T.Std	0.114	T.Std	0.044	T.Std	**0.020**	T.Std	**0.020**	T.Std	0.023
f7	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.480	T.Avg	0.444	T.Avg	0.436	T.Avg	0.435	T.Avg	**0.430**
	T.Std	0.057	T.Std	0.039	T.Std	0.017	T.Std	**0.016**	T.Std	0.018
f8	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.480	T.Avg	0.450	T.Avg	0.433	T.Avg	**0.432**	T.Avg	0.433
	T.Std	0.132	T.Std	0.037	T.Std	0.018	T.Std	0.016	T.Std	**0.015**
f9	R.Avg	579.26	R.Avg	**392.40**	R.Avg	411.74	R.Avg	410.19	R.Avg	404.51
	R.Std	65.77	R.Std	**42.08**	R.Std	44.85	R.Std	53.00	R.Std	47.27
	T.Avg	0.640	T.Avg	0.567	T.Avg	**0.539**	T.Avg	0.543	T.Avg	**0.539**
	T.Std	0.243	T.Std	0.047	T.Std	0.028	T.Std	0.028	T.Std	**0.022**
f10	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.717	T.Avg	0.693	T.Avg	0.668	T.Avg	0.665	T.Avg	**0.502**
	T.Std	0.045	T.Std	0.033	T.Std	0.020	T.Std	0.019	T.Std	**0.013**
f11	R.Avg	0	R.Avg	-0.24	R.Avg	-0.28	R.Avg	-0.18	R.Avg	**-0.42**
	R.Std	0	R.Std	0.431	R.Std	0.454	R.Std	**0.384**	R.Std	0.494
	T.Avg	1.22	T.Avg	1.05	T.Avg	1.06	T.Avg	0.60	T.Avg	**0.40**
	T.Std	0.34	T.Std	0.05	T.Std	0.05	T.Std	0.11	T.Std	**0.03**
f12	R.Avg	0.198	R.Avg	**0.078**	R.Avg	0.094	R.Avg	**0.078**	R.Avg	0.090
	R.Std	0.038	R.Std	0.042	R.Std	**0.024**	R.Std	0.041	R.Std	0.030
	T.Avg	0.641	T.Avg	0.569	T.Avg	0.552	T.Avg	**0.544**	T.Avg	0.545
	T.Std	0.269	T.Std	0.042	T.Std	0.022	T.Std	**0.018**	T.Std	**0.018**
f13	R.Avg	31.19	R.Avg	17.47	R.Avg	16.60	R.Avg	**13.99**	R.Avg	16.54
	R.Std	15.86	R.Std	4.97	R.Std	5.15	R.Std	**3.71**	R.Std	4.77
	T.Avg	0.613	T.Avg	0.553	T.Avg	**0.522**	T.Avg	0.529	T.Avg	0.527
	T.Std	0.224	T.Std	0.037	T.Std	0.019	T.Std	**0.016**	T.Std	0.027
f14	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.842	T.Avg	0.703	T.Avg	0.670	T.Avg	0.676	T.Avg	**0.669**
	T.Std	0.271	T.Std	0.040	T.Std	0.029	T.Std	**0.021**	T.Std	0.026
f15	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.059	T.Avg	0.954	T.Avg	0.934	T.Avg	0.652	T.Avg	**0.473**
	T.Std	0.340	T.Std	0.049	T.Std	**0.019**	T.Std	0.098	T.Std	0.038

TABLE 7. DE 60 dimensions, original DE comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.

	Original		PMT4-DE		PMT12-DE		PMT20-DE		PMT28-DE	
f1	R.Avg	19.931	R.Avg	19.916	R.Avg	19.919	R.Avg	19.919	R.Avg	**19.912**
	R.Std	**0.006**	R.Std	0.021	R.Std	0.014	R.Std	1.757	R.Std	**0.021**
	T.Avg	52.420	T.Avg	55.812	T.Avg	62.378	T.Avg	**25.897**	T.Avg	49.904
	T.Std	19.412	T.Std	27.835	T.Std	25.176	T.Std	19.875	T.Std	**13.851**
f2	R.Avg	16016.79	R.Avg	11207.59	R.Avg	**10639.31**	R.Avg	11203.41	R.Avg	12115.93
	R.Std	**507.40**	R.Std	3282.41	R.Std	2929.12	R.Std	2925.14	R.Std	2598.50
	T.Avg	**31.052**	T.Avg	32.731	T.Avg	33.452	T.Avg	71.631	T.Avg	47.160
	T.Std	9.339	T.Std	**8.786**	T.Std	11.204	T.Std	21.788	T.Std	16.019
f3	R.Avg	264.521	R.Avg	102.792	R.Avg	108.758	R.Avg	**102.658**	R.Avg	103.333
	R.Std	31.295	R.Std	14.472	R.Std	16.324	R.Std	16.076	R.Std	**12.786**
	T.Avg	43.04	T.Avg	76.42	T.Avg	69.53	T.Avg	**26.38**	T.Avg	34.86
	T.Std	9.75	T.Std	31.40	T.Std	25.06	T.Std	27.49	T.Std	**9.30**
f4	R.Avg	0.067	R.Avg	2.4E-07	R.Avg	**2.2E-07**	R.Avg	2.8E-07	R.Avg	3.6E-07
	R.Std	4.31E-07	R.Std	4.65E-07	R.Std	4.65E-07	R.Std	**2.74E-07**	R.Std	5.2E-07
	T.Avg	60.985	T.Avg	74.141	T.Avg	**32.672**	T.Avg	66.111	T.Avg	61.726
	T.Std	33.576	T.Std	28.388	T.Std	**10.802**	T.Std	21.384	T.Std	25.920
f5	R.Avg	584.196	R.Avg	492.167	R.Avg	495.348	R.Avg	**490.425**	R.Avg	492.996
	R.Std	**16.245**	R.Std	21.580	R.Std	22.846	R.Std	22.675	R.Std	20.252
	T.Avg	38.693	T.Avg	28.693	T.Avg	**28.348**	T.Avg	53.213	T.Avg	66.963
	T.Std	11.484	T.Std	9.311	T.Std	7.092	T.Std	**4.995**	T.Std	26.721
f6	R.Avg	3612.318	R.Avg	614.011	R.Avg	606.540	R.Avg	**601.548**	R.Avg	667.115
	R.Std	969.332	R.Std	174.458	R.Std	**94.035**	R.Std	134.212	R.Std	151.891
	T.Avg	78.108	T.Avg	48.011	T.Avg	49.555	T.Avg	**32.842**	T.Avg	33.667
	T.Std	19.052	T.Std	18.396	T.Std	17.615	T.Std	22.861	T.Std	**14.905**
f7	R.Avg	5.633	R.Avg	0.864	R.Avg	0.823	R.Avg	**0.811**	R.Avg	0.854
	R.Std	1.273	R.Std	**0.201**	R.Std	0.249	R.Std	0.245	R.Std	0.231
	T.Avg	64.108	T.Avg	**42.703**	T.Avg	58.512	T.Avg	66.322	T.Avg	50.551
	T.Std	**19.852**	T.Std	29.374	T.Std	24.019	T.Std	26.177	T.Std	25.034
f8	R.Avg	460.247	R.Avg	73.421	R.Avg	76.812	R.Avg	73.822	R.Avg	**67.223**
	R.Std	102.799	R.Std	17.900	R.Std	21.045	R.Std	18.441	R.Std	**15.081**
	T.Avg	47.034	T.Avg	**41.927**	T.Avg	51.972	T.Avg	43.358	T.Avg	59.448
	T.Std	25.203	T.Std	20.516	T.Std	27.872	T.Std	**10.718**	T.Std	19.859
f9	R.Avg	814.299	R.Avg	718.991	R.Avg	738.097	R.Avg	726.712	R.Avg	**714.173**
	R.Std	**58.396**	R.Std	72.715	R.Std	73.230	R.Std	75.649	R.Std	77.470
	T.Avg	39.763	T.Avg	69.804	T.Avg	29.275	T.Avg	27.006	T.Avg	**25.333**
	T.Std	13.508	T.Std	29.173	T.Std	8.542	T.Std	7.727	T.Std	**3.935**
f10	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	80.984	T.Avg	66.773	T.Avg	**47.947**	T.Avg	58.406	T.Avg	50.396
	T.Std	19.125	T.Std	34.714	T.Std	8.810	T.Std	**8.636**	T.Std	25.291
f11	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	48.777	T.Avg	**39.029**	T.Avg	69.231	T.Avg	76.385	T.Avg	54.453
	T.Std	**13.475**	T.Std	18.092	T.Std	30.828	T.Std	32.718	T.Std	23.192
f12	R.Avg	7.211	R.Avg	**4.161**	R.Avg	4.210	R.Avg	4.206	R.Avg	4.242
	R.Std	0.777	R.Std	0.369	R.Std	**0.335**	R.Std	0.409	R.Std	0.353
	T.Avg	53.084	T.Avg	28.789	T.Avg	**25.394**	T.Avg	42.040	T.Avg	44.213
	T.Std	18.871	T.Std	7.827	T.Std	**2.378**	T.Std	4.299	T.Std	10.294
f13	R.Avg	612613268	R.Avg	26840838	R.Avg	28305518	R.Avg	**23084822**	R.Avg	25315458
	R.Std	**10266936**	R.Std	284005339	R.Std	14139764	R.Std	14441467	R.Std	12982566
	T.Avg	33.901	T.Avg	37.162	T.Avg	**30.564**	T.Avg	67.682	T.Avg	44.362
	T.Std	13.271	T.Std	11.818	T.Std	**9.208**	T.Std	14.270	T.Std	15.188
f14	R.Avg	0.0133325	R.Avg	**0**	R.Avg	8E-08	R.Avg	**0**	R.Avg	**0**
	R.Std	0.337	R.Std	**0**	R.Std	2.5E-07	R.Std	**0**	R.Std	**0**
	T.Avg	33.895	T.Avg	32.239	T.Avg	27.120	T.Avg	**26.360**	T.Avg	32.712
	T.Std	11.954	T.Std	12.207	T.Std	5.851	T.Std	**4.390**	T.Std	9.907
f15	R.Avg	3869767	R.Avg	5656	R.Avg	4129	R.Avg	4382	R.Avg	**3141**
	R.Std	3998640	R.Std	5379	R.Std	3881	R.Std	4603	R.Std	**2432**
	T.Avg	52.19	T.Avg	52.19	T.Avg	55.92	T.Avg	35.52	T.Avg	**32.01**
	T.Std	23.44	T.Std	19.27	T.Std	31.23	T.Std	26.91	T.Std	**6.40**

TABLE 8. PSO 60 dimensions, original PSO comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.

		Original	PMT4-PSO	PMT12-PSO	PMT20-PSO	PMT28-PSO
f1	R.Avg	0.331	**0**	**0**	**0**	**0**
	R.Std	0.658	**0**	**0**	**0**	**0**
	T.Avg	44.726	40.869	40.769	**40.690**	40.844
	T.Std	0.788	0.454	**0.361**	0.861	0.411
f2	R.Avg	12988	**11272**	11281	11422	11388
	R.Std	786	837	913	**677**	801
	T.Avg	43.095	39.227	39.002	**38.961**	39.392
	T.Std	0.836	0.398	**0.385**	0.520	0.760
f3	R.Avg	3.406	0.034	0.036	0.038	**0.023**
	R.Std	2.652	0.045	0.051	0.052	**0.033**
	T.Avg	40.777	38.687	**38.449**	38.559	38.807
	T.Std	0.699	0.513	**0.439**	0.582	0.783
f4	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	56.057	52.706	52.599	53.098	**36.172**
	T.Std	0.822	0.755	**0.319**	1.131	13.038
f5	R.Avg	118.08	84.173	**78.880**	81.149	79.975
	R.Std	16.632	**9.394**	10.566	9.808	11.072
	T.Avg	44.796	41.159	**41.153**	41.182	41.262
	T.Std	0.884	0.400	**0.297**	0.415	0.626
f6	R.Avg	163.396	63.538	64.086	**60.748**	64.758
	R.Std	43.509	20.229	21.333	**17.316**	20.967
	T.Avg	38.285	35.308	**35.053**	35.101	35.279
	T.Std	0.699	**0.333**	0.309	0.345	0.525
f7	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	38.141	35.274	35.119	**34.907**	35.190
	T.Std	0.899	0.352	0.287	**0.217**	0.571
f8	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	43.530	40.446	**40.297**	40.319	40.397
	T.Std	0.534	0.407	**0.327**	0.408	0.614
f9	R.Avg	6.626	5.561	5.418	**5.415**	5.426
	R.Std	0.500	0.582	0.527	0.510	**0.482**
	T.Avg	43.200	39.634	**39.367**	39.481	39.715
	T.Std	0.712	0.399	**0.345**	0.407	0.644
f10	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	40.996	38.056	37.776	**37.653**	38.397
	T.Std	0.564	0.382	0.286	**0.253**	0.729
f11	R.Avg	0	0	0	0	0
	R.Std	0	0	0	0	0
	T.Avg	52.910	49.069	49.100	49.889	**47.673**
	T.Std	0.850	0.518	**0.517**	1.253	4.105
f12	R.Avg	1.605	**1.349**	1.355	1.367	1.351
	R.Std	0.108	0.109	0.115	0.100	**0.079**
	T.Avg	43.785	39.404	39.002	**38.869**	39.301
	T.Std	0.788	0.361	0.338	**0.297**	0.845
f13	R.Avg	1492.41	0.00018	0.0023	0.00002	**0.000001**
	R.Std	3270.9	0.001	0.006	**0**	**0**
	T.Avg	43.011	38.946	38.790	**38.789**	39.275
	T.Std	0.788	0.498	0.399	**0.382**	1.085
f14	R.Avg	0.441	**0**	**0**	**0**	**0**
	R.Std	0.414	**0**	**0**	**0**	**0**
	T.Avg	43.903	40.886	**40.420**	40.625	40.856
	T.Std	**0.461**	0.744	0.664	0.898	1.165
f15	R.Avg	1.1E-10	**0**	**0**	**0**	**0**
	R.Std	4.1E-10	**0**	**0**	**0**	**0**
	T.Avg	48.946	45.422	**45.272**	45.330	42.348
	T.Std	0.749	0.273	**0.178**	0.328	4.789

TABLE 9. SA 60 dimensions, original SA comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.

	Original SA		PMT4-SA		PMT12-SA		PMT20-SA		PMT28-SA	
f1	R.Avg	21.33	R.Avg	**20.95**	R.Avg	**20.96**	R.Avg	**20.96**	R.Avg	20.97
	R.Std	**0.06**	R.Std	0.08	R.Std	0.13	R.Std	0.10	R.Std	0.10
	T.Avg	1.029	T.Avg	0.982	T.Avg	0.933	T.Avg	0.814	T.Avg	**0.804**
	T.Std	0.096	T.Std	0.057	T.Std	**0.021**	T.Std	0.092	T.Std	0.111
f2	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.324	T.Avg	1.339	T.Avg	1.272	T.Avg	1.162	T.Avg	**1.041**
	T.Std	0.126	T.Std	0.080	T.Std	**0.027**	T.Std	0.178	T.Std	0.077
f3	R.Avg	3112.2	R.Avg	2767.3	R.Avg	**2758.8**	R.Avg	2759.1	R.Avg	2789.7
	R.Std	122.4	R.Std	149.7	R.Std	**95.6**	R.Std	119.2	R.Std	131.8
	T.Avg	0.931	T.Avg	0.984	T.Avg	0.913	T.Avg	0.821	T.Avg	**0.820**
	T.Std	**0.023**	T.Std	0.027	T.Std	0.071	T.Std	0.042	T.Std	0.060
f4	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.156	T.Avg	1.142	T.Avg	1.097	T.Avg	0.902	T.Avg	**0.895**
	T.Std	0.181	T.Std	0.082	T.Std	**0.023**	T.Std	0.031	T.Std	0.102
f5	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.798	T.Avg	0.804	T.Avg	0.792	T.Avg	0.724	T.Avg	**0.669**
	T.Std	0.021	T.Std	0.036	T.Std	**0.014**	T.Std	0.061	T.Std	0.057
f6	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.07	T.Avg	1.02	T.Avg	0.98	T.Avg	0.94	T.Avg	**0.83**
	T.Std	0.18	T.Std	0.06	T.Std	**0.03**	T.Std	0.09	T.Std	0.07
f7	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.855	T.Avg	0.831	T.Avg	0.874	T.Avg	0.728	T.Avg	**0.693**
	T.Std	0.066	T.Std	**0.032**	T.Std	0.072	T.Std	0.036	T.Std	0.060
f8	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.141	T.Avg	1.073	T.Avg	1.045	T.Avg	0.871	T.Avg	**0.839**
	T.Std	0.196	T.Std	0.065	T.Std	0.050	T.Std	0.079	T.Std	**0.033**
f9	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.808	T.Avg	0.813	T.Avg	0.800	T.Avg	0.745	T.Avg	**0.678**
	T.Std	0.023	T.Std	0.043	T.Std	**0.014**	T.Std	0.066	T.Std	0.046
f10	R.Avg	943.3	R.Avg	1.68E-06	R.Avg	1.54E-06	R.Avg	1.12E-06	R.Avg	**1.02E-06**
	R.Std	29.7	R.Std	1.93E-06	R.Std	2.02E-06	R.Std	1.45E-06	R.Std	**1.17E-06**
	T.Avg	0.990	T.Avg	0.971	T.Avg	0.949	T.Avg	0.889	T.Avg	**0.803**
	T.Std	0.061	T.Std	0.061	T.Std	0.054	T.Std	0.048	T.Std	**0.047**
f11	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.850	T.Avg	1.723	T.Avg	1.462	T.Avg	1.385	T.Avg	**1.375**
	T.Std	0.123	T.Std	0.048	T.Std	0.182	T.Std	0.087	T.Std	**0.035**
f12	R.Avg	47.082	R.Avg	42.760	R.Avg	42.257	R.Avg	42.840	R.Avg	**41.793**
	R.Std	1.439	R.Std	1.807	R.Std	1.380	R.Std	**1.335**	R.Std	1.489
	T.Avg	1.198	T.Avg	1.134	T.Avg	**0.950**	T.Avg	0.958	T.Avg	0.958
	T.Std	0.080	T.Std	0.135	T.Std	**0.059**	T.Std	0.067	T.Std	0.066
f13	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	0.908	T.Avg	0.897	T.Avg	0.767	T.Avg	**0.745**	T.Avg	0.782
	T.Std	**0.019**	T.Std	0.027	T.Std	0.044	T.Std	0.045	T.Std	0.052
f14	R.Avg	201.58	R.Avg	79.16	R.Avg	79.53	R.Avg	80.28	R.Avg	79.76
	R.Std	13.41	R.Std	4.79	R.Std	**4.21**	R.Std	4.40	R.Std	4.93
	T.Avg	0.96	T.Avg	0.86	T.Avg	0.76	T.Avg	0.75	T.Avg	**0.64**
	T.Std	0.07	T.Std	0.05	T.Std	0.05	T.Std	0.05	T.Std	**0.03**
f15	R.Avg	6.56E+10	R.Avg	3.96E+10	R.Avg	4.04E+10	R.Avg	4.07E+10	R.Avg	**3.84E+10**
	R.Std	7.79E+09	R.Std	7.09E+09	R.Std	8.69E+09	R.Std	8.6E+09	R.Std	**1.29E+10**
	T.Avg	0.825	T.Avg	0.831	T.Avg	0.772	T.Avg	0.749	T.Avg	**0.667**
	T.Std	0.054	T.Std	0.051	T.Std	0.051	T.Std	**0.046**	T.Std	0.049

TABLE 10. **WOA 60 dimensions, original WOA comparing with PMT4, PMT12, PMT20, PMT28. Results show the average of the optimum result (R.Avg), average of execution time (T.Avg) and the Standard Deviation of results and time (R.Std and T.Std), respectively.**

	Original WOA		PMT4- WOA		PMT12- WOA		PMT20- WOA		PMT28- WOA	
f1	R.Avg	1.2E-07	R.Avg	**0**	R.Avg	**0**	R.Avg	**0**	R.Avg	**0**
	R.Std	3.28E-07	R.Std	**0**	R.Std	**0**	R.Std	**0**	R.Std	**0**
	T.Avg	1.890	T.Avg	1.730	T.Avg	1.703	T.Avg	1.711	T.Avg	**1.648**
	T.Std	0.359	T.Std	0.076	T.Std	0.075	T.Std	**0.072**	T.Std	0.099
f2	R.Avg	744.5	R.Avg	1.4	R.Avg	**1.01**	R.Avg	1.14	R.Avg	1.37
	R.Std	1385.3	R.Std	1.057	R.Std	**0.671**	R.Std	0.734	R.Std	0.921
	T.Avg	1.974	T.Avg	1.848	T.Avg	1.843	T.Avg	1.829	T.Avg	**1.254**
	T.Std	0.082	T.Std	0.049	T.Std	0.060	T.Std	**0.035**	T.Std	0.130
f3	R.Avg	2.44E-06	R.Avg	7.2E-07	R.Avg	7E-07	R.Avg	7.6E-07	R.Avg	**6.8E-07**
	R.Std	1.45E-06	R.Std	**4.54E-07**	R.Std	5.05E-07	R.Std	4.72E-07	R.Std	4.66E-07
	T.Avg	1.490	T.Avg	1.408	T.Avg	1.391	T.Avg	**1.388**	T.Avg	1.389
	T.Std	0.346	T.Std	0.058	T.Std	**0.038**	T.Std	0.040	T.Std	0.040
f4	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	3.682	T.Avg	3.562	T.Avg	3.419	T.Avg	1.486	T.Avg	**1.450**
	T.Std	0.428	T.Std	0.099	T.Std	0.208	T.Std	**0.032**	T.Std	0.042
f5	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.77	T.Avg	1.66	T.Avg	1.65	T.Avg	**1.64**	T.Avg	1.65
	T.Std	0.36	T.Std	**0.06**	T.Std	**0.06**	T.Std	**0.06**	T.Std	**0.06**
f6	R.Avg	56.79	R.Avg	56.62	R.Avg	**55.43**	R.Avg	**55.43**	R.Avg	56.56
	R.Std	**0.12**	R.Std	**0.12**	R.Std	7.87	R.Std	7.89	R.Std	**0.12**
	T.Avg	1.34	T.Avg	1.27	T.Avg	**1.25**	T.Avg	**1.25**	T.Avg	1.26
	T.Std	0.10	T.Std	0.06	T.Std	**0.03**	T.Std	0.04	T.Std	0.04
f7	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.259	T.Avg	1.241	T.Avg	1.232	T.Avg	1.223	T.Avg	**1.209**
	T.Std	0.066	T.Std	0.061	T.Std	0.058	T.Std	0.038	T.Std	**0.033**
f8	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.44	T.Avg	1.36	T.Avg	1.34	T.Avg	1.35	T.Avg	**1.32**
	T.Std	0.07	T.Std	0.08	T.Std	**0.05**	T.Std	**0.05**	T.Std	**0.05**
f9	R.Avg	1181.50	R.Avg	890.99	R.Avg	899.04	R.Avg	**880.68**	R.Avg	888.35
	R.Std	99.82	R.Std	**70.08**	R.Std	98.52	R.Std	97.89	R.Std	89.28
	T.Avg	1.56	T.Avg	1.43	T.Avg	1.43	T.Avg	1.40	T.Avg	**1.39**
	T.Std	0.35	T.Std	0.10	T.Std	0.07	T.Std	**0.05**	T.Std	**0.05**
f10	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	2.00	T.Avg	1.84	T.Avg	1.84	T.Avg	1.82	T.Avg	**1.27**
	T.Std	0.39	T.Std	0.05	T.Std	0.06	T.Std	**0.04**	T.Std	0.12
f11	R.Avg	0	R.Avg	-0.42	R.Avg	**-0.52**	R.Avg	**-0.52**	R.Avg	-0.42
	R.Std	**0**	R.Std	0.499	R.Std	0.505	R.Std	0.500	R.Std	0.494
	T.Avg	3.04	T.Avg	2.72	T.Avg	2.66	T.Avg	1.76	T.Avg	**1.06**
	T.Std	0.42	T.Std	0.18	T.Std	0.18	T.Std	0.32	T.Std	**0.08**
f12	R.Avg	0.204	R.Avg	0.084	R.Avg	**0.082**	R.Avg	0.090	R.Avg	**0.082**
	R.Std	0.057	R.Std	0.037	R.Std	0.039	R.Std	**0.030**	R.Std	0.038
	T.Avg	1.74	T.Avg	1.60	T.Avg	1.59	T.Avg	1.58	T.Avg	**1.57**
	T.Std	0.34	T.Std	0.07	T.Std	0.06	T.Std	**0.05**	T.Std	**0.05**
f13	R.Avg	470.4	R.Avg	238.1	R.Avg	220.4	R.Avg	221.8	R.Avg	**215.0**
	R.Std	252.6	R.Std	**53.4**	R.Std	53.9	R.Std	64.5	R.Std	63.6
	T.Avg	**1.45**	T.Avg	1.47	T.Avg	1.46	T.Avg	1.46	T.Avg	**1.45**
	T.Std	**0.06**	T.Std	**0.06**	T.Std	**0.06**	T.Std	0.07	T.Std	**0.06**
f14	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	1.49	T.Avg	1.49	T.Avg	1.48	T.Avg	1.48	T.Avg	**1.37**
	T.Std	**0.04**	T.Std	**0.04**	T.Std	**0.04**	T.Std	**0.04**	T.Std	0.13
f15	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0	R.Avg	0
	R.Std	0	R.Std	0	R.Std	0	R.Std	0	R.Std	0
	T.Avg	3.102	T.Avg	2.974	T.Avg	2.944	T.Avg	1.456	T.Avg	**1.377**
	T.Std	0.379	T.Std	0.083	T.Std	0.041	T.Std	**0.013**	T.Std	0.024

of speed and convergence rate? RQ2: Is the PMT a generic technique for meta-heuristic algorithms?, and RQ3: Is the PMT a depends on the problem?

To answer the first two questions (RQ1 and RQ2), in reviewing the experimental results and the statistical analysis shown, the PMT demonstrates promising results for improving the different types of meta-heuristic algorithms. These results mean that the PMT, to the best of our knowledge, can be considered as a generic technique to improve meta-heuristic algorithms. Finally, based on the different sizes and modals of the benchmark functions, the third question (RQ3) can be answered by now because the PMT shows improvement in all kinds and sizes of the benchmark functions in almost all algorithms.

Although all the experiment runs called the fitness function the same number of times, the execution time result was promising. This outperformance can be explained by the simplicity of the PMT, which allows the algorithm to reach a more promising candidate in less time without the need to go through complex calculation steps or re-population.

Although the PMT promises performance, the PMT is still limited by the distance between the mirrors and the candidate values. The distance selection in this research used a random-based selection. Hence, the mirrors could be re-located many times during the algorithm. However, this limitation could be considered as a weakness, and selecting the distance could be an effective way to improve the PMT performance in the future.

In summary, our experiments show promising results can lead a meta-heuristic algorithm to better performance by increasing the convergence rate and can lead to better results in minimum cost (less execution time). Also, our experiment results predict a favourable future for using the PMT and its variants.

VI. CONCLUSION

In this paper, a novel technique was presented to overcome the low convergence rate in many meta-heuristic algorithms, which has been addressed as a common drawback in meta-heuristic algorithms. The Parallel Mirrors Technique, inspired by a unique phenomenon in a parallel mirrors system, generates a virtual image of the candidate into mirrors to test if any image could be a better solution. The PMT has successfully been used to extend DE, PSO, SA, and WOA in order to enhance their search performances. PMT-DE, PMT-PSO, PMT-SA, and PMT-WOA show extraordinary enhancement on the original algorithms in terms of performance stability and convergence rate. Based on the observations made, the PMT shows promising results that can lead many meta-heuristic optimisation algorithms to reach better solutions with low cost. However, the PMT is still in the early stages of development. The random mirror locating could be a sizeable challenge in the future and could be improved by using different locating criteria such as self-adoption locating and manging failures rate. Nonetheless, new PMT variants to extend GA and HS algorithms are currently under development to solve multi-objective problems as well as real-life problems such as Knapsack and feature selection.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, Nov. 1995, pp. 1942–1948.

[3] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.

[4] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–72, Jul. 1992.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[6] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.

[7] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001.

[8] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Modeling Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commer.*, vol. 1, Nov. 2005, pp. 695–701.

[9] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64–79, Feb. 2008.

[10] H. Jabeen, Z. Jalil, and A. R. Baig, "Opposition based initialization in particle swarm optimization (O-PSO)," in *Proc. 11th Annu. Conf. Companion Genetic Evol. Comput. Conf. (GECCO)*, 2009, pp. 2047–2052.

[11] A. Banerjee, V. Mukherjee, and S. P. Ghoshal, "An opposition-based harmony search algorithm for engineering optimization problems," *Ain Shams Eng. J.*, vol. 5, no. 1, pp. 85–101, 2014.

[12] L.-Y. Zhou, L.-X. Ding, H. Peng, and X.-L. Qiang, "Neighborhood centroid opposition-based particle swarm optimization," *Tien Tzu Hsueh Pao/Acta Electron. Sin.*, vol. 45, no. 11, pp. 2815–2824, 2017.

[13] H. Salehinejad, S. Rahnamayan, and H. R. Tizhoosh, "Type-II opposition-based differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1768–1775.

[14] H. R. Tizhoosh, M. Ventresca, and S. Rahnamayan, "Opposition-based computing," in *Oppositional Concepts in Computational Intelligence* (Studies in Computational Intelligence), vol. 155. Berlin, Germany: Springer-Verlag, 2008, pp. 11–28.

[15] X. Yang, J. Cao, K. Li, and P. Li, "Improved opposition-based biogeography optimization," in *Proc. 4th Int. Workshop Adv. Comput. Intell. (IWACI)*, Oct. 2011, pp. 642–647.

[16] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Inf. Sci.*, vol. 181, no. 20, pp. 4699–4714, Oct. 2011.

[17] M. Sevkli and H. Uysal, "A modified variable neighborhood search for minimizing the makespan on identical parallel machines," in *Proc. Int. Conf. Comput. Ind. Eng.*, Jul. 2009, pp. 108–111.

[18] I. Alharkan, K. Bamatraf, M. A. Noman, H. Kaid, E. S. A. Nasr, and A. M. El-Tamimi, "An order effect of neighborhood structures in variable neighborhood search algorithm for minimizing the makespan in an identical parallel machine scheduling," *Math. Problems Eng.*, vol. 2018, Apr. 2018, Art. no. 3586731.

[19] H. S. Alamri, Y. A. Alsariera, and K. Z. Zamli, "Opposition-based whale optimization algorithm," *Adv. Sci. Lett.*, vol. 24, no. 10, pp. 7461–7464, 2018.

[20] H. S. Alamri, K. Z. Zamli, M. F. A. Razak, and A. Firdaus, "Solving 0/1 knapsack problem using opposition-based whale optimization algorithm (OWOA)," in *Proc. 8th Int. Conf. Softw. Comput. Appl.*, 2019, pp. 135–139.

[21] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Quasi-oppositional differential evolution," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Sep. 2007, pp. 2229–2236.

[22] C. Zhang, Z. Ni, Z. Wu, and L. Gu, "A novel swarm model with quasi-oppositional particle," in *Proc. Int. Forum Inf. Technol. Appl. (IFITA)*, May 2009, pp. 325–330.

[23] S. Sultana and P. K. Roy, "Multi-objective quasi-oppositional teaching learning based optimization for optimal location of distributed generator in radial distribution systems," *Int. J. Elect. Power Energy Syst.*, vol. 63, pp. 534–545, Dec. 2014.

[24] M. Ergezer, D. Simon, and D. Du, "Oppositional biogeography-based optimization," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2009, pp. 1009–1014.
[25] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
[26] S. Rahnamayan and G. G. Wang, "Center-based sampling for population-based algorithms," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2009, pp. 933–938.
[27] M. El-Abd, "Opposition-based artificial bee colony algorithm," in *Proc. 13th Annu. Conf. Genetic Evol. Comput. (GECCO)*, 2011, pp. 109–115.
[28] Z. Seif and M. B. Ahmadi, "An opposition-based algorithm for function optimization," *Eng. Appl. Artif. Intell.*, vol. 37, pp. 293–306, Jan. 2015.
[29] J. M. Enoch, "History of mirrors dating back 8000 years," *Optometry Vis. Sci.*, vol. 83, no. 10, pp. 775–781, 2006.
[30] K. Fleetwood, "An introduction to differential evolution," in *Proc. New Ideas Optim.*, 1999, pp. 79–108.
[31] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.
[32] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[33] A. Qing, *Differential Evolution: Fundamentals and Applications in Electrical Engineering*. Hoboken, NJ, USA: Wiley, 2009.
[34] X. Pan, L. Xue, Y. Lu, and N. Sun, "Hybrid particle swarm optimization with simulated annealing," *Multimedia Tools Appl.*, vol. 78 no. 138, pp. 1–16, Sep. 2018.
[35] M. Ventresca and H. R. Tizhoosh, "Simulated annealing with opposite neighbors," in *Proc. IEEE Symp. Found. Comput. Intell. (FOCI)*, Apr. 2007, pp. 186–192.

HAMMOUDEH S. ALAMRI received the B.Sc. degree in computer engineering from Yarmouk University, in 2013, and the M.Sc. degree from University Malaysia Pahang, in 2016, where he is currently pursuing the Ph.D. degree in computer science with the Faculty of Computer Systems and Software Engineering. His current research interests include optimization algorithms, the IoT, artificial intelligence, and machine learning.

KAMAL Z. ZAMLI received the degree in electrical engineering from Worcester Polytechnic Institute, USA, in 1992, the M.Sc. degree in real-time software engineering from Universiti Teknologi Malaysia, in 2000, and the Ph.D. degree in software engineering from the University of Newcastle upon Tyne, U.K., in 2003. He is currently a Professor and the Deputy Vice Chancellor of research and innovation with Universiti Malaysia Pahang. He has authored or coauthored more than 350 papers in journals and conferences worldwide mainly in the area of (combinatorial t-way) software testing, computational intelligence, and search-based software engineering.