International Journal of Innovative Computing, Information and Control
Volume **21**, Number **5**, October **2025**

ICIC International ©2025 ISSN 1349-4198
pp. **1255–1276**

DEVELOPMENT OF PARALLEL GREY WOLF OPTIMIZER FOR GLOBAL OPTIMIZATION

Supaporn Suwannarongsri

Department of Materials Handling and Logistics Engineering
Faculty of Engineering
King Mongkut's University of Technology North Bangkok
1518 Pracharaj-1 Road, Bangsue, Bangkok 10800, Thailand
supaporn.s@eng.kmutnb.ac.th

Received February 2025; revised May 2025

Abstract. *The grey wolf optimizer (GWO), one of the most powerful metaheuristic bio-inspired algorithms, has gained increasing popularity over the past decade for solving real-world optimization problems in engineering and science. As a popular metaheuristic optimization technique, numerous modified versions of the GWO have been developed to enhance its search performance. In this paper, a novel variant of GWO, named the parallel grey wolf optimizer (PGWO), is proposed. The PGWO is based on the original GWO, which mimics the leadership hierarchy and hunting behavior of grey wolves in nature, comprising four key mechanisms, i.e., encircling prey, hunting for prey, searching for prey, and attacking prey. The PGWO is specifically designed to run on a single-CPU platform, employing the original GWO as a search unit. It incorporates the partitioning mechanism (PM) to divide the entire search space into multiple non-overlapping sub-regions, allowing each GWO to independently explore its designated area. The sequencing mechanism (SM) is used to organize each GWO to run one-by-one on each iteration, while the discarding mechanism (DM) is introduced to eliminate the inferior GWOs in order to speed up the search process. The proposed PGWO algorithm is tested against ten selected benchmark functions and compared with the original GWO. From experimental results, it was found that the proposed PGWO performs superior performance in global optimization to the original GWO, with higher success rates, less search iterations, and less search times.*

Keywords: Parallel grey wolf optimizer, Grey wolf optimizer, Bio-inspired algorithm, Swarm intelligence, Global optimization

1. **Introduction.** In the last two decades, the bio-inspired (swarm intelligence) algorithms have become increasingly popular for solving science, engineering, and industrial problems. This is because these algorithms are flexible, versatile, and very efficient in solving nonlinear design problems with real-world applications [1]. Following the literature, the grey wolf optimizer (GWO) firstly proposed in 2014 is one of the most powerful metaheuristic bio-inspired algorithms [2]. The GWO algorithms mimic the leadership hierarchy and hunting mechanism of grey wolves (Canis Lupus) in nature consisting of four main mechanisms, i.e., encircling prey, hunting for prey, searching for prey, and attacking prey. It was tested against several well-known benchmark functions and compared with the particle swarm optimization (PSO), gravitational search algorithm (GSA), differential evolution (DE), evolutionary programming (EP), and evolution strategy (ES). Then, it was found that the GWO algorithm was able to provide very competitive results compared to those well-known metaheuristics [2]. Since then, the GWO has been widely applied to

DOI: 10.24507/ijicic.21.05.1255

a wide variety of optimization problems due to its impressive characteristics over other bio-inspired algorithms. In addition, the GWO has very few parameters, no derivation information, easy to use (simple implementation), rapid convergence, flexible, and capable to balance between the exploration and exploitation properties. From literature review [3-6], significant applications of the GWO include global optimization, machine learning, electrical and electronic engineering, control engineering, renewable energy, networking and communication, mechanical engineering, medical applications, classification, chemical engineering, petroleum engineering, industrial engineering, software engineering, civil engineering, geotechnical and geoenvironmental engineering, planing and scheduling, robotics, image processing, and so on. Also, the algorithms of the original GWO were modified to improve its search performance [3-6]. Variants of GWO are categorized into modified GWO [7-9], multi-objective GWO [10-12], hybrid GWO [13-17], and parallel GWO [18-23].

The modified GWO proposed in [7] focused on achieving a proper balance between exploration and exploitation, which led to improved algorithm performance. This was accomplished by replacing the linear vector used to switch between the searching for prey mechanism (exploration) and the attacking prey mechanism (exploitation) with a non-linear vector. The modified GWO proposed in [8] aimed to achieve a better balance between exploration and exploitation by incorporating the chaotic logistic map and opposition-based learning (OBL) to initialize the candidate solutions. These techniques helped avoid the drawbacks of random population initialization and enhanced the convergence rate of the algorithm. Additionally, DE was employed as a local search mechanism to improve exploitation, while a disruption operator (DO) was introduced to enhance the algorithm's exploratory capabilities. Meanwhile, the modified GWO proposed in [9] focused on adapting the convergence factor and introducing a weighted position factor to improve the overall search performance of the algorithm.

The multi-objective GWO proposed in [10] aimed to obtain Pareto-optimal solutions for ten multi-objective benchmark problems and compared its performance with two popular metaheuristic optimization algorithms. Similarly, the approach presented in [11] focused on solving twelve multi-objective benchmark problems and conducted a comparative analysis with two popular metaheuristic optimization algorithms. The study in [12] reviewed recent advances in multi-objective GWO and highlighted its applications across various domains.

The hybrid GWO proposed in [13] integrated GWO with DE to enhance search performance. It was evaluated on nine well-known benchmark functions and compared against PSO. Similarly, the approach in [14] combined GWO and DE, and was tested on twenty-three benchmark functions as well as a scheduling problem for a three-dimensional (3D) stacked system-on-chip (SoC). Its performance was compared to that of GWO, PSO, and DE. In [15], a hybrid GWO-PSO algorithm was proposed and evaluated on the single-area unit commitment problem using three test systems: a 14-bus system, a 30-bus system, and a 10-generator model. The selection of PSO for hybridization was attributed to its reputation as one of the most promising metaheuristic optimization algorithms [24-27]. The results were compared against PSO and several of its variants. The studies in [16,17] presented hybrid algorithms combining GWO with artificial neural networks (ANN). In [16], GWO was applied to optimizing the weights of a multilayer perceptron (MLP) to predict the tensile strength of siro-spun yarns based on fiber properties, while in [17], GWO was employed for melanoma detection, one of the most dangerous forms of human cancer.

The parallel GWO [18-23] is one of the most notable variants of the original GWO algorithm. From the literature review, the parallel GWO algorithms in [18-20] utilized a

technique that divided the entire wolf population into several sub-groups and employed a communication strategy to enable interactions among the sub-groups while searching for solutions in different regions of the search space. This concept was intended to enhance search performance. However, in certain scenarios, particularly in multimodal optimization problems, this approach potentially reduced the algorithm's effectiveness due to the fragmentation of the wolf population across different sub-groups. The parallel GWO algorithms described in [21-23] were developed to run on multicore processors and comprised multiple unmodified GWOs. Each GWO operated on the same global search space without algorithmic alterations. This configuration resembled a typical parallel personal computer setup in which multiple instances of the original GWO were executed independently. Moreover, the implementations in [21-23] required high-cost processors, making them less suitable for resource-constrained environments. Therefore, the development of a parallel GWO capable of running efficiently on a single-CPU platform remains a challenging task, because it is more difficult but lower-cost implementation.

For other parallel metaheuristics, such as the parallel whale optimization algorithm (PWOA) [28,29], parallel cuckoo search (PCS) [30,31], and parallel Harris hawks optimization (PHHO) [32], they were developed based on advancements in both hardware and software technologies. These approaches typically adopted a master-slave architecture for execution on multi-processor systems or cloud computing platforms. Notably, the core algorithms were either left unchanged or modified only slightly.

In this paper, the parallel grey wolf optimizer (PGWO) is developed to improve the search performance for running on a single-CPU. Once the original GWO is used as its search unit, the proposed PGWO possesses the partitioning mechanism (PM), sequencing mechanism (SM), and discarding mechanism (DM) as appearing in the multipath adaptive tabu search (MATS) [33,34], PCS [35,36], and parallel flower pollination algorithm (PFPA) [37,38]. The PM is utilized to divide an entire search area into many sub-search areas for each GWO to search for solutions in non-overlapping areas. The SM is used to organize the search units (GWOs) to run one-by-one on each iteration. The DM is conducted to eliminate the inferior GWOs to speed up the search process. This paper consists of five sections. After an introduction presented in Section 1, the rest of the paper is arranged as follows. The original GWO and the proposed PGWO algorithms are described in Section 2. Ten selected benchmark functions for global optimization are detailed in Section 3. Experimental results and discussions are illustrated in Section 4. Finally, the conclusions are given in Section 5.

2. PGWO Algorithm.

In this section, the algorithm of the original GWO is briefly described. Then, the proposed PGWO algorithm is elaborately given as follows.

2.1. Original GWO algorithm.

As previously mentioned, the original GWO algorithms mimic the leadership hierarchy and hunting mechanism of grey wolves in nature consisting of four main mechanisms, i.e., encircling prey, hunting for prey, searching for prey, and attacking prey [2]. Four gray wolf types, alpha (α), beta (β), delta (δ), and omega (ω), are used to model the leadership hierarchy within the pack as shown in Figure 1. The alpha (α) is the most powerful wolf standing for the best solution. The second and third best solutions are beta (β) and delta (δ), respectively. The rest of the candidate solutions are assumed to be omega (ω). In the original GWO algorithm, the hunting (optimization) is guided by alpha (α), beta (β), and delta (δ). Then, the omega (ω) wolves follow these three wolves [2].

The encircling prey mechanism is based on the mathematical relationships as given in Equations (1) and (2), where t stands for the current iteration, $\vec{A}$ and $\vec{C}$ stand for the

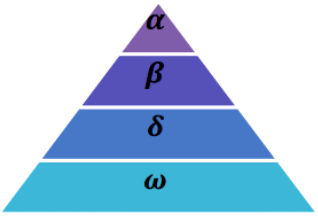


FIGURE 1. Hierarchy of grey wolf [2]

coefficient vectors, $\vec{X}_p$ stands for the position vector of the prey, and $\vec{X}$ stands for the position vector of a grey wolf.

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \tag{1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{2}$$

The vectors $\vec{A}$ and $\vec{C}$ can be calculated by Equations (3) and (4), respectively, where components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations, and r_1, r_2 are random vectors in $[0, 1]$.

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r}_1 - \vec{a} \tag{3}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{4}$$

The hunting for prey mechanism is based on the mathematical relationships as expressed in Equations (5)-(7).

$$\left.\begin{aligned} \vec{D}_\alpha &= \left|\vec{C}_1 \cdot \vec{X}_\alpha(t) - \vec{X}(t)\right| \\ \vec{D}_\beta &= \left|\vec{C}_2 \cdot \vec{X}_\beta(t) - \vec{X}(t)\right| \\ \vec{D}_\delta &= \left|\vec{C}_3 \cdot \vec{X}_\delta(t) - \vec{X}(t)\right| \end{aligned}\right\} \tag{5}$$

$$\left.\begin{aligned} \vec{X}_1(t) &= \vec{X}_\alpha(t) - \vec{A}_1 \cdot \left(\vec{D}_\alpha\right) \\ \vec{X}_2(t) &= \vec{X}_\beta(t) - \vec{A}_2 \cdot \left(\vec{D}_\beta\right) \\ \vec{X}_3(t) &= \vec{X}_\delta(t) - \vec{A}_3 \cdot \left(\vec{D}_\delta\right) \end{aligned}\right\} \tag{6}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1(t) + \vec{X}_2(t) + \vec{X}_3(t)}{3} \tag{7}$$

Referring to Equation (3), the fluctuation range of $\vec{A}$ is decreased by $\vec{a}$. In other words, $\vec{A}$ is a random value in the interval $[-a, a]$, where a is decreased from 2 to 0 over the course of iterations. Once random values of $\vec{A}$ are in $[-1, 1]$, the next position of a search agent can be in any position between its current position and the position of the prey. If $|A| \geq 1$, the searching for prey mechanism (exploration) will be invoked, and if $|A| < 1$, the attacking prey mechanism (exploitation) will be activated [2]. The algorithm of the original GWO can be represented by the pseudo code as visualized in Figure 2.

2.2. **Proposed PGWO algorithm.** The PGWO is proposed to improve the search performance of the original GWO for running on a single-CPU. The algorithm of the proposed PGWO in this paper differs from that of the parallel GWO in [18-20]. The parallel GWO in [18-20] utilized the technique to divide the entire wolf population into several sub-groups which probably reduces the ability to search for solutions due to the

Initialize:
- Initialize the objective function $f(\boldsymbol{x})$ and search spaces
- Initialize the grey wolf population X_i (i = 1,2,…,n)
- Initialize a, A, and C
- Evaluate all agents X_i via $f(\boldsymbol{x})$
- Ranking X_α = the best agent, X_β = the second best agent, and X_δ = the third best agent

while ($t \leq$ Max number of iterations)
  for i = 1 : n (all n grey wolf population)
    - Update the position of the current search by using (7)
  end for
  - Update a, A, and C
  - Evaluate all agents X_i via $f(\boldsymbol{x})$
  - Update X_α, X_β and X_δ
  - $t = t + 1$
end while
- Report the best solution X_α

FIGURE 2. Pseudo code of the original GWO algorithm

wolf population being divided into different sub-groups, while the entire wolf population of the proposed PGWO in this paper are not divided. Therefore, the search performance of the proposed PGWO will not be reduced. In addition, the algorithm of the proposed PGWO in this paper differs from that of the parallel GWO in [21-23]. The parallel GWO in [21-23] was developed for running on multicore (high-cost) processors, while the proposed PGWO in this paper is developed for running on a simple single-CPU platform. By utilizing the original GWO algorithm shown in Figure 2 as its search unit, the proposed PGWO algorithm consists of PM, SM, and DM mechanisms as following details.

The PM serves to divide an entire search space into many sub-search-spaces defined by the user. The number of sub-search spaces should be determined based on the specific characteristics of the problem of interest. Furthermore, both symmetrical and asymmetrical configurations can be employed, depending on the nature of the problem. Bearing in mind that too many sub-search-spaces would result in a slow search process. The PM is also conducted to randomly generate an individual initial solution for each GWO. The PM procedures can be summarized and represented by the pseudo code as shown in Figure 3. In detail, the PM proposed for the PGWO in this paper differs from that of the MATS [33,34], PCS [35,36], and the PFPA [37,38]. For the PM of the MATS [33,34], PCS [35,36], and PFPA [37,38], it will remove the sub-search-spaces (sub-boundaries) before launching the search, such that all search agents could search freely on the entire search space to reduce any conflicts that may arise during the search along the border lines of boundaries. However, the PM of the proposed PGWO shown in Figure 3 does not have

PM Procedures:
Step 1: Load number of GWO_i, i = 1,…, N, i.e., GWO_1, GWO_2,…, GWO_N, and the entire search space.
Step 2: Partition the entire search space into N sub-search-spaces.
Step 3: Generate initial solutions for each sub-search-space.

FIGURE 3. Pseudo code of the PM procedures

the procedures to remove the sub-search-spaces. This is because defining the sub-search-spaces explicitly can avoid any conflicts that may arise during the search along the border lines of boundaries. Also, maintaining the sub-search-spaces for GWOs can speed up the search process.

The SM serves to organize the search units (GWOs) to run one-by-one on a single iteration approach providing the hardware having a single CPU. As represented by the simple diagram in Figure 4, the SM can be considered as the time sharing with multiple points single strategy (MPSS) used in the MATS [33,34], PCS [35,36], and the PFPA [37,38]. Assuming that the proposed PGWO possesses GWO_i, $i = 1, \ldots, N$, the first GWO (GWO_1) begins its first iteration. Afterward, it goes to the wait state. Once the CPU finishes its service to the GWO_1, it provides the service to the second GWO (GWO_2). Once the GWO_2 finishes its first iteration, it has to wait. The operation goes on in this manner until the GWO_N finishes its first iteration. The CPU then returns to service the GWO_1 for its second iteration, then the GWO_2, GWO_3, ..., and GWO_N in sequential manner. The operation is repeated until one of the GWOs hits the optimal solution, or the termination criteria (TC) are met. With this sequential manner, the PGWO algorithm is suitable for running on a single CPU platform. Nonetheless, it can be easily adapted for use on multi-core CPU or parallel platforms. The SM will communicate with the DM mechanism. Some inferior GWOs will be discarded by the DM to speed up the search process. After the DM eliminates some inferior GWOs, the existing GWOs will be transferred to SM. Then, the SM will launch only existing GWOs to be operated one-by-one for each iteration in a sequential manner as early explained. The SM procedures can be summarized and represented by the pseudo code as shown in Figure 5.

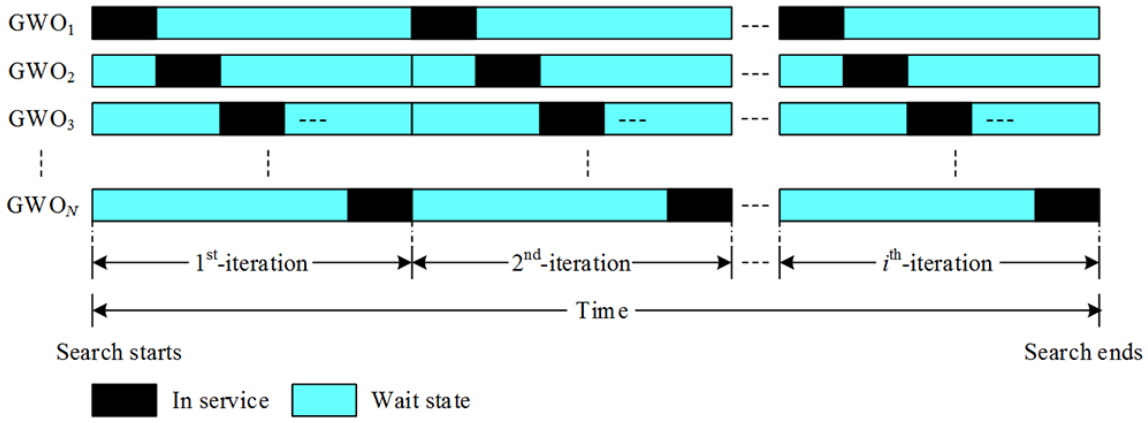


FIGURE 4. Time sharing with MPSS in SM

SM Procedures:
Step 1: Communicate with DM to receive the existing GWOs.
Step 2: Launch all existing GWOs one-by-one for each iteration in a sequential manner.
Step 3: If the TC are met, exit the search, otherwise go to Step 1.

FIGURE 5. Pseudo code of the SM procedures

The DM serves to discard some inferior GWOs to reduce the search time or speed up the search process. Several approaches can be conducted to force some inferior GWOs to stop as soon as possible. For a simple implementation as appearing in the MATS [33,34], PCS [35,36], and the PFPA [37,38], the DM will discard some GWOs based on the cost

values of their current best-solutions. The number of GWOs will be reduced by half in each time once the DM is activated. After this forced termination made to the inferior GWOs, the DM transfers the existing GWOs to the SM. The DM procedures can be summarized and represented by the pseudo code as shown in Figure 6.

DM Procedures:
Step 1: Load cost values of the current-best solutions of all GWOs.
Step 2: Do min-max sorting of cost values.
Step 3: Keep GWOs from the top to the middle of the sorted list as active, and discard the rest of GWOs.
Step 4: Transfer existing GWOs to the SM.

FIGURE 6. Pseudo code of the DM procedures

The procedures of the proposed PGWO can be summarized and represented by the pseudo code as shown in Figure 7, while the algorithm of the proposed PGWO can be visualized by the flow diagram shown in Figure 8, where GWO_1, GWO_2, ..., and GWO_N are the GWO algorithms represented in Figure 2.

PGWO Procedures:
Step 1: Initialize the objective function and entire search spaces, number of GWOs, Max_Iter, and Iter = 1.
Step 2: Activate the PM to decompose the search domain, and initialize all GWOs situated to sub-search-spaces.
Step 3: Invoke the SM to perform sequential search by using GWO_1, GWO_2,..., GWO_N, for $N = N - K$, $K \leq N - 1$, and $N_{min} = 1$.
Step 4: Activate the DS to force some low-quality GWOs to stop.
Step 5: If the TC is met (Iter > Max_Iter), exit with the global solution, otherwise update Iter = Iter + 1, and go to Step 3.

FIGURE 7. Pseudo code of the proposed PGWO procedures

3. **Benchmark Functions.** To perform its effectiveness, the proposed PGWO will be tested against ten selected benchmark functions for global minimization. They are selected from the difference of their characteristics, i.e., nonlinearity, symmetry and unsymmetry, as well as multi-modality. Such ten selected benchmark functions include 1) Ackley function (AF), 2) De Jong function (DJF), 3) Drop-Wave function (DWF), 4) Griewank function (GF), 5) Lévy function (LF), 6) Michaelwicz function (MF), 7) Salomon function (SalF), 8) Shekel's fox-holes function (SkF), 9) Yang's first function (Y1F), and 10) Yang's second function (Y2F) [39-42]. Details of these selected benchmark functions including their names, functions, search spaces, coefficients, optimal solutions ($\boldsymbol{x}^*$) optimal function values $f(\boldsymbol{x}^*)$, and 2D-surfaces are summarized in Table 1, where d is the dimension.

4. **Experimental Results and Discussions.** This section reports the search performance of the proposed PGWO algorithm against ten selected benchmark functions as detailed in the previous section for global optimization (minimization). The benchmark functions are referred shortly as the AF, DJF, DWF, GF, LF, MF, SalF, SkF, Y1F, and Y2F, respectively, summarized in Table 1. Performance comparisons are made among the original GWO and the proposed PGWO with 2, 4, 8, 16, 32, and 64 GWOs denoted as

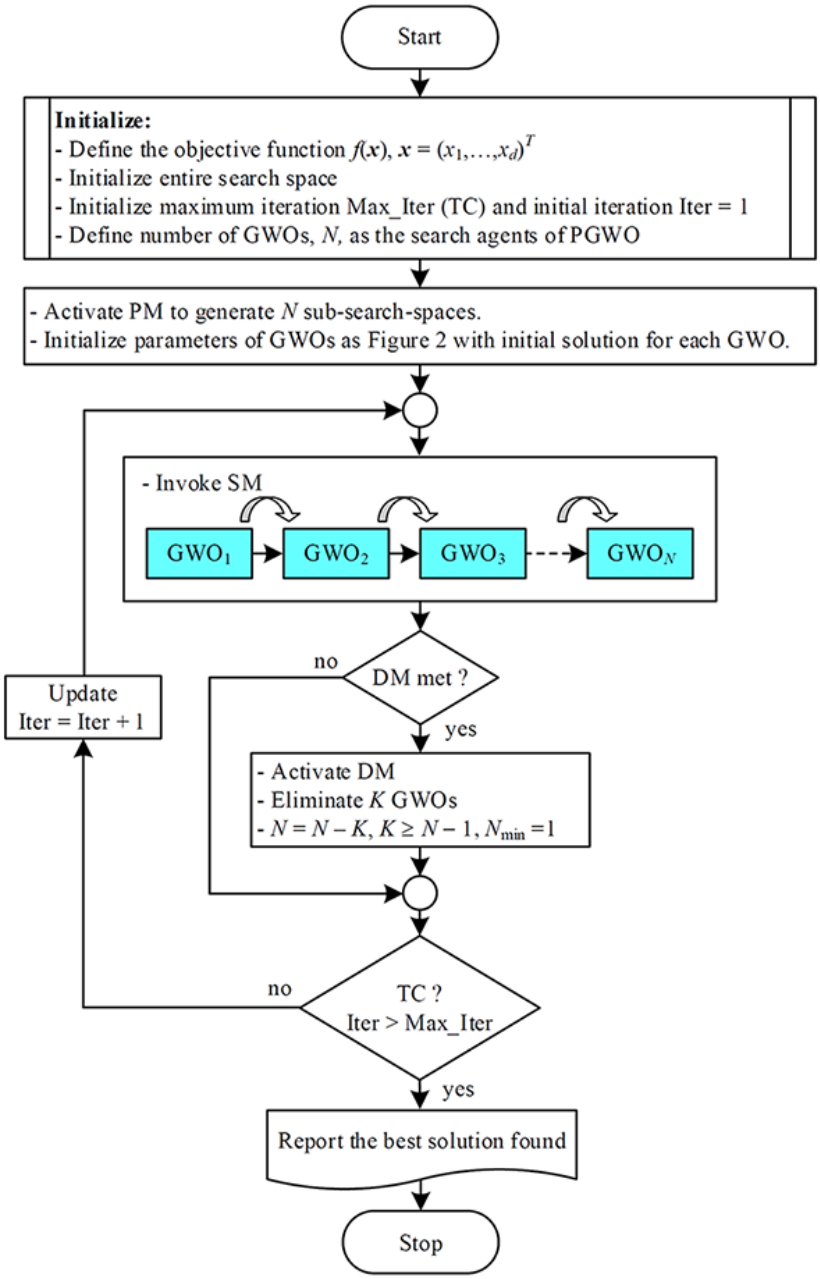
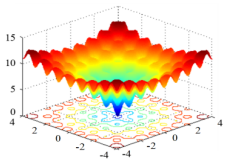
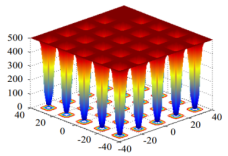
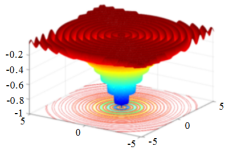
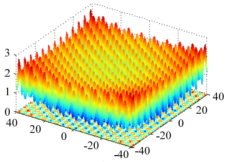
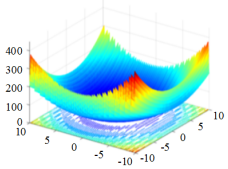
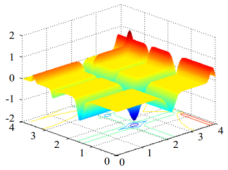
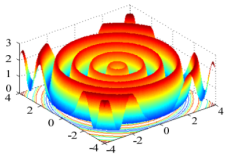
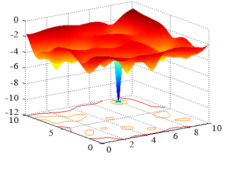
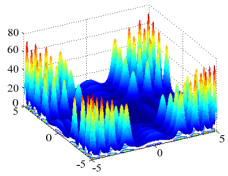
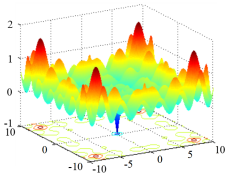


FIGURE 8. Flow diagram of the proposed PGWO algorithm

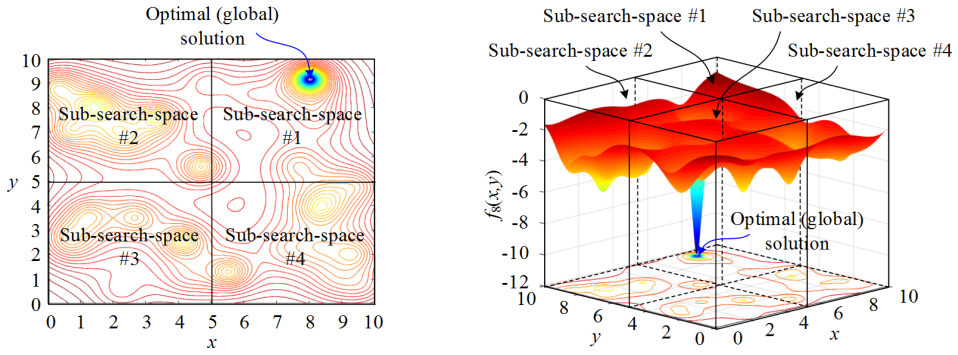
PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64, respectively. Both the original GWO and the proposed PGWO algorithms were coded by MATLAB version 2018b run on the Intel(R) Core(TM) i5-3470 CPU@3.60GHz, 4.0GB-RAM. The key searching parameter of the original GWO in Figure 2 is the number of grey wolf population n. In the preliminary studies, $n = 10, 20, 30, \ldots, 200$ are tested against ten selected benchmark functions. From the preliminary tests, it was found that the best value of n is 20-40 allowing the original GWO algorithm can reach the optimal solutions of all benchmark functions quickly with different initial solutions due to the random process. Thus, $n = 30$ is set for the original GWO. For a fair comparison, $n = 30$ is also set for all GWOs in the PGWO algorithm. Both the original GWO and the proposed PGWO will be run with 100 trials to find the best solution of each function, and terminated once one of two TCs is met, i.e., TC1 the function values are less than a given tolerance $\varepsilon \leq 10^{-12}$

TABLE 1. Selected benchmark functions

Benchmark functions	Functions, Coefficients, Optimal solutions ($\boldsymbol{x}^*$), and Optimal function values $f(\boldsymbol{x}^*)$	Search spaces	2D-surfaces
Ackley function (AF)	$f_1(\boldsymbol{x}) = -20\exp\left[-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right] - \exp\left[\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right] +20+e$ $f_1(\boldsymbol{x}^*) = (0,0,\ldots,0) = 0$	$-32.768 \le x_i \le 32.768$	
De Jong function (DJF)	$f_2(\boldsymbol{x}) = \left[1/500 + \sum_{j=1}^{24}\left\{1\Big/\sum_{i=1}^{d}(x_i - a_{ij})^6\right\}\right]^{-1}$ $a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \ldots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \ldots & 32 & 32 & 32 \end{pmatrix}$ $f_2(\boldsymbol{x}^*) = (-32,-32,\ldots,-32) = 0.9980$	$-40 \le x_i \le 40$	
Drop-Wave function (DWF)	$f_3(\boldsymbol{x}) = -\frac{1+\cos\left(12\sqrt{\sum_{i=1}^{d}x_i^2}\right)}{\frac{1}{2}\sum_{i=1}^{d}x_i^2+2}$ $f_3(\boldsymbol{x}^*) = (0,0,\ldots,0) = -1$	$-5.12 \le x_i \le 5.12$	
Griewank function (GF)	$f_4(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{d}x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ $f_4(\boldsymbol{x}^*) = (0,0,\ldots,0) = 0$	$-600 \le x_i \le 600$	
Lévy function (LF)	$f_5(\boldsymbol{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i-1)^2\left[1+10\sin^2(\pi w_i - 1)\right] + (w_d-1)^2\left[1+\sin^2(2\pi w_d)\right]$, $w_i = 1 + \left(\frac{x_i-1}{4}\right)$, $f_5(\boldsymbol{x}^*) = (1,1,\ldots,1) = 0$	$-10 \le x_i \le 10$	
Michaelwicz function (MF)	$f_6(\boldsymbol{x}) = -\sum_{i=1}^{d}\sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}$, $(m=10)$ $f_6(\boldsymbol{x}^*) = (2.20319, 1.57049) = -1.8013$ for $d=2$, $f_6(\boldsymbol{x}^*) = -4.6877$ for $d=5$, and $f_6(\boldsymbol{x}^*) = -9.6602$ for $d=10$	$0 \le x_i \le \pi$	
Salomon function (SalF)	$f_7(\boldsymbol{x}) = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{d}x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{n}x_i^2}$ $f_7(\boldsymbol{x}^*) = (0,0,\ldots,0) = 0$	$-100 \le x_i \le 100$	
Shekel's fox-holes function (SkF)	$f_8(\boldsymbol{x}) = -\sum_{j=1}^{30}\left[1\Big/\left\{\sum_{i=1}^{d}(x_i-a_{ij})^2 + c_j\right\}\right]$, $a_{ij} = \begin{pmatrix} 9.6810 & 9.4000 & 8.0250 & \ldots & 4.1380 \\ 0.6670 & 2.0410 & 9.1520 & \ldots & 2.5620 \end{pmatrix}$, $c_j = (\,0.806 \quad 0.517 \quad 0.100 \quad 0.908 \quad \ldots \quad 0.326\,)$, $f_8(\boldsymbol{x}^*) = (8.0241, 9.1465) = -12.1190$ for $d=2$, $f_8(\boldsymbol{x}^*) = -10.4056$ for $d=5$, and $f_8(\boldsymbol{x}^*) = -10.2088$ for $d=10$	$0 \le x_i \le 10$	
Yang's first function (Y1F)	$f_9(\boldsymbol{x}) = \left(\sum_{i=1}^{d}\lvert x_i\rvert\right)\exp\left[-\sum_{i=1}^{d}\sin\left(x_i^2\right)\right]$ $f_9(\boldsymbol{x}^*) = (0,0,\ldots,0) = 0$	$-2\pi \le x_i \le 2\pi$	
Yang's second function (Y2F)	$f_{10}(\boldsymbol{x}) = \left\{\left[\sum_{i=1}^{d}\sin^2(x_i)\right] - \exp\left(-\sum_{i=1}^{d}x_i^2\right)\right\}\exp\left(-\sum_{i=1}^{d}\sin^2\sqrt{\lvert x_i\rvert}\right)$ $f_{10}(\boldsymbol{x}^*) = (0,0,\ldots,0) = -1$	$-10 \le x_i \le 10$	

from $f(\boldsymbol{x}^*)$ of each function shown in Table 1 or TC2 the current iteration (Iter) is greater than the maximum iteration (Max_Iter = 1,000). The former criterion implies that the search is success, while the later means that the search is not success. For the proposed PGWO, the PM, SM and DM mechanisms are set as following details.

For the PM setting, a symmetrical partitioning technique is employed in this work to divide the entire search space into non-overlapping sub-search-spaces for 2, 4, 8, 16, 32, 64 GWOs for each function. For instance, the surface of the SkF, when partitioned into 4 sub-search-spaces for PGWO#4, is illustrated in Figure 9. The boundaries of search spaces are defined as $[x_{\min} \leq x \leq x_{\max}; y_{\min} \leq y \leq y_{\max}]$. Accordingly, the entire search space of the SkF can be defined as $[0 \leq x \leq 10; 0 \leq y \leq 10]$. For the PGWO#4 consisting of 4 GWOs, the SkF is decomposed for non-overlapping sub-domains into $\#1[5 \leq x \leq 10; 5 \leq y \leq 10]$ for the 1st sub-search space, $\#2[0 \leq x < 5; 5 \leq y \leq 10]$ for the 2nd sub-search space, $\#3[0 \leq x < 5; 0 \leq y < 5]$ for the 3rd sub-search space and $\#4[5 \leq x \leq 10; 0 \leq y < 5]$ for the 4th sub-search space, respectively. Then, the GWO is assigned to each sub-search-space, while the boundaries are not removed. This means that the search-spaces (boundaries) of GWO_1, GWO_2, GWO_3, and GWO_4 are $[5 \leq x \leq 10; 5 \leq y \leq 10]$, $[0 \leq x < 5; 5 \leq y \leq 10]$, $[0 \leq x < 5; 0 \leq y < 5]$, and $[5 \leq x \leq 10; 0 \leq y < 5]$, respectively. It is clear that when each GWO is assigned to a specific sub-search space, the boundaries of the original entire search space are preserved and remain unchanged. The sub-search-spaces of the SkF partitioned by the PS for PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64, are summarized in Table 2. This approach is also applied to other functions, as it effectively avoids boundary conflicts, buffer zones, and explicit constraints commonly encountered in overlapping partitioning techniques.



(a) Contour (b) Surface

FIGURE 9. Partitioned surface of the SkF (2D) for PGWO#4

In practical applications, alternative geometrical configurations and coordinate systems can be utilized for the partitioning process to suit the applications.

For the SM setting, it is simply set by a time-sharing technique for all GWOs running one-by-one as sequential manner in each iteration based on a single-CPU. As an example, the PGWO#4 consists of 4 GWOs. In the 1st iteration, the GWO_1 will be run, while GWO_2, GWO_3, and GWO_4 are in waiting state. When the GWO_1 finishes its process, it goes to the waiting state. At this time, the GWO_2 will be run, while the GWO_1, GWO_3, and GWO_4 are in waiting state. The operation goes on in this manner until the GWO_4 finishes its process. This means that the 1st iteration is finished. Afterward, the CPU then returns to start the 2nd iteration by running GWO_1, GWO_2, GWO_3, and GWO_4 in sequential manner again. The operation is repeated until one of the GWOs hits the optimal solution or some GWOs are discarded by the DM. The SM operation of

TABLE 2. Sub-search-spaces of SkF partitioned by PS for PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32 and PGWO#64

Entire search space	Sub-search-spaces of SkF defined by PS						
	PGWO#2	PGWO#4	PGWO#8	PGWO#16	PGWO#32	PGWO#64	
$[0 \le x \le 10; 0 \le y \le 10]$	#1$[5 \le x \le 10; 0 \le y \le 10]$	#1$[5 \le x \le 10; 5 \le y \le 10]$	#1$[7.5 \le x \le 10; 5 \le y \le 10]$	#1$[7.5 \le x \le 10; 7.5 \le y \le 10]$	#1$[8.75 \le x \le 10; 7.5 \le y \le 10]$	#1$[8.75 \le x \le 10; 8.75 \le y \le 10]$	#33$[8.75 \le x \le 10; 3.75 \le y < 5]$
	#2$[0 \le x < 5; 5 \le y \le 10]$	#2$[0 \le x < 5; 5 \le y \le 10]$	#2$[5 \le x < 7.5; 5 \le y \le 10]$	#2$[5 \le x < 7.5; 7.5 \le y \le 10]$	#2$[7.5 \le x < 8.75; 7.5 \le y \le 10]$	#2$[7.5 \le x < 8.75; 8.75 \le y \le 10]$	#34$[7.5 \le x < 8.75; 3.75 \le y < 5]$
	#2$[0 \le x < 5; 0 \le y \le 10]$	#3$[0 \le x < 5; 0 \le y < 5]$	#3$[2.5 \le x < 5; 5 \le y \le 10]$	#3$[2.5 \le x < 5; 7.5 \le y \le 10]$	#3$[6.25 \le x < 7.5; 7.5 \le y \le 10]$	#3$[6.25 \le x < 7.5; 8.75 \le y \le 10]$	#35$[6.25 \le x < 7.5; 3.75 \le y < 5]$
		#4$[5 \le x \le 10; 0 \le y < 5]$	#4$[0 \le x < 2.5; 5 \le y \le 10]$	#4$[0 \le x < 2.5; 7.5 \le y \le 10]$	#4$[5 \le x < 6.25; 7.5 \le y \le 10]$	#4$[5 \le x < 6.25; 8.75 \le y \le 10]$	#36$[5 \le x < 6.25; 3.75 \le y < 5]$
			#5$[0 \le x < 2.5; 5 \le y < 7.5]$	#5$[0 \le x < 2.5; 5 \le y < 7.5]$	#5$[3.75 \le x < 5; 7.5 \le y \le 10]$	#5$[3.75 \le x < 5; 8.75 \le y \le 10]$	#37$[3.75 \le x < 5; 3.75 \le y < 5]$
			#6$[2.5 \le x < 5; 5 \le y < 7.5]$	#6$[2.5 \le x < 5; 5 \le y < 7.5]$	#6$[2.5 \le x < 3.75; 7.5 \le y \le 10]$	#6$[2.5 \le x < 3.75; 8.75 \le y \le 10]$	#38$[2.5 \le x < 3.75; 3.75 \le y < 5]$
			#7$[5 \le x < 7.5; 0 \le y < 5]$	#7$[5 \le x < 7.5; 5 \le y < 7.5]$	#7$[1.25 \le x < 2.5; 7.5 \le y \le 10]$	#7$[1.25 \le x < 2.5; 8.75 \le y \le 10]$	#39$[1.25 \le x < 2.5; 3.75 \le y < 5]$
			#8$[7.5 \le x \le 10; 0 \le y < 5]$	#8$[7.5 \le x \le 10; 5 \le y < 7.5]$	#8$[0 \le x < 1.25; 7.5 \le y \le 10]$	#8$[0 \le x < 1.25; 8.75 \le y \le 10]$	#40$[0 \le x < 1.25; 3.75 \le y < 5]$
				#9$[0 \le x < 1.25; 5 \le y < 7.5]$	#9$[0 \le x < 1.25; 7.5 \le y < 8.75]$	#41$[0 \le x < 1.25; 2.5 \le y < 3.75]$	
				#10$[1.25 \le x < 2.5; 5 \le y < 7.5]$	#10$[1.25 \le x < 2.5; 7.5 \le y < 8.75]$	#42$[1.25 \le x < 2.5; 2.5 \le y < 3.75]$	
				#11$[2.5 \le x < 3.75; 5 \le y < 7.5]$	#11$[2.5 \le x < 3.75; 7.5 \le y < 8.75]$	#43$[2.5 \le x < 3.75; 2.5 \le y < 3.75]$	
				#12$[3.75 \le x < 5; 5 \le y < 7.5]$	#12$[3.75 \le x < 5; 7.5 \le y < 8.75]$	#44$[3.75 \le x < 5; 2.5 \le y < 3.75]$	
				#13$[5 \le x < 6.25; 5 \le y < 7.5]$	#13$[5 \le x < 6.25; 7.5 \le y < 8.75]$	#45$[5 \le x < 6.25; 2.5 \le y < 3.75]$	
				#14$[6.25 \le x < 7.5; 5 \le y < 7.5]$	#14$[6.25 \le x < 7.5; 7.5 \le y < 8.75]$	#46$[6.25 \le x < 7.5; 2.5 \le y < 3.75]$	
				#15$[7.5 \le x < 8.75; 5 \le y < 7.5]$	#15$[7.5 \le x < 8.75; 7.5 \le y < 8.75]$	#47$[7.5 \le x < 8.75; 2.5 \le y < 3.75]$	
				#16$[8.75 \le x \le 10; 5 \le y < 7.5]$	#16$[8.75 \le x \le 10; 7.5 \le y < 8.75]$	#48$[8.75 \le x \le 10; 2.5 \le y < 3.75]$	
				#17$[8.75 \le x \le 10; 2.5 \le y < 5]$	#17$[8.75 \le x \le 10; 6.25 \le y < 7.5]$	#49$[8.75 \le x \le 10; 1.25 \le y < 2.5]$	
				#18$[7.5 \le x < 8.75; 2.5 \le y < 5]$	#18$[7.5 \le x < 8.75; 6.25 \le y < 7.5]$	#50$[7.5 \le x < 8.75; 1.25 \le y < 2.5]$	
				#19$[6.25 \le x < 7.5; 2.5 \le y < 5]$	#19$[6.25 \le x < 7.5; 6.25 \le y < 7.5]$	#51$[6.25 \le x < 7.5; 1.25 \le y < 2.5]$	
				#20$[5 \le x < 6.25; 2.5 \le y < 5]$	#20$[5 \le x < 6.25; 6.25 \le y < 7.5]$	#52$[5 \le x < 6.25; 1.25 \le y < 2.5]$	
				#21$[3.75 \le x < 5; 2.5 \le y < 5]$	#21$[3.75 \le x < 5; 6.25 \le y < 7.5]$	#53$[3.75 \le x < 5; 1.25 \le y < 2.5]$	
				#22$[2.5 \le x < 3.75; 2.5 \le y < 5]$	#22$[2.5 \le x < 3.75; 6.25 \le y < 7.5]$	#54$[2.5 \le x < 3.75; 1.25 \le y < 2.5]$	
				#23$[1.25 \le x < 2.5; 2.5 \le y < 5]$	#23$[1.25 \le x < 2.5; 6.25 \le y < 7.5]$	#55$[1.25 \le x < 2.5; 1.25 \le y < 2.5]$	
				#24$[0 \le x < 1.25; 2.5 \le y < 5]$	#24$[0 \le x < 1.25; 6.25 \le y < 7.5]$	#56$[0 \le x < 1.25; 1.25 \le y < 2.5]$	
				#25$[0 \le x < 1.25; 0 \le y < 2.5]$	#25$[0 \le x < 1.25; 5 \le y < 6.25]$	#57$[0 \le x < 1.25; 0 \le y < 1.25]$	
				#26$[1.25 \le x < 2.5; 0 \le y < 2.5]$	#26$[1.25 \le x < 2.5; 5 \le y < 6.25]$	#58$[1.25 \le x < 2.5; 0 \le y < 1.25]$	
				#27$[2.5 \le x < 3.75; 0 \le y < 2.5]$	#27$[2.5 \le x < 3.75; 5 \le y < 6.25]$	#59$[2.5 \le x < 3.75; 0 \le y < 1.25]$	
				#28$[3.75 \le x < 5; 0 \le y < 2.5]$	#28$[3.75 \le x < 5; 5 \le y < 6.25]$	#60$[3.75 \le x < 5; 0 \le y < 1.25]$	
				#29$[5 \le x < 6.25; 0 \le y < 2.5]$	#29$[5 \le x < 6.25; 5 \le y < 6.25]$	#61$[5 \le x < 6.25; 0 \le y < 1.25]$	
				#30$[6.25 \le x < 7.5; 0 \le y < 2.5]$	#30$[6.25 \le x < 7.5; 5 \le y < 6.25]$	#62$[6.25 \le x < 7.5; 0 \le y < 1.25]$	
				#31$[7.5 \le x < 8.75; 0 \le y < 2.5]$	#31$[7.5 \le x < 8.75; 5 \le y < 6.25]$	#63$[7.5 \le x < 8.75; 0 \le y < 1.25]$	
				#32$[8.75 \le x \le 10; 0 \le y < 2.5]$	#32$[8.75 \le x \le 10; 5 \le y < 6.25]$	#64$[8.75 \le x \le 10; 0 \le y < 1.25]$	

PGWO#2, PGWO#8, PGWO#16, PGWO#32, and PGWO#64, can be described by that of PGWO#4.

For the DM setting, various possible approaches can be used to force some inferior GWOs to stop. In this work, the number of GWOs will be reduced by half in each time once the DM is activated. Regarding to PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64, let them be PGWO#N, N = 2, 4, 8, 16, 32, and 64, respectively. Let K be the number of DM activations. K can be expressed in Equation (8). From Equation (8), the number of DM activations for PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64, are 1, 2, 3, 4, 5, and 6, respectively. Let q_i be the iteration at which the DM is invoked, and let D_i be the number of GWOs remaining after DM is activated each time. q_i and D_i can be calculated by Equations (9) and (10), respectively. The "Round" in Equation (9) stands for rounding towards nearest integer. Once Max_Iter = 1,000 is set, K, q_i and D_i of PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64 can be represented as shown in Figure 10. Referring to Figure 10, it was found that eventually there is only one GWO

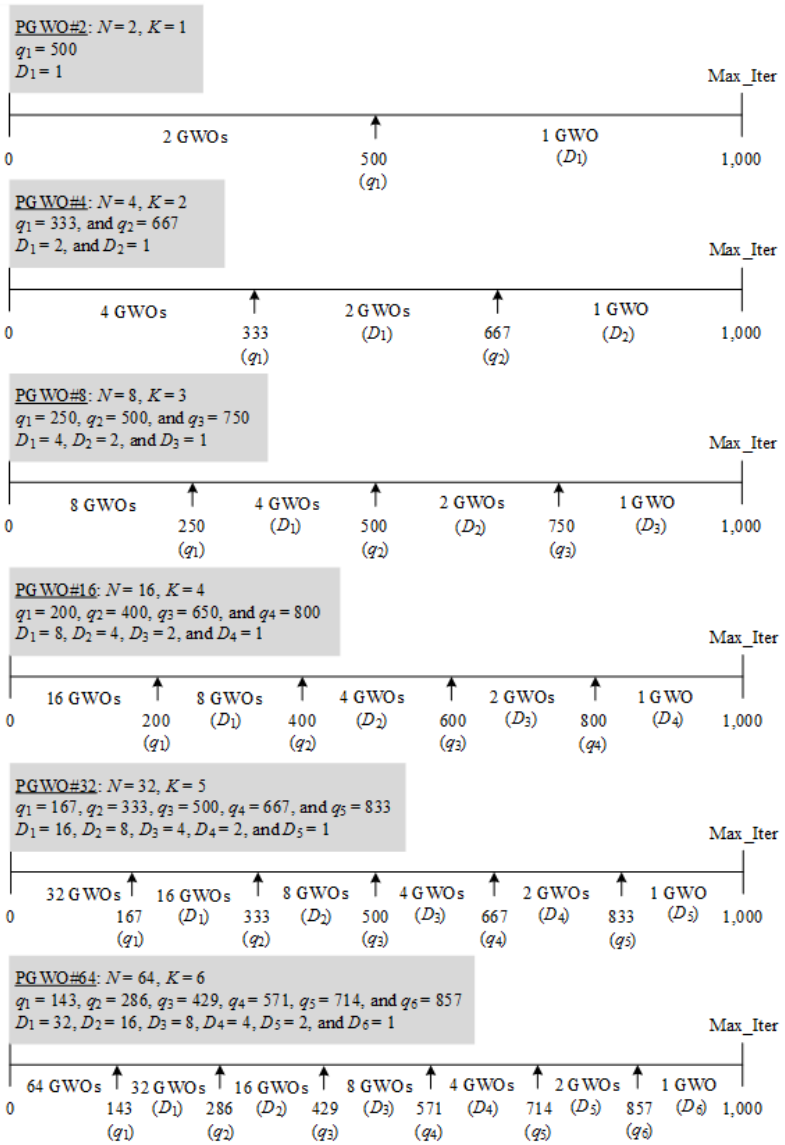


FIGURE 10. Discarding approach in DM setting

left to continue searching for the optimal solution.

$$N = 2^K, \quad N = 2, 4, 8, 16, 32, 64 \tag{8}$$

$$q_i = \text{Round}\left[\frac{(\text{Max\_Iter})i}{(K+1)}\right], \quad i = 1, \ldots, K \tag{9}$$

$$D_i = \frac{N}{2^i}, \quad i = 1, \ldots, K \tag{10}$$

Experimental results obtained by the original GWO and the proposed PGWO algorithms over all ten selected two-dimensional (2D) benchmark functions from 100 trials are given as follows. Table 3 reveals the percentages of the success rate (PSR) of the original GWO and the proposed PGWOs. The PSR informs that the algorithms are terminated by TC1, i.e., the optimal solutions are found. Regarding to Table 3, it was found

Table 3. PSR of GWO and PGWOs

Benchmark functions	Original GWO	Proposed PGWOs					
		PGWO#2	PGWO#4	PGWO#8	PGWO#16	PGWO#32	PGWO#64
AF	74%	75%	94%	97%	100%	100%	100%
DJF	78%	81%	99%	100%	100%	100%	100%
DWF	73%	84%	100%	100%	100%	100%	100%
GF	74%	77%	98%	100%	100%	100%	100%
LF	72%	76%	97%	100%	100%	100%	100%
MF	82%	87%	100%	100%	100%	100%	100%
SalF	70%	73%	93%	98%	100%	100%	100%
SkF	81%	84%	100%	100%	100%	100%	100%
Y1F	79%	83%	100%	100%	100%	100%	100%
Y2F	75%	76%	95%	99%	100%	100%	100%
Averages	75.80%	79.60%	97.60%	99.40%	100.00%	100.00%	100.00%

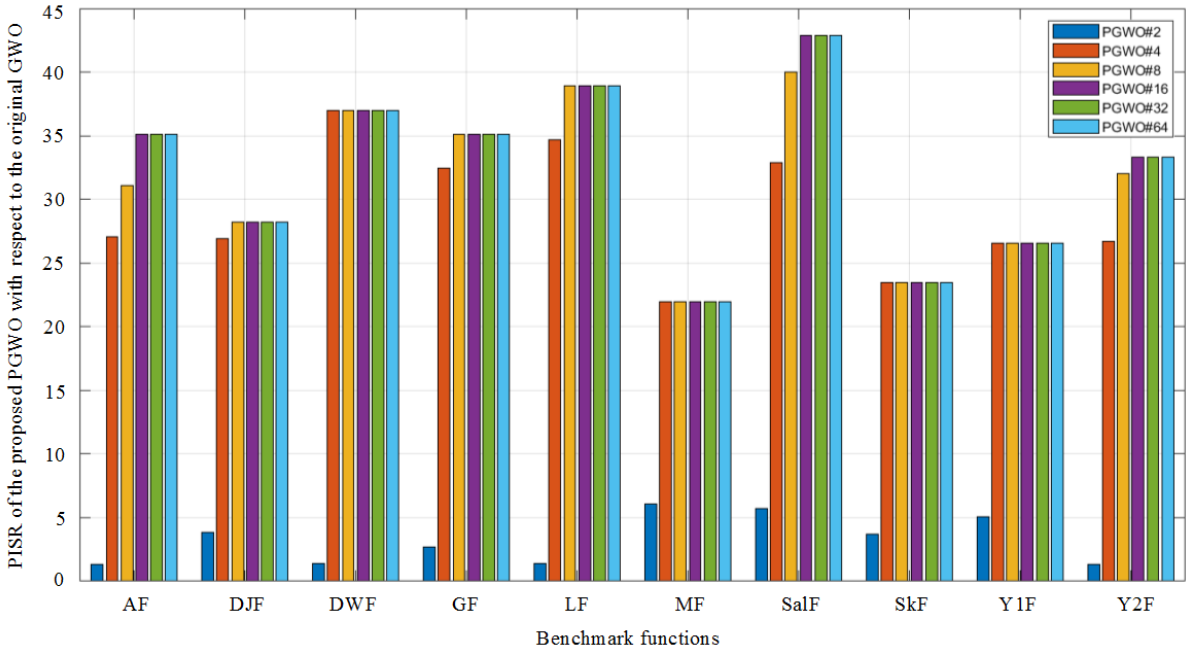


Figure 11. PISR of PGWOs with respect to the original GWO

that the proposed PGWOs can satisfactorily increase the PSR. The numeric data in Table 3 are converted into percentage increase of the success rate (PISR) of the proposed PGWO with respect to the original GWO by using the mathematical relation as stated in Equation (11), where $\mathrm{PSR}_{\mathrm{GWO}}$ is the PSR of the original GWO, $\mathrm{PSR}_{\mathrm{PGWO}}$ is the PSR of the proposed PGWOs, and $\mathrm{PISR}_{\mathrm{PGWO}}$ is the PISR of the proposed PGWOs with respect to the original GWO. The converted data obtained by Equation (11) are plotted by the bar graphs as shown in Figure 11 to give a clear view of the merits of the proposed PGWO over ten selected benchmark functions. From Equation (11) and Figure 11, it can be observed that the PGWO#2, PGWO#4, and PGWO#8, and PGWO#16 can averagely increase the PSR by 3.26%, 28.96%, 31.43%, and 32.25%, respectively, once compared with the original GWO. This means that the more the number of GWOs employed in the PGWO, the more the value of PISR. Readers can also observe that the PGWO#16, PGWO#32, and PGWO#64 have the same PISR values, i.e., 32.25%, 32.25% and 32.25%, respectively. This is because the proposed PGWOs have PSR = 100% for all ten selected benchmark functions since PGWO#16 as appearing in Table 3.

$$\mathrm{PISR}_{\mathrm{PGWO}} = \left(\frac{\mathrm{PSR}_{\mathrm{PGWO}} - \mathrm{PSR}_{\mathrm{GWO}}}{\mathrm{PSR}_{\mathrm{GWO}}} \right) \times 100 \tag{11}$$

Table 4 shows the average search iterations (ASI) of the original GWO and the proposed PGWOs over all ten selected benchmark functions from 100 trials. Referring to Table 4, the proposed PGWOs consume the ASI less than the original GWO. The numeric data in Table 4 are transformed into percentage decrease of average search iterations (PDASI) of the proposed PGWOs with respect to the original GWO by using the mathematical relation as stated in Equation (12), where $\mathrm{ASI}_{\mathrm{GWO}}$ is the ASI of the original GWO, $\mathrm{ASI}_{\mathrm{PGWO}}$ is the ASI of the proposed PGWOs, and $\mathrm{PDAIS}_{\mathrm{PGWO}}$ is the PDASI of the proposed PGWOs with respect to the original GWO. The converted data obtained by Equation (12) are displayed by the bar graphs as shown in Figure 12 reflecting the merits of the PGWO. From Equation (12) and Figure 12, it can be noticed that the PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64 can averagely decrease the ASI by 9.25%, 20.12%, 53.80%, 64.22%, 71.91%, and 77.03%, respectively, once compared with the original GWO. This implies that the more the number of GWOs utilized

Table 4. ASI of GWO and PGWOs

Benchmark functions	Original GWO	Proposed PGWO					
		PGWO#2	PGWO#4	PGWO#8	PGWO#16	PGWO#32	PGWO#64
AF	712.46	652.37	582.47	327.16	254.22	194.52	160.94
DJF	682.53	614.22	545.49	308.44	232.76	191.40	158.86
DWF	754.30	673.16	616.12	366.37	271.24	216.14	172.07
GF	723.27	655.54	598.04	341.02	267.10	211.85	175.82
LF	798.74	718.35	613.66	363.61	294.33	228.87	183.55
MF	615.82	583.76	497.30	296.40	218.78	186.02	148.03
SalF	816.16	724.81	632.93	378.35	302.42	224.38	187.44
SkF	707.48	629.32	561.47	316.83	253.57	190.64	155.73
Y1F	721.59	668.48	574.61	320.96	248.31	193.20	157.29
Y2F	709.51	645.09	558.05	325.74	251.28	195.26	162.46
Averages	724.19	656.51	578.01	334.49	259.40	203.23	166.22

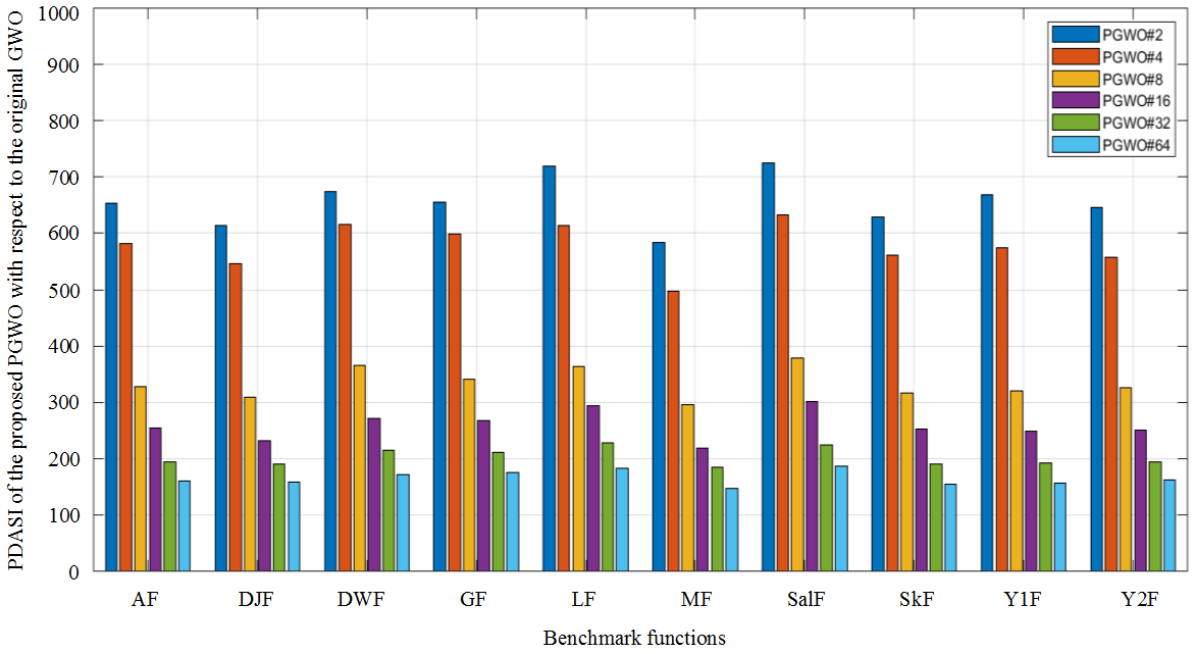


FIGURE 12. PDASI of PGWOs with respect to the original GWO

in the PGWO, the more the value of PDASI.

$$\mathrm{PDASI}_{\mathrm{PGWO}} = \left(\frac{\mathrm{ASI}_{\mathrm{GWO}} - \mathrm{ASI}_{\mathrm{PGWO}}}{\mathrm{ASI}_{\mathrm{GWO}}}\right) \times 100 \tag{12}$$

Table 5 reveals the average search time (AST) of the original GWO and the proposed PGWOs over all ten selected benchmark functions from 100 trials. Regarding Table 5, it was found that the proposed PGWOs consume the AST greater than the original GWO. This is because the PGWO is proposed for running on a single-CPU platform. However, the AST of the PGWOs can be reduced by implementation for running on multicore processors. To prove this, the numeric data in Table 5 are converted into the equivalent values of the AST (EAST) with respect to the original GWO by using the mathematical relation as stated in Equation (13), where N is the number of GWOs, $\mathrm{AST}_{\mathrm{GWO}\#N}$ is the AST of the Nth GWO, and $\mathrm{EAST}_{\mathrm{PGWO}\#N}$ is the EAST of the Nth PGWO. The

TABLE 5. AST in seconds of GWO and PGWOs

Benchmark functions	Original GWO	Proposed PGWO					
		PGWO#2	PGWO#4	PGWO#8	PGWO#16	PGWO#32	PGWO#64
AF	1.58×10^{-1}	2.42×10^{-1}	3.58×10^{-1}	4.40×10^{-1}	5.91×10^{-1}	7.05×10^{-1}	8.48×10^{-1}
DJF	2.72×10^{-1}	3.98×10^{-1}	4.72×10^{-1}	5.72×10^{-1}	6.96×10^{-1}	8.83×10^{-1}	9.72×10^{-1}
DWF	2.24×10^{-1}	3.50×10^{-1}	4.76×10^{-1}	5.53×10^{-1}	6.88×10^{-1}	8.76×10^{-1}	9.55×10^{-1}
GF	2.06×10^{-1}	3.21×10^{-1}	4.34×10^{-1}	5.01×10^{-1}	6.23×10^{-1}	8.21×10^{-1}	9.23×10^{-1}
LF	2.15×10^{-1}	3.48×10^{-1}	4.66×10^{-1}	5.37×10^{-1}	6.54×10^{-1}	8.56×10^{-1}	9.44×10^{-1}
MF	1.47×10^{-1}	2.33×10^{-1}	3.40×10^{-1}	4.38×10^{-1}	5.80×10^{-1}	6.94×10^{-1}	8.23×10^{-1}
SalF	2.51×10^{-1}	3.74×10^{-1}	4.89×10^{-1}	5.62×10^{-1}	6.72×10^{-1}	7.58×10^{-1}	8.80×10^{-1}
SkF	2.23×10^{-1}	3.40×10^{-1}	4.63×10^{-1}	5.18×10^{-1}	6.26×10^{-1}	7.32×10^{-1}	8.61×10^{-1}
Y1F	1.98×10^{-1}	3.03×10^{-1}	4.15×10^{-1}	4.97×10^{-1}	6.10×10^{-1}	7.96×10^{-1}	9.10×10^{-1}
Y2F	1.65×10^{-1}	2.59×10^{-1}	3.57×10^{-1}	4.34×10^{-1}	5.87×10^{-1}	6.93×10^{-1}	8.36×10^{-1}
Averages	2.06×10^{-1}	3.17×10^{-1}	4.27×10^{-1}	5.05×10^{-1}	6.33×10^{-1}	7.81×10^{-1}	8.95×10^{-1}

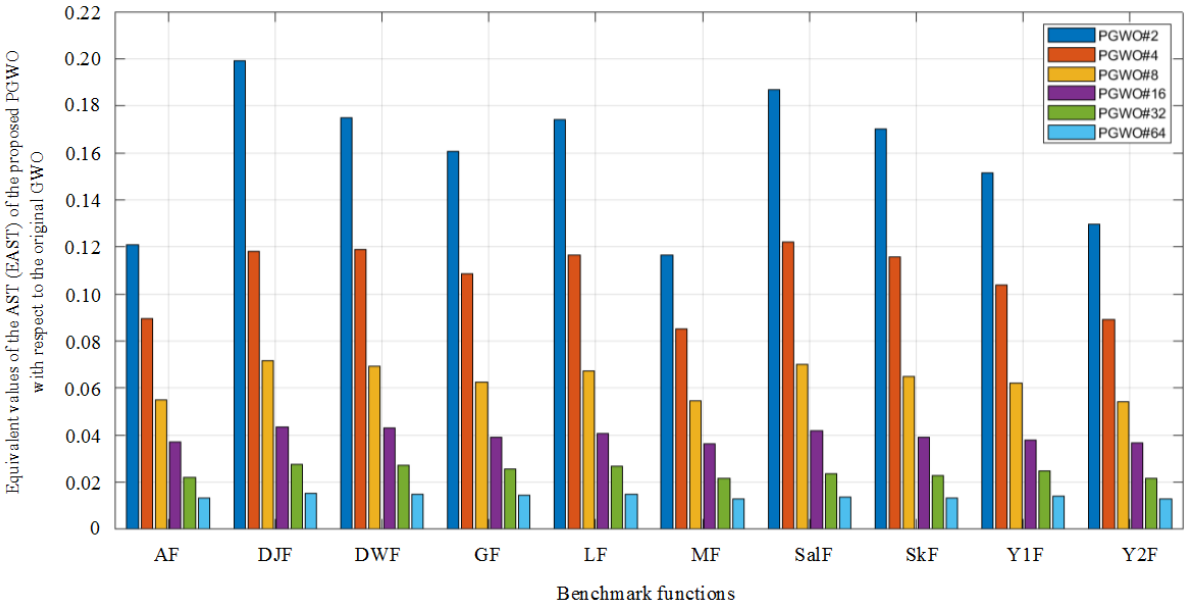


FIGURE 13. EAST of PGWOs with respect to the original GWO

converted data obtained by Equation (13) are depicted by the bar graphs as shown in Figure 13. From Equation (13) and Figure 13, it can be noticed that the PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64 averagely consume the EAST with respect to the original GWO of 0.158, 0.107, 0.063, 0.040, 0.024, and 0.014 s, respectively, while the original GWO consumes the AST of 0.206 s.

$$\mathrm{EAST}_{\mathrm{PGWO}\#N} = \frac{\mathrm{AST}_{\mathrm{GWO}\#N}}{N} \tag{13}$$

The percentage decrease of AST (PDAST) of the equivalent PGWOs with respect to the original GWO by using the mathematical relation as expressed in Equation (14) for comparison purposes is conducted. The PDAST obtained by Equation (14) is plotted by the bar graphs as shown in Figure 14 reflecting the merits of the PGWO. From Equation (14) and Figure 14, it can be noticed that the equivalent PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64 averagely consume the AST less than the original GWO by 23.30%, 48.06%, 69.42%, 80.58%, 88.35%, and 93.20%, respectively.

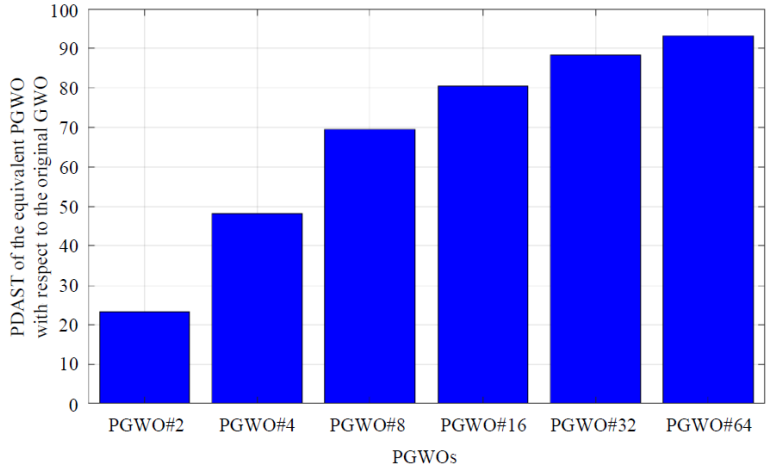


FIGURE 14. PDAST of equivalent PGWOs with respect to the original GWO

This means that the more the number of GWOs used in the PGWO, the more the value of PDAST.

$$\mathrm{PDAST}_{\mathrm{PGWO}} = \left(\frac{\mathrm{AST}_{\mathrm{GWO}} - \mathrm{AST}_{\mathrm{PGWO}}}{\mathrm{AST}_{\mathrm{GWO}}} \right) \times 100 \tag{14}$$

For instance, the function values $f(\boldsymbol{x})$ of the SkF obtained by the original GWO and the proposed PGWOs are monitored and plotted by the convergent curves as illustrated in Figure 15. The convergent curves of other benchmark functions are omitted because they have a similar form to those in Figure 15. For the SkF, the accepted solutions must have the function values $f(\boldsymbol{x})$ of less than $-12.1190 + \varepsilon = -12.1190 + 1 \times 10^{-12} = -12.118999999999 \approx -12.1190$. Referring to Figure 15, the original GWO hits the optimal solution within 708 iterations, whereas the PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64 hit the optimal solution within 630, 562, 317, 254, 191, 156 iterations, respectively. It can be noticed that the proposed PGWO demonstrates superior performance of global minimization compared to the original GWO.

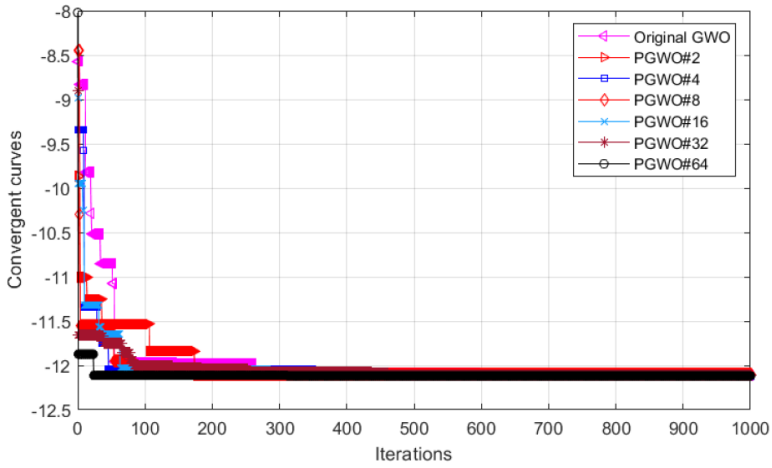


FIGURE 15. Convergent curves of GWO and PGWOs

For the proposed PGWO, the PM, SM, and DM mechanisms are designed to operate collaboratively for running on a single-CPU platform in order to achieve fully optimal search performance. In this context, the operation of PGWO without the SM mechanism is not feasible. However, the effects of the PM and DM mechanisms on search performance are investigated to complement the revealed results. In this investigation, the proposed PGWO algorithm is divided into three configurations, i.e., the full PGWO, PGWO without DM, and PGWO with temporary PM. The full PGWO refers to the original proposed PGWO algorithm incorporating all three mechanisms (PM, SM, and DM). The PGWO without DM retains the PM and SM mechanisms but excludes the DM. The PGWO with temporary PM includes SM and DM, but the PM is applied only at the initialization stage to generating individual initial solutions for all GWOs in the PGWO. After initialization, the PM removes the sub-search-spaces before launching the search, allowing all search units (GWOs) to operate freely across the entire search space. To evaluate their performance, ten selected 2D benchmark functions, including AF, DJF, DWF, GF, LF, MF, SalF, SkF, Y1F, and Y2F, are conducted. The parameter settings for the original GWO, PGWOs, and TC follow those specified earlier. Results of this investigation obtained from PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64 over 100 trials are summarized in Tables 6 and 7, where PGWO^1 stands for the full PGWO, PGWO^2 stands for the PGWO without DM, and PGWO^3 stands for the PGWO with temporary PM. Table 6 presents the PSR values, whereas Table 7 reports the number

TABLE 6. PSR of PGWO[1] (full PGWO), PGWO[2] (PGWO without DM), and PGWO[3] (PGWO with temporary PM)

Benchmark functions	PGWO#2			PGWO#4			PGWO#8		
	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]
AF	75%	75%	74%	94%	94%	91%	97%	98%	95%
DJF	81%	82%	79%	99%	99%	93%	100%	100%	99%
DWF	84%	85%	84%	100%	100%	99%	100%	100%	99%
GF	77%	77%	76%	98%	98%	96%	100%	100%	98%
LF	76%	76%	75%	97%	98%	95%	100%	100%	100%
MF	87%	87%	84%	100%	100%	100%	100%	100%	100%
SalF	73%	75%	73%	93%	93%	91%	98%	98%	96%
SkF	84%	84%	82%	100%	100%	100%	100%	100%	99%
Y1F	83%	83%	81%	100%	100%	99%	100%	100%	98%
Y2F	76%	75%	74%	95%	95%	93%	99%	99%	97%
Averages	79.60%	79.90%	78.20%	97.60%	97.70%	95.70%	99.40%	99.50%	98.10%
Benchmark functions	PGWO#16			PGWO#32			PGWO#64		
	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]
AF	100%	100%	96%	100%	100%	98%	100%	100%	99%
DJF	100%	100%	99%	100%	100%	99%	100%	100%	99%
DWF	100%	100%	100%	100%	100%	100%	100%	100%	100%
GF	100%	100%	99%	100%	100%	99%	100%	100%	99%
LF	100%	100%	100%	100%	100%	100%	100%	100%	100%
MF	100%	100%	100%	100%	100%	100%	100%	100%	100%
SalF	100%	100%	97%	100%	100%	99%	100%	100%	99%
SkF	100%	100%	99%	100%	100%	99%	100%	100%	99%
Y1F	100%	100%	98%	100%	100%	98%	100%	100%	100%
Y2F	100%	100%	98%	100%	100%	99%	100%	100%	99%
Averages	100.00%	100.00%	98.60%	100.00%	100.00%	99.10%	100.00%	100.00%	99.40%

of function evaluations (NFE). The NFE value is indicative of the computational time consumed during the search process. This implies that the lower the NFE, the shorter the search time consumed.

Referring to Table 6, it was observed that PGWO[1] and PGWO[2] yield very similar PSR values, whereas PGWO[3] performs a lower PSR when compared to both PGWO[1] and PGWO[2]. Once considering the results in Table 7, it was found among them that PGWO[2] requires the highest NFE, followed by PGWO[3], while PGWO[1] requires the lowest NFE. These results indicate that the DM mechanism significantly contributes to speeding up the search process by discarding inferior GWOs. Furthermore, the proposed PM mechanism enhances the search performance of the proposed PGWO for global optimization more effectively than the temporary PM configuration.

To further evaluate and reinforce the global optimization performance of the proposed PGWO, the experiments have been extended to high-dimensional benchmark functions. As shown in Table 1, the selected benchmark functions including AF ($d = 128$), DJF ($d = 10$), DWF ($d = 16$), GF ($d = 10$), LF ($d = 10$), MF ($d = 10$), SalF ($d = 30$), SkF ($d = 10$), Y1F ($d = 16$), and Y2F ($d = 16$), are employed for this evaluation. Performance comparisons are conducted among the original GWO and the PGWOs (PGWO#2, PGWO#4, PGWO#8, PGWO#16, PGWO#32, and PGWO#64). The parameter settings for the

TABLE 7. NFE of PGWO[1] (full PGWO), PGWO[2] (PGWO without DM), and PGWO[3] (PGWO with temporary PM)

Benchmark functions	PGWO#2			PGWO#4			PGWO#8		
	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]
AF	11,781	23,680	17,318	10,521	42,189	17,254	5,984	47,932	11,310
DJF	11,088	22,287	16,299	9,849	39,494	16,152	5,632	45,112	10,644
DWF	12,159	24,440	17,874	11,130	44,631	18,253	6,688	53,571	12,640
GF	11,844	23,806	17,411	10,794	43,284	17,702	6,226	49,870	11,767
LF	12,978	26,086	19,078	11,088	44,463	18,184	6,644	53,218	12,557
MF	10,542	21,189	15,497	8,988	36,042	14,740	5,412	43,350	10,229
SalF	13,083	26,297	19,232	11,424	45,810	18,735	6,908	55,333	13,056
SkF	11,361	22,836	16,701	10,143	40,673	16,635	5,786	46,346	10,936
Y1F	12,075	24,271	17,750	10,374	41,600	17,013	5,852	46,875	11,060
Y2F	11,655	23,427	17,133	10,080	40,421	16,531	5,940	47,579	11,227
Averages	1.19×10^4	2.38×10^4	1.74×10^4	1.04×10^4	4.19×10^4	1.71×10^4	6.11×10^3	4.89×10^4	1.15×10^4
Benchmark functions	PGWO#16			PGWO#32			PGWO#64		
	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]	PGWO[1]	PGWO[2]	PGWO[3]
AF	5,064	81,075	9,723	4,563	146,062	8,989	4,340	277,803	8,767
DJF	4,632	74,158	8,893	4,509	144,333	8,883	4,278	273,835	8,642
DWF	5,400	86,454	10,368	5,076	162,483	10,000	4,650	297,647	9,393
GF	5,328	85,301	10,230	4,968	159,026	9,787	4,743	303,599	9,581
LF	5,856	93,755	11,244	5,373	171,990	10,585	4,960	317,490	10,019
MF	4,368	69,932	8,387	4,374	140,012	8,617	3,999	255,976	8,078
SalF	6,024	96,444	11,566	5,265	168,533	10,372	5,053	323,443	10,207
SkF	5,040	80,690	9,677	4,482	143,469	8,830	4,185	267,882	8,454
Y1F	4,944	79,153	9,492	4,536	145,197	8,936	4,247	271,850	8,579
Y2F	5,016	80,306	9,631	4,590	146,926	9,042	4,371	279,788	8,829
Averages	5.17×10^3	8.27×10^4	9.92×10^3	4.77×10^3	1.53×10^5	9.40×10^3	4.48×10^3	2.87×10^5	9.05×10^3

original GWO, PGWOs, and TC follow those previously specified, with the exception of Max_Iter in TC2, which is set to 2,000.

The PSR values obtained from the original GWO and the proposed PGWOs over 100 trials are summarized in Table 8. According to Table 8, it was found that the proposed PGWO significantly improves the PSR, indicating enhanced search performance. These results further confirm the performance and the effectiveness of the proposed PGWO for global optimization. The ASI and AST values are omitted in this evaluation, as their trends are closely aligned with those presented in Tables 4 and 5. Consequently, the trends of PISR, PDASI, EAST, and PDAST for the PGWOs with respect to the original GWO exhibit strong similarities to those illustrated in Figures 11, 12, 13, and 14, respectively.

5. **Conclusions.** In this paper, the PGWO algorithm has been proposed for global optimization (minimization). The PGWO has been developed for running on a single-CPU platform. By utilizing the original GWO as its search unit, the proposed PGWO possesses three main mechanisms, i.e., the PM used to divide an entire search space into many subsearch spaces for all GWOs, the SM employed to organize the search units to run one-by-one on each iteration and the DM conducted to discard some inferior GWOs. To perform its effectiveness, the PGWO has been tested against ten selected benchmark functions for global minimization compared with the original GWO. As experimental results against 2D and high-dimensional benchmark functions, the proposed PGWO performs more efficient with higher success rates, less search iterations and less search times, than the original GWO. The DM could speed up the search process, significantly, while the PM could enhance the search performance of the proposed PGWO for global optimization

Table 8. PSR of GWO and PGWOs for high-dimensional benchmark functions

Benchmark functions	Original GWO	Proposed PGWOs					
		PGWO#2	PGWO#4	PGWO#8	PGWO#16	PGWO#32	PGWO#64
AF ($d = 128$)	68%	71%	89%	96%	99%	100%	100%
DJF ($d = 10$)	72%	78%	96%	100%	100%	100%	100%
DWF ($d = 16$)	64%	81%	97%	99%	100%	100%	100%
GF ($d = 10$)	67%	72%	95%	98%	100%	100%	100%
LF ($d = 10$)	61%	68%	94%	100%	100%	100%	100%
MF ($d = 10$)	75%	83%	100%	100%	100%	100%	100%
SalF ($d = 30$)	58%	69%	91%	97%	98%	100%	100%
SkF ($d = 10$)	74%	81%	99%	100%	100%	100%	100%
Y1F ($d = 16$)	73%	79%	100%	100%	100%	100%	100%
Y2F ($d = 16$)	69%	72%	93%	97%	100%	100%	100%
Averages	68.10%	75.40%	95.40%	98.70%	99.70%	100.00%	100.00%

effectively. It can be concluded that the proposed PGWO is one of the most efficient metaheuristic optimization techniques for global optimization running on a single-CPU. Moreover, the more the number of GWOs utilized in the PGWO, the higher the search performance. For future research, the PGWO will be applied to solving various real-world logistic and industrial engineering optimization problems including the scheduling problem (SP), assembly line balancing problem (ALBP), multiple knapsack problem (MKP), multiple cutting stock problem (MCSP), multiple vehicle routing problems (MVRP), and multiple-depots multiple-vehicle routing problems (MDMVRP).

Acknowledgment. This paper was funded by King Mongkut's University of Technology North Bangkok with contract no. KMUTNB-68-BASIC-60.

REFERENCES

[1] X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi and M. Karamanoglu, *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, Elsevier Inc., 2013.

[2] S. Mirjalili, S. M. Mirjalili and A. Lewis, Grey wolf optimizer, *Advances in Engineering Software*, vol.69, pp.46-61, 2014.

[3] H. Faris, I. Aljarah, M. A. Al-Betar and S. Mirjalili, Grey wolf optimizer: A review of recent variants and applications, *Neural Computing and Applications*, vol.30, pp.413-435, 2018.

[4] Y. Liu, A. As'arry, M. K. Hassan, A. A. Hairuddin and H. Mohamad, Review of the grey wolf optimization algorithm: Variants and applications, *Neural Computing and Applications*, vol.36, pp.2713-2735, 2024.

[5] S. N. Makhadmeh, M. A. Al-Betar, I. A. Doush, M. A. Awadallah, S. Kassaymeh, S. Mirjalili and R. A. Zitar, Recent advances in grey wolf optimizer, its versions and applications: Review, *IEEE Access*, vol.12, pp.22991-23028, 2024.

[6] M. H. Nadimi-Shahraki, H. Zamani, Z. A. Varzaneh, A. S. Sadiq and S. Mirjalili, A systematic review of applying grey wolf optimizer, its variants, and its developments in different Internet of Things applications, *Internet of Things*, vol.26, 101135, 2024.

[7] N. Mittal, U. Singh and B. S. Sohi, Modified grey wolf optimizer for global engineering optimization, *Applied Computational Intelligence and Soft Computing*, vol.2016, 2016.

[8] R. A. Ibrahim, M. A. Elaziz and S. Lu, Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization, *Expert Systems with Applications*, vol.108, pp.1-27, 2018.

[9] W. Xiao, H. Deng, Y. Sheng and L. Hu, Factored grey wolf optimizer with application to resource-constrained project scheduling, *International Journal of Innovative Computing, Information and Control*, vol.14, no.3, pp.881-897, 2018.

[10] S. Mirjalili, S. Saremi, S. M. Mirjalili and L. S. Coelho, Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization, *Expert Systems with Applications*, vol.47, pp.106-119, 2016.

[11] K. Jiang, H. Ni, R. Han and X. Wang, An improved multi-objective grey wolf optimizer for dependent task scheduling in edge computing, *International Journal of Innovative Computing, Information and Control*, vol.15, no.6, pp.2289-2304, 2019.

[12] S. N. Makhadmeh, O. A. Alomari, S. Mirjalili, M. A. Al-Betar and A. Elnagar, Recent advances in multi-objective grey wolf optimizer, its versions and applications, *Neural Computing and Applications*, vol.34, no.22, pp.19723-19749, 2022.

[13] D. Jitkongchuen, A hybrid differential evolution with grey wolf optimizer for continuous global optimization, *Proc. of the 7th International Conference on Information Technology and Electrical Engineering*, pp.51-54, 2015.

[14] A. Zhu, C. Xu, Z. Li, J. Wu and Z. Liu, Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC, *Journal of Systems Engineering and Electronics*, vol.26, no.2, pp.317-328, 2015.

[15] V. K. Kamboj, A novel hybrid PSO-GWO approach for unit commitment problem, *Neural Computing and Applications*, vol.27, no.6, pp.1643-1655, 2016.

[16] E. Hadavandi, S. Mostafayi and P. Soltani, A grey wolf optimizer-based neural network coupled with response surface method for modeling the strength of siro-spun yarn in spinning mills, *Applied Soft Computing*, vol.72, pp.1-13, 2018.

[17] A. Parsian, M. Ramezani and N. Ghadimi, A hybrid neural network-gray wolf optimization algorithm for melanoma detection, *Biomedical Research*, vol.28, no.8, pp.3408-3411, 2017.

[18] T.-S. Pan, T.-K. Dao, T.-T. Nguyen and S.-C. Chu, A communication strategy for paralleling grey wolf optimizer, *Genetic and Evolutionary Computing*, vol.388, pp.253-262, 2015.

[19] M. S. Nasrabadi, Y. Sharafi and M. Tayari, A parallel grey wolf optimizer combined with opposition-based learning, *Proc. of the 1st Conference on Swarm Intelligence and Evolutionary Computation*, pp.18-23, 2016.

[20] H. Chen, L. Han, Z. Hu, Q. Hou, Z. Ye and J. Zeng, A feature selection method of parallel grey wolf optimization algorithm based on spark, *Proc. of the 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2019.

[21] J. Jayapriya and M. Arock, A parallel GWO technique for aligning multiple molecular sequences, *Proc. of the International Conference on Advances in Computing, Communications and Informatics*, pp.210-215, 2015.

[22] S. Younesi, B. Ahmadi, O. Ceylan and A. Ozdemir, Allocation of distributed generators using parallel grey wolf optimization, *Proc. of the 9th International Conference on Modern Power Systems*, 2021.

[23] R. Jarray, M. Al-Dhaifallah, H. Rezk and S. Bouallègue, Parallel cooperative coevolutionary grey wolf optimizer for path planning problem of unmanned aerial vehicles, *Sensors*, vol.22, no.1826, pp.1-29, 2022.

[24] H. Ye, J. Guo and X. Li, Task migration for cloudlet federation based on improved particle swarm optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.20, no.3, pp.693-707, 2024.

[25] A. A. Ilham, E. Warni and Pahrul, Soft voting classifier with optimized weight using particle swarm optimization on sentiment analysis for online credit and loan application reviews, *International Journal of Innovative Computing, Information and Control*, vol.20, no.2, pp.359-372, 2024.

[26] F. Li and T. Li, Economic modeling and simulation in the industrial supply chain model using particle swarm optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.20, no.1, pp.163-179, 2024.

[27] A. G. Gad, Particle swarm optimization algorithm and its applications: A systematic review, *Archives of Computational Methods in Engineering*, vol.29, pp.2531-2561, 2022.

[28] Z. A. Khan, I. A. Aziz, N. A. B. Osman and S. Nabi, Parallel enhanced whale optimization algorithm for independent tasks scheduling on cloud computing, *IEEE Access*, vol.12, pp.23529-23548, 2024.

[29] Q. Jiang, Y. Guo, Z. Yang and X. Zhou, A parallel whale optimization algorithm and its implementation on FPGA, *Proc. of the IEEE Congress on Evolutionary Computation*, 2020.

[30] N. Tzy-Luen, Y. T. Keat and R. Abdullah, Parallel cuckoo search algorithm on OpenMP for traveling salesman problem, *Proc. of the 3rd International Conference on Computer and Information Sciences*, 2016.

[31] M. Subotic, M. Tuba, N. Bacanin and D. Simian, Parallelized cuckoo search algorithm for unconstrained optimization, *Proc. of the World Congress: Applied Computing Conference*, pp.151-156, 2012.
[32] A. G. Hussien, L. Abualigah, R. A. Zitar, F. A. Hashim, M. Amin, A. Saber, K. H. Almotairi and A. H. Gandomi, Recent advances in Harris Hawks optimization: A comparative study and applications, *Electronics*, vol.11, no.12(1919), pp.1-50, 2022.
[33] J. Kluabwang, D. Puangdownreong and S. Sujitjorn, Multipath adaptive tabu search for a vehicle control problem, *Journal of Applied Mathematics*, vol.2012, pp.1-20, 2012.
[34] D. Puangdownreong, J. Kluabwang and S. Sujitjorn, Multipath adaptive tabu search: Its convergence and application to identification problem, *Journal of Physical Sciences*, vol.7, no.33, pp.5288-5296, 2012.
[35] S. Suwannarongsri, Parallel cuckoo search for global optimization, *International Journal of Innovative Computing, Information and Control*, vol.17, no.3, pp.887-903, 2021.
[36] S. Suwannarongsri, Optimal solving multi-vehicle routing problems via parallel cuckoo search, *International Journal of Innovative Computing, Information and Control*, vol.17, no.6, pp.1921-1935, 2021.
[37] P. Khluabwannarat and D. Puangdownreong, Parallel flower pollination algorithm and its application to fractional-order PID controller design optimization for BLDC motor speed control system, *Przeglad Elektrotechniczny*, vol.96, no.11, pp.78-83, 2020.
[38] P. Khluabwannarat and D. Puangdownreong, Optimal fractional-order PID controller design for BLDC motor speed control system by using parallel flower pollination algorithm, *ICIC Express Letters*, vol.15, no.2, pp.165-172, 2021.
[39] M. M. Ali, C. Khompatraporn and Z. B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, *Journal of Global Optimization*, vol.31, pp.635-672, 2005.
[40] M. Jamil, X.-S. Yang and H-J. Zepernick, Test functions for global optimization: A comprehensive survey, *Swarm Intelligence and Bio-Inspired Computation Theory and Applications*, pp.193-222, 2013.
[41] M. Jamil and X.-S. Yang, A literature survey of benchmark functions for global optimization problems, *International Journal of Mathematical Modelling and Numerical Optimisation*, vol.4, no.2, pp.150-194, 2013.
[42] V. Plevris and G. Solorzano, A collection of 30 multidimensional functions for global optimization benchmarking, *Data*, vol.7, no.4(46), pp.1-51, 2022.

Author Biography

Supaporn Suwannarongsri received the B.Eng. degree in Industrial Engineering from Thonburi University (TRU), Thailand, 2004; the M.Eng. degree in Industrial Engineering from King Mongkut's Institute of Technology Ladkrabang (KMITL), Thailand, 2008; the Ph.D. degree in Sustainable Energy and Environment Technology and Management from Rajamangala University of Technology Rattanakosin (RMUTR), Thailand, 2014.

Dr. Suwannarongsri is currently an associate professor at the Department of Materials Handling and Logistics Engineering, Faculty of Engineering, King Mongkut's University of Technology North Bangkok (KMUTNB), Thailand. Her research interests include logistics and materials handling, operation research, production planning and design, and applications of metaheuristic algorithms to various real-world industrial engineering problems. She has published over 100 papers in journals and conferences, nationally and internationally.