Moth-flame glowworm swarm optimisation

Alboaneen, Dabiah Ahmed; Tianfield, Huaglory; Zhang, Yan

Published in:
Multiagent and Grid Systems

DOI:
10.3233/MGS-190314

Publication date:
2019

Document Version
Author accepted manuscript

Moth-Flame Glowworm Swarm Optimisation

Dabiah Ahmed Alboaneen, Huaglory Tianfield, and Yan Zhang

Abstract—One of the drawbacks of glowworm swarm optimisation (GSO) is its premature convergence, which leaves it often ineffective for solving complex practical problems. This paper proposes a new hybrid metaheuristic algorithm, that is, moth-flame glowworm swarm optimisation (MFGSO). The main idea of the hybrid algorithm is to combine the exploration ability in moth-flame optimisation (MFO) with the exploitation ability in GSO. Performance evaluations are conducted on benchmarking test functions in comparison with the basic GSO and other metaheuristic algorithms. The results show that MFGSO outperforms the basic GSO and other metaheuristic algorithms on most test functions in terms of local optima avoidance and convergence speed.

Index Terms—Glowworm swarm optimisation (GSO); metaheuristic; hybrid metaheuristic algorithm; moth-flame optimisation (MFO)

1 Introduction

Metaheuristic algorithms generally fall into two categories, namely (i) individual-based metaheuristics (IBM), which modify and improve a single candidate solution, e.g., simulated annealing (SA) [1], and (ii) population-based metaheuristics (PBM), which improve multiple candidate solutions and use population characteristics to guide the search, e.g., ant colony optimisation (ACO) [2], particle swarm optimisation (PSO) [3] and genetic algorithm (GA) [4]. Based on process strategies, PBM can be further classified into (i) PBM with reproductive strategies, which reproduce new solutions or generations, e.g., GA and (ii) PBM with non-reproductive strategies, e.g., biogeography-based optimisation (BBO) [5] [6].

In solving real-world problems, metaheuristics algorithms often trap in local minima, which could prevent an algorithm from moving toward the global solution. Recently, it has been shown that hybrid metaheuristic algorithms have the potential to overcome these challenges.

Glowworm swarm optimisation (GSO) algorithm was introduced by Krishnan and Ghose in 2006 [7]. It is inspired by the social behaviour of glowworm that a swarm of glowworms move through problem space and communicate with each other in order to determine a search direction. GSO algorithm is a population-based algorithm. As control is not centralised at a single point, thus it is more scalable. It has been successfully applied to solve many optimisation problems such as multimodal optimisation problems [7], clustering [8] [9], resources scheduling [10] [11]. However, GSO algorithm has some limitations that it is often unable to explore the space quickly and effectively.

On the other hand, moth-flame optimisation (MFO) was introduced by Mirjalili in 2015 [12] as a population-based metaheuristic algorithm. MFO algorithm has the ability to find competitive results compared with other metaheuristic algorithms such as GA and PSO. MFO's strengths are in two folds. Firstly, the exploration of MFO algorithm is strong, resulting in effective avoidance of local optima . Secondly, the exploitation is simple and effective in finding the near optimal solution when solving real problems [12].

There are two important mechanisms of metaheuristics algorithms, exploration and exploitation. Exploration refers to the process of searching new solution regions in the problem space, while exploitation refers to the process of searching in the neighbourhood of the so far found solutions. The primary principle of a metaheuristic algorithm is to efficiently balance the exploration and exploitation so as to find the global optimum quickly and efficiently. Hybridising metaheuristic algorithms is one way to balance the exploration and exploitation abilities. The aim of hybrid metaheuristic algorithms is to improve the convergence speed and to avoid local optima. Hybrid algorithms can be devised by combining one metaheuristic with another, e.g., chaos theory, levy flights strategy or genetic operators such as crossover and mutation.

In this paper we will propose a new hybrid metaheuristic algorithm, called moth-flame glowworm swarm optimisation (MFGSO), to improve the performance of the basic GSO. The fundamental idea of our proposed hybrid algorithm is that to update glowworm position according to the best position explored in MFO algorithm.

The remainder of this paper is arranged as follows. Section 2 presents literature review on existing hybrid metaheuristic algorithms. Section 3 presents our proposed hybrid algorithm MFGSO. Performance evaluations are conducted in section 4. Finally, Section 5 draws conclusion and sets out the future work.

2 Literature Review

There are several hybridisation methods for metaheuristic algorithms. Generally, algorithms may be hybridised in exploration phase, in exploitation phase, or to tune parameters of another algorithm.

2.1 Hybrid Metaheuristic Algorithms

PSO has good ability for exploitation. PSO is widely combined with other algorithms in order to improve the per-

- *D. Alboaneen is with Computer Department, College of Science and Humanities, Imam Abdulrahman Bin Faisal University, P.O.Box 31961, Jubail, Saudi Arabia.*
  E-mail: dabuainain@iau.edu.sa
- *H. Tianfield and Y. Zhang are with* School of Computing, Engineering *and Built Environment, Glasgow Caledonian University, Glasgow, United Kingdom.*
  E-mail: h.tianfield@gcu.ac.uk; yan.zhang@gcu.ac.uk

formance of algorithms. In [13], GA and PSO are combined to solve global optimisation of multimodal functions. The hybrid algorithm creates individuals in a new generation by crossover and mutation operations from GA and mechanisms of PSO. Another PSO hybrid with GA is proposed in [14] [15]. In [15], PSO is incorporated with mutation operator from GA to solve the stagnation problem and to prevent the particles from being trapped in local minima. PSO hybrid with differential evolution (DE) was proposed in [16], which is based on two populations. One population is enhanced by PSO and the other population is evolved by DE. The interplay between the two populations influences the balance between exploration and exploitation and can reduce the chance of being trapped in local sub-optima. PSO is combined with gravitational search algorithm (GSA) in [17], the ability of exploitation in PSO is combined with the ability of exploration in GSA to improve the convergence speed and avoid local optima. In addition, a binary version of this hybrid algorithm is also proposed in [18] to solve optimisation problems which have binary parameters. In [19], a new hybrid algorithm combined artificial fish swarm (AFS) algorithm and PSO. This algorithm has the advantages of both AFS and PSO. Exploring the problem space is with AFS while searching exact solution with PSO. Recently, PSO is combined with MFO in [20]. PSO is used for exploitation and MFO for exploration. Position and velocity of particle are updated based on moth and flame position in each iteration.

Harmony search has proved efficiency when using with different algorithms. A harmony search hybrid with firefly algorithm (FF) is proposed in [21], in which harmony search is used for the exploration and FF for the exploitation. Bat algorithm is added an adjustment operation with harmony search serving as a mutation operator during the process of the bat updating with the aim of speeding up convergence in [22]. The harmony search operator is added to the cuckoo updating in [23] so as to speed up convergence of cuckoo search algorithm and in [24] to improve the BBO algorithm.

GA is added hill-climbing to solve global problems in [25]. In [26], a hybrid optimisation algorithms of ACO and GA is proposed. GA is used for exploring the search space while ACO for exploitation.

In [27], a hybrid evolutionary FF is proposed. It combines the basic FF with the evolutionary operations of DE method to improve the searching accuracy and information sharing among the fireflies. Moreover, basic population-based incremental learning (PBIL) can exploit the solution quickly while it explores space poorly. Hence, a hybrid algorithm by integrating krill updating operator and probability updating operator together with krill herd method is proposed to solve optimisation problems in [28].

2.2 Hybrid GSO Algorithms

In basic GSO algorithm, the swarm is divided into subswarms in a dynamic manner and the movement of glowworms is dependent only on local information in each iteration. However, later in the search process, it is easy to fall into local optima, and difficult to jump out even on repeated iterations, which causes slow convergence, low precision and poor performance in high-dimensional problems. In order to address this issue, a number of hybrid GSO algorithms are proposed.

In the movement phase of basic GSO, each glowworm selects probabilistically a neighbour that glow brighter and moves a step that a fix size step multiplied by the distance between the neighbours. To improve the procedure of the movement in GSO, the movement formula in the basic GSO was replaced with formulas from other algorithms in [29]. The new movement formulas, inspired by artificial bee colony (ABC) and PSO algorithms, are proposed. It's showed that the proposed algorithm with PSO movement formula can find better solutions compared to basic GSO, proposed algorithm with ABC movement formula and other algorithms. However, the proposed algorithm only use a part from ABC and PSO which is the movement formula. In addition, the algorithm calls each formula when needed.

However, some researches aimed to overcome the local optimal trappings of GSO by adding local search. In [30], a hybrid algorithm of GSO and AFS, called GSO-FS, is proposed to overcome the local optimal trappings of GSO. In this algorithm, GSO implements local search and then AFS algorithm carries out exact solution and captures the global optimisation. It was demonstrated that the performance of GSO-FS algorithm is better than the basic GSO and AFS algorithms.

Yang *et al.* [31] proposed a hybrid artificial GSO (AGSO) algorithm for solving a system of non-linear equations in which the Hooke-Jeeves pattern search works as a local search operator. The AGSO algorithm showed high convergence and accuracy.

GSO with chaotic local search (CLS-GSO) is proposed in [32]. In CLS-GSO, chaos method as a local search operator is embedded into GSO, that is to say, during the course of each iteration, GSO implements the global search, then chaos method implements the local search. Simulation results showed that CLS-GSO outperformed GSO in terms of efficiency, precision, success rate of convergence and reliability.

Qu *et al.* [33] proposed a GSO algorithm hybrid with simplex search method for solving a system of non-linear equations. In this algorithm, simplex search method is applied as a local search operator, on those glowworms that have no neighbours. So, if a glowworm has neighbours, it moves to its one neighbour with a probability, if it has not neighbours, this shows it is local optimum. Its quality is better and can be used as the initial point of simplex search method. The hybrid algorithm improved the convergence rate, high accuracy, and robustness.

In [34], Ouyang *et al.* proposed a hybrid algorithm using the Broyde-Fletcher-Goldfarb-Shanno (BFGS) algorithm and GSO. BFGS is a quasi-newton method and is used as a local operator whereas GSO is used to search globally in the BFGS-GSO algorithm. It's showed that BFGS-GSO is feasible and effective in solving multi-extremum global optimisation.

Zhou *et al.* [35] proposed a hybrid GSO to solve constrained problems by embedding predatory behaviour of the AFS algorithm into GSO. This algorithm avoids the situation in which glowworms have empty neighbour sets. The glowworm whose neighbours set is empty, is only allowed to predatory in the area determined by the dynamic deci-

sion, then the result of the previous predatory determines the next state. Moreover, to escape from the local optimum, the local search strategy based on SA is applied to the best solution of the population of each generation. It's showed that the algorithm has faster convergence speed and high computational precision as compared to other algorithms.

PSO is combined with GSO in [36] to balance the diversity and convergence. In hybrid ensemble PSO-GSO algorithm (HEPGO), the PSO or GSO is called depending on a selection probability. And in the early stage, in order to ensure the convergence, PSO is called frequently. And after evolution finished, the whole swarm is combined using ensemble learning. The hybrid algorithm outperforms many versions of PSO.

2.3 Discussion

One significant finding of the literature review is that some of the existing optimisation algorithms suffer from premature convergence, lack in finding global optimised solution and struck into local optima. Moreover, no algorithm can provide optimal solution for all types of problems according to Wolpert and Macready's no free lunch (NFL) theorem [37]. This theorem encourages researchers to develop new algorithms or improve the existing ones.

Hybrid GSO algorithms in the literature improved the performance of the basic GSO by adding another algorithm as a local search such as AFS [30], Hooke-Jeeves [31], chaotic local search [32], simplex search [33], BFGS [34] and SA [35]. So, the GSO algorithm carries out the global initial search while the local search carries out the exact solution. However, GSO has some drawbacks. It highly depends on initial solutions. Initialisation process of GSO is random. Although random initialisation can guarantee the initial solutions distributed evenly in the solution space, the quality of solutions are unreliable, because some solutions apart from the global optimum. If the initial solutions are distributed evenly and also high-quality, it will contribute to the quality and efficiency of solutions, and prevent algorithm to be prematurely trapped in local optima in a certain extent.

To address this problem, this paper aims to improve the performance of GSO by combining the ability of exploration in MFO with the ability of exploitation in GSO, to benefit from both algorithms. MFO carries out the global search, while GSO carries out the local search in the satisfactory solution domains that are obtained by MFO, which makes the algorithm can obtain high-quality and evenly distributed initial solutions. MFGSO outperforms the basic GSO algorithm and most of metaheuristic algorithms on most test functions, which means that improving the exploration of GSO can give superior results in terms of local optima avoidance, convergence speed and stability.

However, in contrast to [29] [33] [35], MFGSO is not designed to improve the situation in which glowworms have no neighbour sets, which is the default situation of the basic GSO algorithm. In MFGSO, a random glowworm is generated when there is no neighbours.

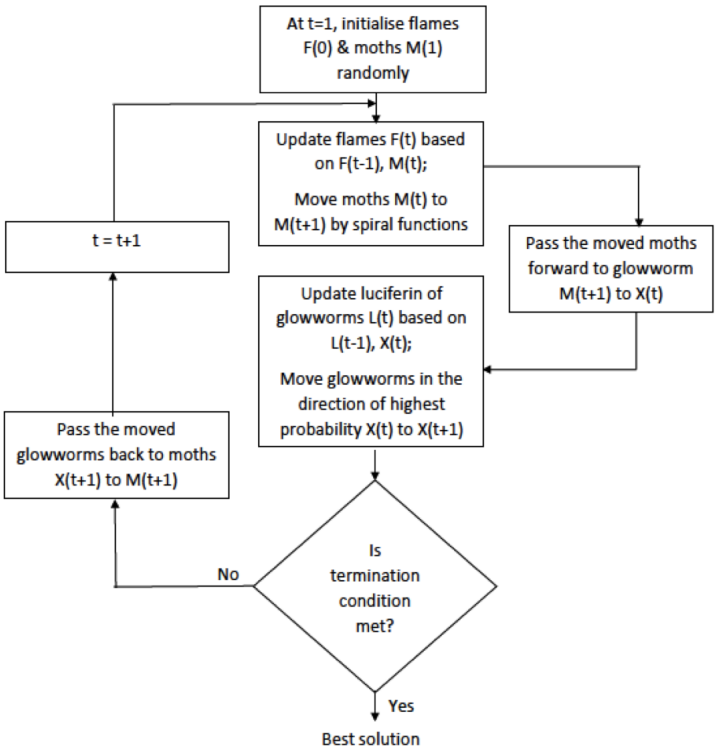


Fig. 1: Schematic of MFGSO.

3 Proposed Hybrid Metaheuristic Algorithm

Exploration is the ability of an algorithm to explore the search space whereas exploitation is the ability of the algorithm to exploit the best solution. The main aim of hybridising algorithms is to reduce the probability of being trapped in local optimum and speed up convergence, thus improving the performance of algorithm to solve a wide range of real-world problems. The basic GSO algorithm may get trapped into some local optima and it is often unable to explore the space efficiently and cannot perform global search efficiently as well. Moreover, the search by GSO depends on random moves, so a fast convergence cannot be guaranteed. In order to increase the diversity of the GSO population and therefore to avoid being trapped into local optima, MFO is incorporated as a mutation operator into the GSO with the aim of speeding up the convergence.

The basic idea of our proposed hybrid metaheuristic algorithm - moth-flame glowworm swarm optimisation (MFGSO) is to take advantage of the exploration and exploitation abilities of MFO and GSO. MFO is used for exploration as it uses logarithmic spiral function so it covers larger area in search space, whereas GSO is used for exploitation. The working principle of the proposed MFGSO is illustrated in Figure 1. Position of glowworms that is responsible for finding the optimum solution is replaced by the best position generated from MFO algorithm which is highly efficient to direct the glowworms faster toward

Algorithm 1: MFGSO: Moth-Flame Glowworm Swarm Optimisation

1 Set parameters β , λ, ζ, $max\#\_neighbour$, max_radius
2 Set the population size of glowworms and moths N
3 Set number of variables (i.e., problem's dimension) n
4 $\forall i = \{1, 2, ..., N\}$, set initial value $\ell^i(0)$, $radius^i(0)$
5 Set maximum number of iterations = T , set t = 1
6 **for** i = *1: N* **do**
7 **for** s =*1: n* **do**
8 $m_s^i(1) = (x_s^{upper} - x_s^{lower}) * rand() + x_s^{lower}$
9 $f_s^i(0) = (x_s^{upper} - x_s^{lower}) * rand() + x_s^{lower}$
10 **while** $t \leq T$ **do**
11 $N_F = round(N - t * \frac{N-1}{T})$
12 **for** i = *1 : N* **do**
13 Evaluate $fit(moth^i)$
14 Evaluate $fit(flame^i)$
15 $F(t) = sort(F(t-1), M(t))$ with the flames and moths fitness values from best to worst
16 $\phi = -1 + t * ((-1)/T)$
17 **for** i =*1:N* **do**
18 **for** j= *1 :*N_F **do**
19 $\nu = (\phi - 1) * rand + 1$
20 $D^i(t) = |F^i(t) - M^i(t)|$
21 **if** $i \leq N_F$ **then**
22 $M^i(t+1) = S(M^i(t), F^i(t)) = D^i(t) * e^{b\nu}.\cos(2\pi\nu) + F^i(t)$
23 **if** $i > N_F$ **then**
24 $M^i(t+1) = S(M^i(t), F^{N_F}(t)) = D^i(t) * e^{b\nu}.\cos(2\pi\nu) + F^{N_F}(t)$
25 $x^i(t) = M^i(t+1)$
26 **for** i = *1: N* **do**
27 $\ell^i(t) = (1-\lambda)\ell^i(t-1) + \gamma fit(x^i(t))$
28 **for** i = *1: N* **do**
29 $G^i(t) = \{j : ||x^j(t) - x^i(t)|| \leq radius^i(t)$ and $\ell^i \leq \ell^j(t)\}$
30 **for** *each* $j \epsilon G^i(t)$ **do**
31 $p_j^i(t) = \frac{\ell^j(t) - \ell^i(t)}{\sum_{j \epsilon G^i(t)} \ell_j(t) - \ell^i(t)}$
32 $j^{*i} = \arg\max_{j \epsilon G^i(t)} \{p_j^i(t)\}$
33 $x^i(t+1) = x^i(t) + \zeta\left(\frac{x^{j^{*i}}(t) - x^i(t)}{||x^{j^{*i}}(t) - x^i(t)||}\right)$
34 $radius^i(t+1) = \min\{max\_radius, \max\{0, radius^i(t) + \beta(max\#\_neighbour - |G^i(t)|)\}\}$
35 $M^i(t+1) = x^i(t+1)$
36 $t = t+1$
37 **return** Optimal Solution of MFGSO

optimal value. Hence, best features, namely exploration by MFO and exploitation by GSO are combined to obtain best possible optimal solution and also avoid being trapped in local optima.

The hybridisation of GSO with MFO is a high-level, relay, heterogeneous hybrid method. According to [38], the hybrid is high-level because the two metaheuristics are self-contained. It is relay in that one metaheuristic is applied after another, each using the output of the preceding as its input. It is heterogeneous in that the two algorithms for the hybrid are different.

The MFGSO algorithm can be formulated as in Algorithm 1. The process of MFGSO algorithm as follows:

Phase 1: Initialisation phase (lines $1-5$), initialise the population size N. The following initial values are set: (i) the initial luciferin $\ell^i(0)$, initial decision domains range $radius(0)$, sensor range maximum max_radius, number of neighbours $max\#\_neighbour$; (ii) number of the variables n, namely the dimension of the problem; (iii) step moving size ζ of glowworm; (iv) the maximum iterations T.

Phase 2: Exploration by MFO algorithm, the moths $i = 1, 2, ..., N$ and the flames $i = 1, 2, ..., N_F$ are initialised (lines $6-9$). Flames are sorted based on its fitness values and saved in F in case of first iteration. However, in next iteration, F is the sorted of merge moths and best flames from previous iteration (line 15). In the main loop, the number of flames will be decreased in each iteration (line 11). The fitness values of each moth and flame are obtained (lines $12-14$). After that, positions of moths are updated (lines $17-24$). The moth's current optimal position, $M^i(t+1)$ is obtained (line 25).

Phase 3: Exploitation by GSO algorithm (lines $26-35$), the current optimal position obtained from MFO, $x^i(t)$ and corresponding fitness value $fit(x^i(t))$ as the initial values of GSO algorithm are set.

Phase 4: Termination, when MFGSO algorithm reaches the maximum iterations, the optimal position searching by MFGSO algorithm is obtained and the algorithm exits (line 37). Otherwise, go to (line 11).

In MFO algorithm, the moths are considered as the candidate solutions and their position is considered as a vector of decision variables. Flames are the best positions of moths that are obtained so far by the moth. Therefore, each moth searches around a flame and updates it in case of finding a better solution.

In the initialisation phase of MFO algorithm, moths are randomly initialised. Also, flames are initialised randomly and then they are sorted based on its fitness function not based on the fittest moth and then saved in the flame matrix.

A moth represents a position in the problem's search space, that is for i-th moth,

$$moth^i = M^i = \begin{bmatrix} m_1^i \\ m_2^i \\ ... \\ m_n^i \end{bmatrix} \tag{1}$$

where n is the number of variables, namely the dimension of the problem.

In order to set the population of MFO algorithm, the set of moths M is represented in matrix.

$$M = \begin{bmatrix} M^1 & M^2 & ... & M^N \end{bmatrix} = \begin{bmatrix} m_1^1 & m_1^2 & ... & m_1^N \\ m_2^1 & m_2^2 & ... & m_2^N \\ ... & ... & ... & ... \\ m_n^1 & m_n^2 & ... & m_n^N \end{bmatrix} \tag{2}$$

where M is the position matrix of moths and N is the number of moths.

Likewise, a flame represents a (better) position in the problem's search space, that is for i-th flame,

$$flame^i = F^i = \begin{bmatrix} f_1^i \\ f_2^i \\ ... \\ f_n^i \end{bmatrix} \tag{3}$$

At the initialisation, since each moth flies around its corresponding flame, therefore, the flame matrix F is in the same size as the moth's matrix.

$$F = \begin{bmatrix} F^1 & F^2 & ... & F^{N_F} \end{bmatrix} = \begin{bmatrix} f_1^1 & f_1^2 & ... & f_1^{N_F} \\ f_2^1 & f_2^2 & ... & f_2^{N_F} \\ ... & ... & ... & ... \\ f_n^1 & f_n^2 & ... & f_n^{N_F} \end{bmatrix} \quad (4)$$

where F is the position matrix of flames and N_F is the number of flames.

At initialisation, a random population of moths N and flames N are generated (lines $6-9$) and the corresponding fitness values are calculated (lines $12-14$) as below.

The position of each individual moth in the population is initialised using:

$$m_s^i(0) = (x_s^{upper} - x_s^{lower}) * rand() + x_s^{lower}, \quad i = 1, 2, ..., N, s = 1, 2, ..., n \quad (5)$$

The position of each individual flame in the population is initialised using:

$$f_s^i(0) = (x_s^{upper} - x_s^{lower}) * rand() + x_s^{lower}, \quad i = 1, 2, ..., N, s = 1, 2, ..., n \quad (6)$$

where x_s^{upper} and x_s^{lower} are the upper and lower bounds of the variable x_s, respectively and $rand()$ is the random number generated with uniform distribution in the interval $[0, 1]$.

For all moths, assume there is an array for storing the corresponding objective function (fitness) value of the moths.

$$OM = \begin{bmatrix} OM^1 & OM^2 & ... & OM^N \end{bmatrix} \quad (7)$$

$$OM^i = fit(moth^i), i = 1, 2, ..., N \quad (8)$$

where fit denotes the fitness function.

Moreover, assume that there is an array for storing the corresponding fitness value of the flames.

$$OF = \begin{bmatrix} OF^1 & OF^2 & ... & OF^{N_F} \end{bmatrix} \quad (9)$$

$$OF^i = fit(flame^i), i = 1, 2, ..., N_F \quad (10)$$

Moths and flames are sorted with their fitness values in order from best to worst.

Each moth i is committed to update its position using only one of the flames. In each iteration and after updating the list of flames, the flames are sorted based on their best fitness values and moths (line 15).

$$F(t) = sort(F(t-1), M(t)) \quad (11)$$

The movements of moths and flames are the essential function that determines how the moths move around the search space. Moth's and flame's movements are iteratively updated until the termination condition is satisfied. For each iteration, update the position and fitness of moths and flames. Moths update their positions in hyper spheres around the best solutions obtained so far. The sequence of flames is changed based on the best solutions in each iteration, and the moths are required to update their positions with respect to the updated flames. The logarithmic spiral function is chosen as the main updating mechanism of the position of each moth with respect to the flame (lines $17-24$).

The flames are used to update the position of moths in two cases. Firstly, when the number of moths is lower than the number of flames (line 21), the position of moths is updated with respect to its corresponding flame (line 22). The first moth always updates its position with respect to the best flame, whereas the last moth updates its position with respect to the worst flame in the list.

However, when the number of moths is higher than the number of flames (line 23), the position of moths is updated with respect to the last flame (line 24) as below.

$$M^i(t+1) = \begin{cases} S(M^i(t), F^i(t)) = D^i(t) * e^{b\nu}.\cos(2\pi\nu) \\ +F^i(t), i = 1, 2, ..., N_F \\ S(M^i(t), F^{N_F}(t)) = D^i(t) * e^{b\nu}.\cos(2\pi\nu) \\ +F^{N_F}(t), i = N_F + 1, ..., N \end{cases} \quad (12)$$

where $M^i(t+1)$ is the updated position of moth $M^i(t)$, S is the spiral function, $M^i(t)$ denotes i^{th} moth, $F^i(t)$ denotes i^{th} flame after sorting flames based on its fitness from best to worst. b is a constant defining the shape of the logarithmic spiral. ν parameter defines how much the next position of the moth should be close to the flame, it is a random value in $[\phi, 1]$. where ϕ, is linearly decreasing from -1 to -2 over the course of iteration, called convergence constant. $D^i(t)$ is the vector of absolute difference between $F^i(t)$ and $M^i(t)$.

$$D^i(t) = \begin{bmatrix} d_1^i(t) \\ d_2^i(t) \\ ... \\ ... \\ d_n^i(t) \end{bmatrix} = |F^i(t) - M^i(t)| = \begin{bmatrix} |f_1^i(t) - m_1^i(t)| \\ |f_2^i(t) - m_2^i(t)| \\ ... \\ |f_n^i(t) - m_n^i(t)| \end{bmatrix} \quad (13)$$

$i = 1, 2, ..., N_F$. For $i > N_F, D^i(t) = |F^{N_F}(t) - M^i(t)|$.

where $|.|$ is the sign for absolute value.

Then, the moth $\{M^1, M^2, ..., M^N\}$ are sorted with their fitness values in order from best to worst as below.

$\{M^{s^1}, M^{s^2}, ..., M^{s^N}\}, s^1, s^2, ..., s^N \epsilon \{1, 2, ..., N\}$

For each i, if the updated moth $M^{s^i}(t+1)$ has better fitness value than the current flame $F^i(t)$, update the flame with the updated moth. Otherwise, the flame remains as before, that is,

$$F^i(t+1) = \begin{cases} M^{s^i}(t+1), if M^{s^i}(t+1) has better fit than F^i(t) \\ F^i(t), otherwise \end{cases} \quad (14)$$

The operations of $sort(M(t+1))$ and eq.14 are unified through eq.11.

To allow for much exploitation of the best promising solutions, the number of flames to be followed is adaptively decreased over iterations (line 11) as below.

$$N_F(t) = round(N - t * \frac{N-1}{T}) \quad (15)$$

where N_F is the number of flames, t indicates the index of search iterations, N is the maximum number of flames and T is the maximum number of iterations.

After moving the best positions of moths around the flames, the moved moths, from eq. 12, are passed forward as initial positions x^i of glowworms (line 25), that is, $x^i(t) = M^i(t+1), i = 1, 2, ..., N$ to exploit the solution space.

The set of glowworms is denoted as a matrix, i.e., $X = [x^1 x^2 ... x^N]$. Initially, each glowworm i where $i = 1, 2, ..., N$ has its own luciferin value. Each glowworm converts the objective function value $fit(x^i(t))$ at its current position $x^i(t)$ to a luciferin value $\ell^i(t)$ (lines $26 - 27$) as below.

$$\ell^i(t) = (1 - \lambda)\ell^i(t-1) + \gamma fit(x^i(t)) \tag{16}$$

where $\ell^i(t)$ is the luciferin value of glowworm at time t, λ is the luciferin decay coefficient ($0 < \lambda < 1$), γ is the luciferin enhancement coefficient, $fit(x^i(t))$ represents the value of the objective function at iteration t.

Therefore, the luciferin values of glowworms are updated according to the objective function values. A higher luciferin value means a better result. Also, the luciferin value decreases along the time to simulate the decay. Then, a glowworm chooses to move toward one of its neighbours j that has a higher luciferin value and within the local radial range $radius$. The set of glowworm's neighbours in the local radial range (line 29) can be written as below.

$$G^i(t) = \{j : ||x^j(t) - x^i(t)|| \leq radius^i(t) \; and \; \ell^i \leq \ell^j(t)\} \tag{17}$$

where $G^i(t)$ is the neighbour set, j is the index of glowworms close to i, $x^j(t)$ and $x^i(t)$ are positions of glowworms j and i, respectively, $\ell^i(t)$ and $\ell^j(t)$ are luciferin values for glowworms i and j, respectively. $||x||$ is the Euclidean norm of x, and $radius^i(t)$ represents the local radial range.

For each glowworm, the probability about the glowworm's movement direction toward the neighbour with a higher luciferin value (lines $30 - 31$) is calculated as below.

$$p_j^i(t) = \frac{\ell^j(t) - \ell^i(t)}{\sum_{j \epsilon G^i(t)} \ell_j(t) - \ell^i(t)} \tag{18}$$

where $p_j^i(t)$ is the probability of glowworm i moving toward glowworm j.

Glowworm i selects a glowworm j^{*i} from the neighbour set that has the highest probability of movement over others in the neighbour set (line 32).

$$j^{*i} = \arg \max_{j \epsilon G^i(t)} \{p_j^i(t)\} \tag{19}$$

where j^{*i} is the glowworm that has the maximum p_j^i.

The position of the glowworm is changed based on the position of the selected glowworm j^{*i} (line 33) as below.

$$x^i(t+1) = x^i(t) + \zeta \left(\frac{x^{j^{*i}}(t) - x^i(t)}{||x^{j^{*i}}(t) - x^i(t)||} \right) \tag{20}$$

where $x^i(t+1)$ and $x^i(t)$ are the new and current positions of glowworm i, respectively and ζ is the step size of moving.

Finally, the local radial range $radius^i$ is updated (line 34) using Eq. (21) in order to form the set of neighbours.

$$radius^i(t+1) = \min\{max\_radius, \max\{0, radius^i(t) + \beta(max\#\_neighbour - |G^i(t)|)\}\} \tag{21}$$

where max_radius is the maximum range, β is the change rate of the neighbourhood range, $max\#\_neighbour$ is a parameter used to control the number of neighbours and $|G^i(t)|$ is cardinality of the set, i.e., the actual number of neighbours.

After moving the glowworms, the moved glowworms, from eq.20, are passed back to the moths for the next iteration (line 35), that is, $M^i(t+1) = x^i(t+1), i = 1, 2, ..., N$.

When the maximum number of iterations reached, the best solution of MFGSO algorithm is returned (line 37). Otherwise, return to initialise MFO algorithm (line 11).

4 Performance Evaluations

Experiments are conducted to compare the proposed hybrid algorithm MFGSO to the basic GSO, MFO and four other metaheuristic algorithms namely, GA [4], DE [39], multiverse optimiser (MVO) [40] and FF [41].

GA and DE are the most popular algorithms. MVO is one of the recent algorithms developed in 2016. FF follows a similar logic as GSO with some variations. The similarities are that the attractiveness of each firefly is proportional to its brightness and the brightness of the glow of each firefly is determined by the landscape of the objective function. However, the FF algorithm has two variations. In GSO, the attractiveness of each glowworm is constant within a fixed sensor range and zero beyond the range, while the attractiveness of each firefly in FF exponentially decays with distance. The second variation is the addition of a small randomisation into the movement update of each firefly.

The maximum iteration times is set as $T = 1000$. Empirically we have found that with this number of iterations all implemented algorithms can converge for all the test functions selected in this paper.

Most metaheuristic algorithms have some parameters that can affect their performance. In general, the problem of optimising these parameters is outside the scope of this paper. However, in the experiments we follow the recommended value of parameters for all algorithm as presented in Table 1 [4] [39] [41] [40] [7] .

To test the performance of the proposed hybrid algorithm MFGSO, a set of 20 test functions is used. These test functions are generally used to evaluate the performance of algorithms. Table 2 presents formulation, properties, dimension n, search space and the global optimum f_k^{min} value for each test function [42]. According to their properties, the test functions are divided into two groups: unimodal functions (f_1 - f_6) and multimodal functions (f_7 - f_{20}). A function is considered as unimodal if it has only one optimum solution, while the multimodal functions have two or more local optimum solutions. All test functions are minimisation problems.

For the performance evaluations, all algorithms are implemented in Python. We implement MFO, MVO and FF algorithms from EvoloPy open source [43]. Each experiment

TABLE 1: Parameter settings of the algorithms

Algorithm	Notation	Description of the parameter	Value
-	R	Number of experimental runs	20
-	N	Number of population	250
-	T	Number of iterations	1000
GA	p_c	Crossover probability	0.8
	p_m	Mutation probability	0.2
	-	Selection mechanism	Roulette wheel
DE	p_c	Crossover probability	0.9
	w	Differential weight	0.5
FF	α	Alpha	0.5
	β	Beta	0.20
	γ	Gamma	1
MVO	min	Minimum wormhole existence probability	0.2
	max	Maximum wormhole existence probability	1
MFO	b	Constant defining the shape of the logarithmic spiral	1
GSO	λ	Luciferin decay coefficient	0.4
	γ	Luciferin enhancement coefficient	0.6
	β	Rate of the neighbourhood range	0.08
	$max\#\_neighbour$	No. of neighbours	5
	max_radius	Maximum range	8
	ζ	Step size of moving	0.03
	ℓ	Initial luciferin	0.05

TABLE 2: Test functions

Test function ($f_k(x)$)	Property	Dimension (n)	Search Space	Global Optimum (f_k^{min})
$f_1(x) = \sum_{i=1}^{n} x_i^2$	Unimodal	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	Unimodal	30	$[-100, 100]^n$	0
$f_3(x) = \max_i \{\|x_i\|, 1 \leq i \leq n\}$	Unimodal	30	$[-100, 100]^n$	0
$f_4(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	Unimodal	30	$[-30, 30]^n$	0
$f_5(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	Unimodal	30	$[-100, 100]^n$	0
$f_6(x) = \sum_{i=1}^{n} ix^4 + rand(0, 1)$	Unimodal	30	$[-1.28, 1.28]^n$	0
$f_7(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	Multimodal	30	$[-5.12, 5.12]^n$	0
$f_8(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	Multimodal	30	$[-32, 32]^n$	0
$f_9(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	Multimodal	30	$[-600, 600]^n$	0
$f_{10}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	Multimodal	30	$[-50, 50]^n$	0
$f_{11}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Multimodal	30	$[-50, 50]^n$	0
$f_{12}(x) = (\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-1}$	Multimodal	2	$[-65.536, 65.536]^n$	1
$f_{13}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Multimodal	2	$[-5, 5]^n$	-1.0316
$f_{14}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	Multimodal	2	$[-5, 10] * [0, 15]$	0.398
$f_{15}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	Multimodal	2	$[-2, 2]^n$	3
$f_{16}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	Multimodal	3	$[0, 1]^n$	-3.86
$f_{17}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	Multimodal	6	$[0, 1]^n$	-3.32
$f_{18}(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	Multimodal	4	$[0, 10]^n$	-10.1532
$f_{19}(x) = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	Multimodal	4	$[0, 10]^n$	-10.4028
$f_{20}(x) = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	Multimodal	4	$[0, 10]^n$	-10.5363

is repeated for R times so as to minimise the influence of random effects and to ensure that the results are statistically acceptable.

For each test function, we calculate the averaged fitness values (average best function values found in the last iteration) over the experimental results from the R runs (AVG), the standard deviation (STD) and the best (Best) value, which is the minimum value that algorithm has reached in R runs.

To plot the convergence curve, we record $f_k(x)$ in the full process of search iterations and over the R runs of experiments, i.e., on each test function,

$$\overline{f_k(x)^{<t>}} = \frac{1}{R} * \sum_{r=1}^{R} f_k(x)^{<r,t>} \tag{22}$$

where r is index of the experimental runs , k is the index of test functions $f_k(x), k = 1, 2, ..., 20$, and t is index of search iterations $t = 1, ..., T$.

Otherwise, for comparisons amongst the algorithms, we are only interested in the $f_k(x)$ at the end of search iterations, i.e., $t = T$. On each function, over the R runs of experiments, we have

$$AVG_k^{<T>} = \overline{f_k(x)^{<T>}} = \frac{1}{R} * \sum_{r=1}^{R} f_k(x)^{<r,T>} \tag{23}$$

$$STD_k^{<T>} = \sqrt{\frac{\sum_{r=1}^{R}(f_k(x)^{<r,T>} - \overline{f_k(x)^{<T>}})^2}{R - 1}} \tag{24}$$

$$Best_k^{<T>} = \min_{r=1,\ldots,R} \{f_k(x)^{<r,T>}\} \quad (25)$$

The AVG_k and $Best_k$ values represent the global convergence of the algorithm. However, considering the AVG_k and $Best_k$ values only is not enough because two algorithms can have equal averages, but have different performance in terms of finding the global optimum in each run. Therefore, STD is adopted to determine how the results are closed to each other and reflect the spread. The smaller the STD, the lower the dispersion of results.

The results of unimodal test functions are presented in Table 3. The unimodal test functions are useful to examine exploitation of the algorithms. For unimodal functions (f_1-f_6), MFGSO obtains better results than basic GSO on five out of six test functions (f_1-f_5). Unimodal functions are suitable for testing the convergence because there is no local solution. These results prove that MFGSO improves the convergence rate of GSO on five of the six unimodal functions.

The rest of functions, (f_7-f_{20}), are multimodal, which have multiple local solutions. Tables 4 and 5 show that the MFGSO algorithm outperforms GSO on all the multimodal test functions except f_{17} and obtains similar results in function f_{16}. This demonstrates that the avoidance of local optima has been significantly improved by MFGSO. In addition, the results can also evidence high exploration of MFGSO.

For the average best of 20 runs, MFGSO reaches global minima on nine test functions, i.e., f_{12} - f_{20}. These functions share the same proprieties, i.e., they are multimodal and low-dimensional functions.

Statistically speaking, for the best average over 20 runs on 20 test functions, GA has the best results on 11 functions (f_1-f_{11}). MFO is the best on f_{19} and f_{20} and has similar results with other algorithms on 5 functions (f_{12} - f_{16}). MFGSO is the best on f_{18} function. In addition, MFGSO has similar results with other algorithms on 5 functions as well (f_{12}-f_{16}). Lastly, FF has the worst performance among all algorithms except the basic GSO. However, FF reaches the same values on f_{12}, f_{14} and f_{16} with other algorithms.

Figures 2(a)-2(t) and 3(a)-3(t) depict the convergence curves of all algorithms on all test functions (f_1-f_{20}) over t = 1-150 and t = 1-1000, respectively. In the convergence plots, x axis represents the number of iterations and y axis represents the fitness values averaged over 20 runs. Here, the convergence of MFGSO is competitively faster than the basic GSO algorithm on all functions except f_6 and f_{17}. On f_{13}, f_{15} and f_{16} the convergence curves of the basic GSO are close to those of MFGSO.

In addition, Figures 4(a)-4(t) depict the box-plots on all test functions employed using all algorithms. The box-plots are used to analyse the variability in getting fitness values averaged of best function values in 20 runs obtained by each algorithm. In this plot, the box relates to the interquartile range, the whiskers represent the maximum and minimum fitness values and the bar in the box represents the median value. The box-plots show that MFGSO algorithm performs well for minimising these test functions.

To further evaluate the performance of the metaheuristic algorithms, the non-parametric statistical test, Wilcoxon's rank sum test is performed using the results of MFGSO against all algorithms at 5% significance level. Table 6 lists the p-values obtained by the test using GraphPad Prism software between the MFGSO and the other algorithms in each function. If p-values less than 0.05 that means the null hypothesis is rejected which proves the significant difference between algorithms at a level of 5%. The advantage of the Wilcoxon rank sum test, compared to other tests like the t-test, is that it is more robust to outliers.

The p-values in Table 6 confirm that MFGSO is significant different from the basic GSO on all test functions. Moreover, it is different from FF, MVO, GA and DE on most test functions. However, there is no significantly difference between MFGSO and MFO algorithms on most test functions.

5 Conclusion and Future Work

Existing hybrid GSO algorithms focus on improving the basic GSO by combining it with other methods to carry out the local search, while the random initialisation process of GSO makes the quality of solutions unreliable, because a part of solutions are apart from the global optimum. To address the problem, this paper has proposed a hybrid algorithm of GSO with MFO to improve the performance of basic GSO algorithm. The main idea of MFGSO is to balance between local optima avoidance and convergence speed of the algorithm. In the proposed hybrid algorithm MFGSO, MFO carries out the global search, while GSO carries out the local search in the satisfactory solution domains that are obtained by MFO, which enables the hybrid algorithm to obtain high-quality and evenly distributed initial solutions.

The MFGSO algorithm has been compared with the basic GSO and other metaheuristic algorithms on 20 test functions in terms of local optima avoidance and convergence speed. The results have showed that MFGSO outperforms the basic GSO algorithm and some of metaheuristic algorithms on most test functions.

One future work will be improving the MFGSO to avoid the situation in which glowworms have no neighbour sets. Another will be applying MFGSO to solve real-world problems.

References

[1] S. Kirkpatrick, M. P. Vecchi, *et al.*, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Trans. on*, vol. 1, no. 1, pp. 53–66, 1997.

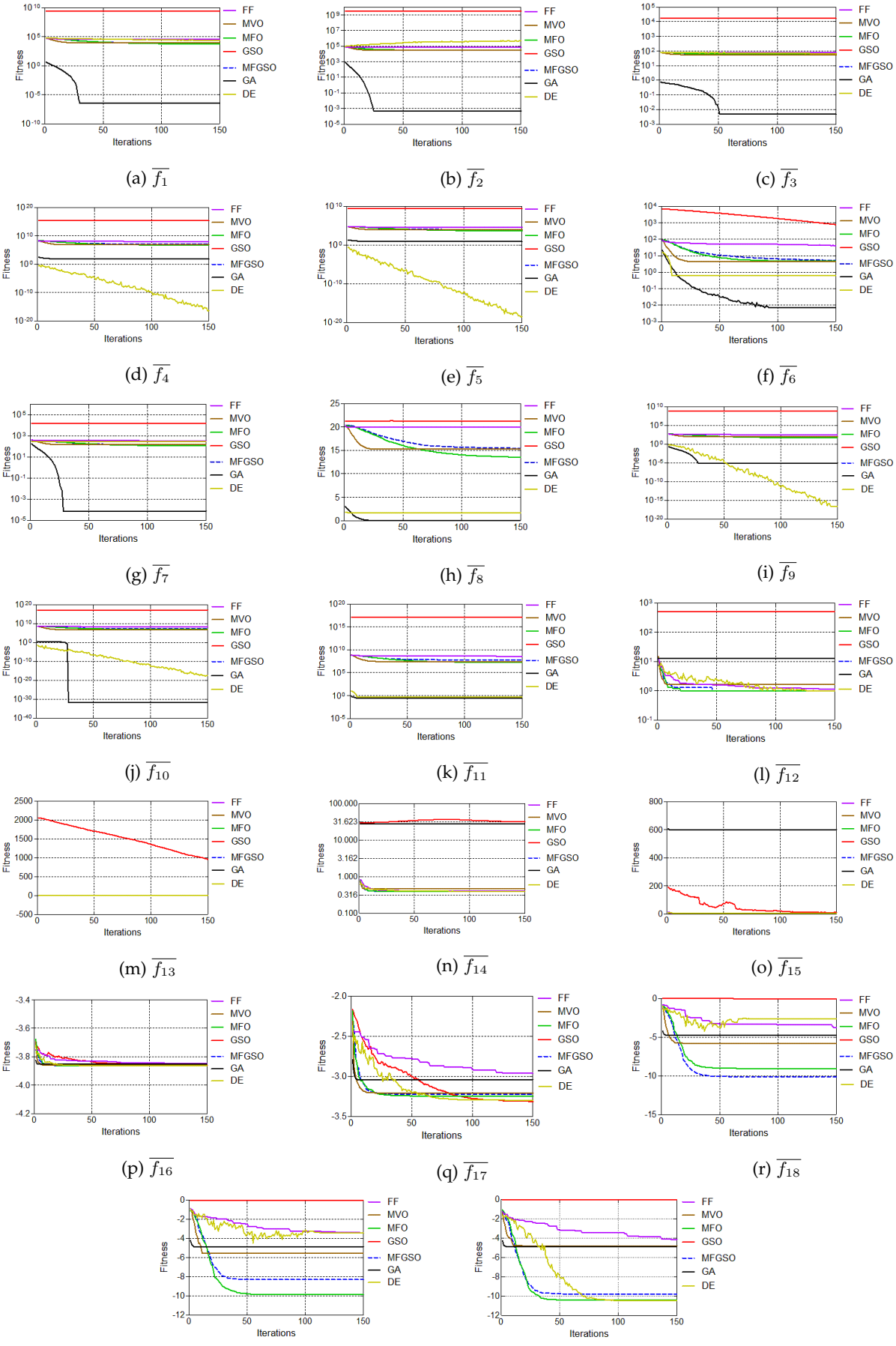
[3] J. Kenndy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE Int. Conf. on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[4] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Comp.*, vol. 2, no. 2, pp. 88–105, 1973.

[5] D. Simon, "Biogeography-based optimization," *Evolutionary Computation, IEEE Trans. on*, vol. 12, no. 6, pp. 702–713, 2008.

[6] D. Alboaneen, H. Tianfield, and Y. Zhang, "Metaheuristic approaches to virtual machine placement in cloud computing: A review," in *Proc. of Int. Conf. on Cloud Comp. and Big Data*, pp. 1–8, 2016.

[7] K. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications," *Multiagent and Grid Sys.*, vol. 2, no. 3, pp. 209–222, 2006.



(a) $\overline{f_1}$ (b) $\overline{f_2}$ (c) $\overline{f_3}$

(d) $\overline{f_4}$ (e) $\overline{f_5}$ (f) $\overline{f_6}$

(g) $\overline{f_7}$ (h) $\overline{f_8}$ (i) $\overline{f_9}$

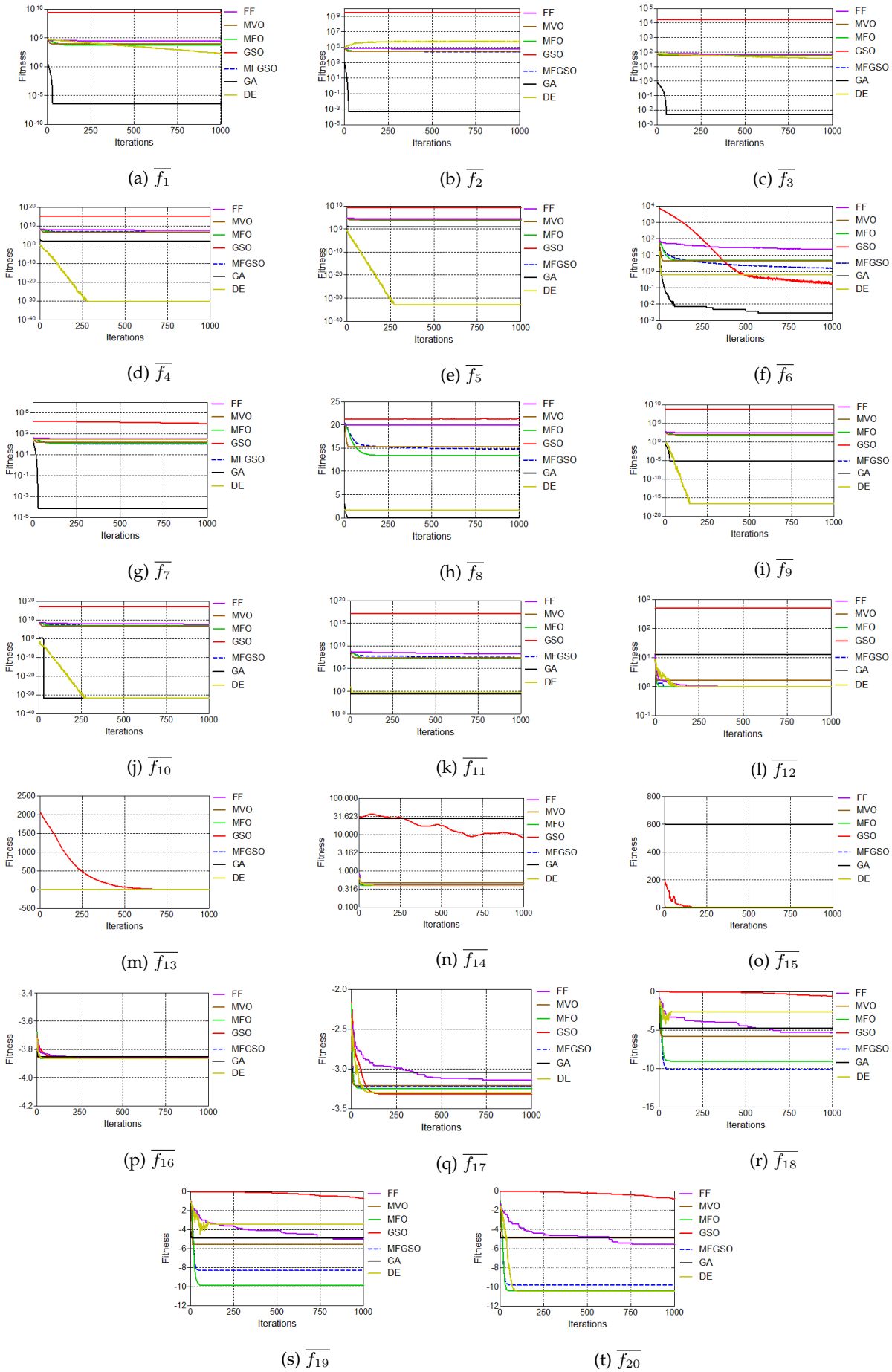
(j) $\overline{f_{10}}$ (k) $\overline{f_{11}}$ (l) $\overline{f_{12}}$

(m) $\overline{f_{13}}$ (n) $\overline{f_{14}}$ (o) $\overline{f_{15}}$

(p) $\overline{f_{16}}$ (q) $\overline{f_{17}}$ (r) $\overline{f_{18}}$

(s) $\overline{f_{19}}$ (t) $\overline{f_{20}}$

Fig. 2: Convergence curves of functions (f_1-f_{20}) over $t = 0 - 150$



(a) $\overline{f_1}$ (b) $\overline{f_2}$ (c) $\overline{f_3}$

(d) $\overline{f_4}$ (e) $\overline{f_5}$ (f) $\overline{f_6}$

(g) $\overline{f_7}$ (h) $\overline{f_8}$ (i) $\overline{f_9}$

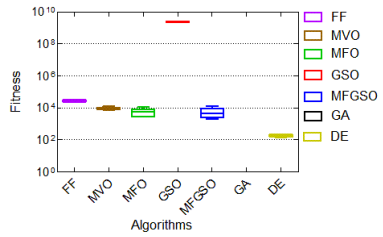
(j) $\overline{f_{10}}$ (k) $\overline{f_{11}}$ (l) $\overline{f_{12}}$

(m) $\overline{f_{13}}$ (n) $\overline{f_{14}}$ (o) $\overline{f_{15}}$

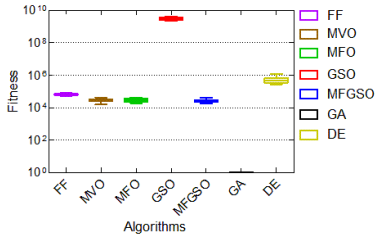
(p) $\overline{f_{16}}$ (q) $\overline{f_{17}}$ (r) $\overline{f_{18}}$

(s) $\overline{f_{19}}$ (t) $\overline{f_{20}}$

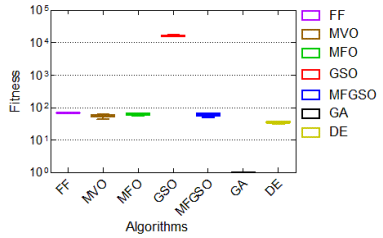
Fig. 3: Convergence curves of functions (f_1-f_{20}) over $t = 0 - 1000$



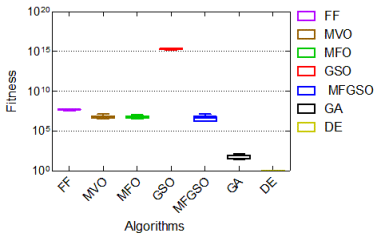
(a) f_1



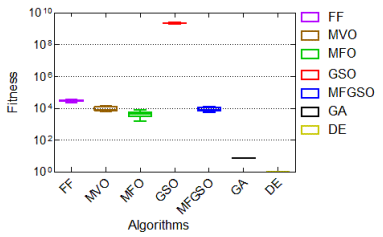
(b) f_2



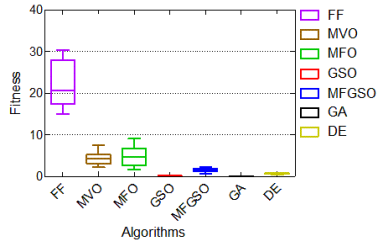
(c) f_3



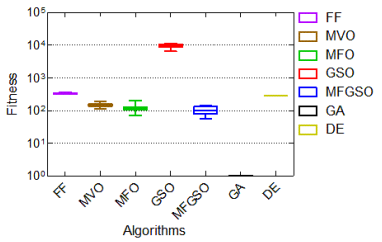
(d) f_4



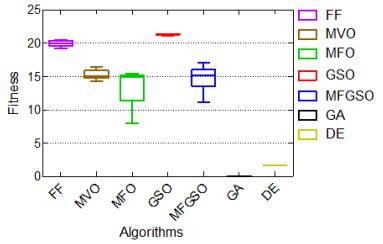
(e) f_5



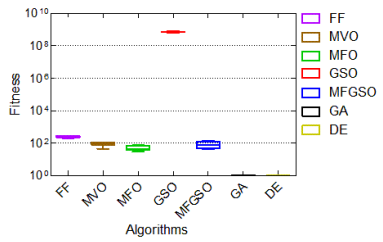
(f) f_6



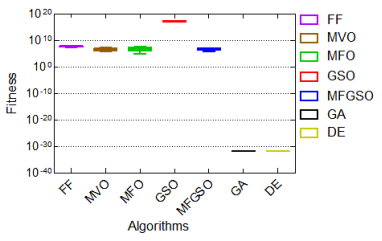
(g) f_7



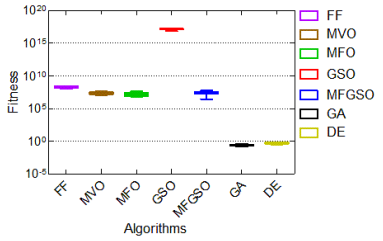
(h) f_8



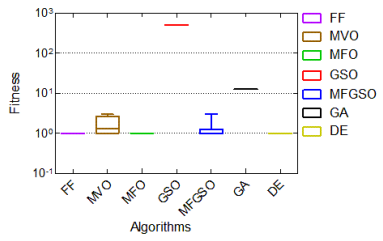
(i) f_9



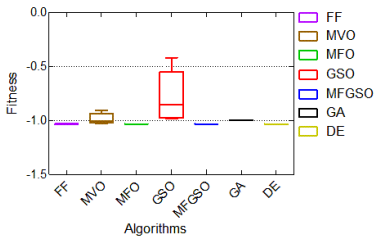
(j) f_{10}



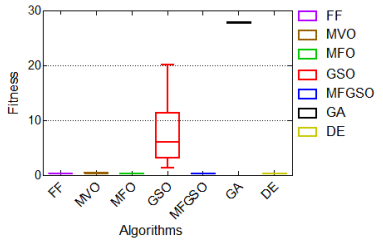
(k) f_{11}



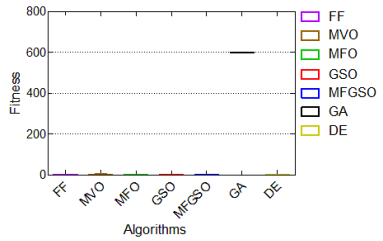
(l) f_{12}



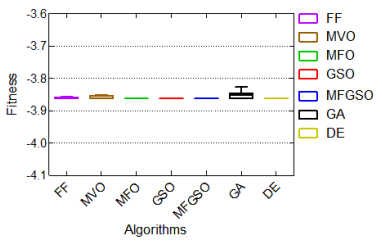
(m) f_{13}



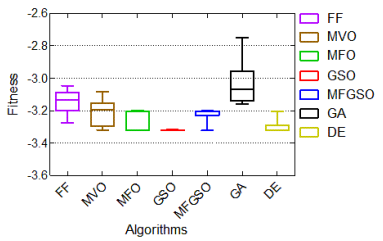
(n) f_{14}



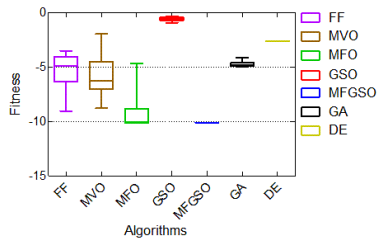
(o) f_{15}



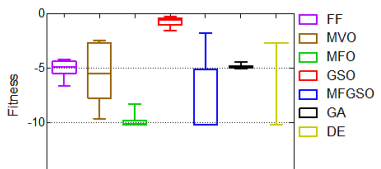
(p) f_{16}



(q) f_{17}



(r) f_{18}



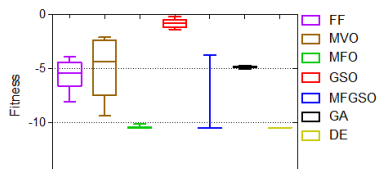


TABLE 3: Results on unimodal functions (f_1 - f_6) at the end of search iterations, i.e., $t = T$ (Best values are framed)

Test function	Measure	GA	DE	FF	MVO	MFO	GSO	MFGSO
f_1	AVG	0	188.8123	2.79E+04	9.46E+03	6.03E+03	2.39E+09	6.32E+03
	STD	0	28.3091	4.01E+03	1.72E+03	2.88E+03	1.54E+08	4.13E+03
	Best	0	150.0612	2.25E+04	7.40E+03	2.83E+03	2.17E+09	2.12E+03
f_2	AVG	0	5.46E+05	6.45E+04	2.99E+04	3.06E+04	3.15E+09	2.60E+04
	STD	0	2.71E+05	8.68E+03	7.63E+03	7.42E+03	6.27E+08	5.85E+03
	Best	0	2.76E+05	5.19E+04	1.60E+04	1.92E+04	2.30E+09	1.79E+04
f_3	AVG	0	35.7160	69.80	54.67	63.37	1.66E+04	60.94
	STD	0	2.3798	1.8558	5.4774	4.09	4.84E+02	5.8621
	Best	0	31.5692	66.13	44.45	56.83	1.59E+04	49.90
f_4	AVG	59.37	0	5.11E+07	6.55E+06	5.89E+06	2.24E+15	5.08E+06
	STD	39.60	0	9.25E+06	3.08E+06	2.49E+06	4.0349E+14	3.57E+06
	Best	28.58	0	3.37E+07	3.34E+06	3.51E+06	1.54E+15	1.67E+06
f_5	AVG	7.5	0	3.02E+04	9.62E+03	4.47E+03	2.31E+09	9.27E+03
	STD	0	0	4.29E+03	2.70E+03	1.88E+03	1.72E+08	2.62E+03
	Best	7.5	0	2.33E+04	6.35E+03	1.61E+03	2.01E+09	5.85E+03
f_6	AVG	0.00287	0.6433	22.09	4.35	4.72	0.1716	1.57
	STD	0.0040	0.0141	5.5212	1.6010	2.43	0.0615	0.47
	Best	0.0006	0.6433	15.08	2.20	1.76	0.0935	0.72

TABLE 4: Results on multimodal functions (f_7 - f_{13}) at the end of search iterations, i.e., $t = T$ (Best values are framed)

Test function	Measure	GA	DE	FF	MVO	MFO	GSO	MFGSO
f_7	AVG	0	290	332.94	147.47	119.40	9.37E+03	101.86
	STD	0	0	11.8354	21.4647	32.78	1.36E+03	30.0280
	Best	0	290	316.40	113.90	72.44	6.61E+03	56.45
f_8	AVG	4.44E-16	1.6844	19.97	15.30	13.45	21.34	14.78
	STD	0	0	0.4201	0.7299	2.52	0.0768	1.8578
	Best	4.44E-16	1.6844	19.26	14.36	7.98	21.18	11.14
f_9	AVG	0	0	255.70	89.56	49.75	7.25E+08	84.14
	STD	0	0	29.4141	21.7526	16.05	6.04E+07	34.8975
	Best	0	0	203.26	43.88	29.82	6.10E+08	43.89
f_{10}	AVG	1.57E-32	1.57E-32	5.61E+07	6.11E+06	1.21E+07	1.43E+17	6.05E+06
	STD	2.88E-48	2.88E-48	2.11E+07	6.57E+06	1.35E+07	2.32899E+16	4.41E+06
	Best	1.57E-32	1.57E-32	2.67E+07	8.64E+05	8.84E+04	1.05E+17	7.34E+05
f_{11}	AVG	2.64E-01	0.5468	1.89E+08	2.62E+07	1.93E+07	1.42E+17	2.77E+07
	STD	6.17E-02	3.24E-01	6.38E+07	1.19E+07	1.22E+07	3.26353E+16	1.54E+07
	Best	1.75E-01	0.3178	9.99E+07	1.04E+07	7.14E+06	7.41E+16	2.92E+06
f_{12}	AVG	12.67	1	1	1.6838	1	500	1
	STD	0	0	0	0.8252	0	0	0
	Best	12.67	1	1	1	1	500	1
f_{13}	AVG	-0.9992	-1.0316	-1.0313	-0.9868	-1.0316	-0.7764	-1.0316
	STD	0.0008	0	0.0003	0.0479	0	0.2220	0
	Best	-1.0000	-1.0316	-1.0316	-1.0304	-1.0316	-0.9856	-1.0316

[8] Z. Huang and Y. Zhou, "Using glowworm swarm optimization algorithm for clustering analysis," *Journal of Convergence Information Technology*, vol. 6, no. 2, pp. 78–85, 2011.

[9] I. Aljarah and S. A. Ludwig, "A new clustering approach based on glowworm swarm optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2642–2649, IEEE, 2013.

[10] D. Alboaneen, H. Tianfield, and Y. Zhang, "Glowworm swarm optimisation algorithm for virtual machine placement in cloud computing," in *Ubiquitous Intelligence & Comp., Advanced and Trusted Comp., Scalable Comp. and Communications, Cloud and Big Data Comp., Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Intl IEEE Conf.*, pp. 808–814, IEEE, 2016.

[11] D. A. Alboaneen, H. Tianfield, and Y. Zhang, "Glowworm swarm optimisation based task scheduling for cloud computing," in *Proceedings of the Second International Conference on Internet of Things and Cloud Computing*, ICC '17, (New York, NY, USA), pp. 152:1–152:7, ACM, 2017.

[12] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[13] Y.-T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimization for multimodal functions," *Applied Soft Computing*, vol. 8, no. 2, pp. 849–857, 2008.

[14] X. Lai and M. Zhang, "An efficient ensemble of ga and pso for real function optimization," in *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 651–655, IEEE, 2009.

[15] A. A. A. Esmin, G. Lambert-Torres, and G. B. Alvarenga, "Hybrid evolutionary algorithm based on pso and ga mutation," in *Hybrid Intelligent Systems, 2006. HIS'06. Sixth International Conference on*, pp. 57–57, IEEE, 2006.

[16] B. Niu and L. Li, "A novel pso-de-based hybrid algorithm for global optimization," *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*, pp. 156–163, 2008.

[17] S. Mirjalili and S. Z. M. Hashim, "A new hybrid psogsa algorithm for function optimization," in *Computer and information application (ICCIA), 2010 international conference on*, pp. 374–377, IEEE, 2010.

TABLE 5: Results on multimodal functions (f_{14} - f_{20}) at the end of search iterations, i.e., $t = T$ (Best values are framed)

Test function	Measure	GA	DE	FF	MVO	MFO	GSO	MFGSO
f_{14}	AVG	27.85	0.398	0.398	0.4622	0.398	7.937	0.398
	STD	0.08	0	0.0005	0.0547	0	6.2342	0
	Best	27.76	0.398	0.398	0.4120	0.398	1.447	0.398
f_{15}	AVG	600	3	3.01	3.52	3	3.0153	3
	STD	0	0	0.0110	0.4918	0	0.0101	0
	Best	600	3	3	3.05	3	3.0033	3
f_{16}	AVG	-3.85	-3.86	-3.86	-3.86	-3.86	-3.86	-3.86
	STD	0.0113	0	0.0011	0.0043	0	0.0004	0
	Best	–3.86	-3.86	-3.86	-3.86	-3.86	-3.86	-3.86
f_{17}	AVG	-3.04	-3.30	-3.14	-3.21	-3.25	-3.32	-3.28
	STD	0.1266	0.05	0.0756	0.0764	0.0615	0.0011	0.0618
	Best	-3.16	-3.32	-3.27	-3.32	-3.32	-3.32	-3.32
f_{18}	AVG	-4.7376	-2.6338	-5.3502	-5.8114	-9.0409	-0.5854	-10.1037
	STD	0.2420	0	1.6392	1.9189	2.2118	0.1885	0.0009
	Best	-4.9803	-2.6338	-9.0538	-8.7952	-10.1040	-0.9416	-10.1042
f_{19}	AVG	-4.8777	-3.4627	-5.0321	-5.5553	-9.8891	-0.7337	-8.3172
	STD	0.1744	2.3568	0.7459	2.6213	0.5856	0.3938	3.1096
	Best	-5.0381	-10.1703	-6.6601	-9.6601	-10.1694	-1.5472	-10.1702
f_{20}	AVG	-4.8851	-10.4833	-5.5847	-4.8124	-10.4223	-0.8008	-9.8137
	STD	0.0943	0	1.3553	2.5720	0.1018	0.3814	2.1160
	Best	-5.0788	-10.4833	-8.0900	-9.3691	-10.4828	-1.3821	-10.4833

TABLE 6: P-values of the Wilcoxon test of MFGSO results versus other algorithms (framed where p $\geq$ 0.05)

Test function	MFGSO vs FF	MFGSO vs MVO	MFGSO vs MFO	MFGSO vs GSO	MFGSO vs GA	MFGSO vs DE
f_1	0.0020	0.0371	0.9219	0.0020	0.0020	0.0020
f_2	0.0020	0.3223	0.1934	0.0020	0.0020	0.0020
f_3	0.0020	0.1055	0.2324	0.0020	0.0020	0.0020
f_4	0.0020	0.4316	0.5566	0.0059	0.0020	0.0020
f_5	0.0020	0.8457	0.0020	0.0020	0.0020	0.0020
f_6	0.0020	0.0020	0.0020	0.0020	0.0020	0.0039
f_7	0.0020	0.0098	0.4922	0.0020	0.0020	0.0020
f_8	0.0020	0.5566	0.1055	0.0020	0.0020	0.0020
f_9	0.0020	0.8457	0.0371	0.0020	0.0020	0.0020
f_{10}	0.0020	0.9219	0.2324	0.0020	0.0020	0.0020
f_{11}	0.0020	0.8457	0.2324	0.0020	0.0020	0.0020
f_{12}	0.5703	0.0371	0.5000	0.0036	0.0036	0.5000
f_{13}	0.0020	0.0020	1	0.0020	0.0020	1
f_{14}	0.0020	0.0020	0.5000	0.0020	0.0020	1
f_{15}	0.0020	0.0020	0.7500	0.0020	0.0027	0.5000
f_{16}	0.0020	0.0020	1	0.0020	0.0020	0.1250
f_{17}	0.0273	0.3223	0.4316	0.0039	0.0020	0.0020
f_{18}	0.0020	0.0020	0.0039	0.0020	0.0020	0.0020
f_{19}	0.0273	0.0840	0.7695	0.0020	0.0098	0.0098
f_{20}	0.0039	0.0098	0.0840	0.0020	0.0039	0.0020

[18] S. Mirjalili, G.-G. Wang, and L. d. S. Coelho, "Binary optimization using hybrid particle swarm optimization and gravitational search algorithm," *Neural Computing and Applications*, vol. 25, no. 6, pp. 1423–1435, 2014.

[19] X.-G. Yao, Y.-Q. Zhou, and Y.-M. Li, "Hybrid algorithm with artificial fish swarm algorithm and pso," *Application Research of Computers*, vol. 6, p. 027, 2010.

[20] R. Bhesdadiya, I. N. Trivedi, P. Jangir, A. Kumar, N. Jangir, and R. Totlani, "A novel hybrid approach particle swarm optimizer with moth-flame optimizer algorithm," in *Advances in Computer and Computational Sciences*, pp. 569–577, Springer, 2017.

[21] L. Guo, G.-G. Wang, H. Wang, and D. Wang, "An effective hybrid firefly algorithm with harmony search for global numerical optimization," *The Scientific World Journal*, 2013.

[22] G.-G. Wang and L. Guo, "A novel hybrid bat algorithm with harmony search for global numerical optimization," *Journal of Applied Mathematics*, vol. 2013, 2013.

[23] G.-G. Wang, A. H. Gandomi, X. Zhao, and H. C. E. Chu, "Hybridizing harmony search algorithm with cuckoo search for global numerical optimization," *Soft Computing*, vol. 20, no. 1, pp. 273–285, 2016.

[24] G.-G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, and M. Shao, "Hybridizing harmony search with biogeography based optimization for global numerical optimization," *Journal of Computational and Theoretical Nanoscience*, vol. 10, no. 10, pp. 2312–2322, 2013.

[25] J.-M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways," in *Evolutionary Computation, 1994. IEEE World Congress on*

Computational Intelligence., Proceedings of the First IEEE Conference on, pp. 312–317, IEEE, 1994.

[26] I. Ciornei and E. Kyriakides, "Hybrid ant colony-genetic algorithm (gaapi) for global continuous optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 1, pp. 234–245, 2012.

[27] A. Abdullah, S. Deris, M. S. Mohamad, and S. Z. M. Hashim, "A new hybrid firefly algorithm for complex and nonlinear problem.," in *DCAI*, pp. 673–680, Springer, 2012.

[28] G.-G. Wang, S. Deb, A. H. Gandomi, and A. H. Alavi, "A hybrid pbil-based krill herd algorithm," in *Computational and Business Intelligence (ISCBI), 2015 3rd International Symposium on*, pp. 39–44, IEEE, 2015.

[29] B. Wu, C. Qian, W. Ni, and S. Fan, "The improvement of glowworm swarm optimization for continuous optimization problems," *Expert systems with applications*, vol. 39, no. 7, pp. 6335–6342, 2012.

[30] H. Liu, Y. Zhou, Y. Yang, Q. Gong, and Z. Huang, "A novel hybrid optimization algorithm based on glowworm swarm and fish school," *Journal of Computational Information Systems*, vol. 6, no. 13, pp. 4533–4541, 2010.

[31] Y. Yang and Y. Zhou, "A hybrid glowworm swarm optimization algorithm for solving matrix eigenvalues," *Information-An International Interdisciplinary Journal*, vol. 14, pp. 999–1004, 2011.

[32] J.-l. Zhang, G. Zhou, and Y.-q. Zhou, "A new artificial glowworm swarm optimization algorithm based on chaos method," in *Quantitative Logic and Soft Computing 2010*, pp. 683–693, Springer, 2010.

[33] L. Qu, D. He, and J. Wu, "Hybrid coevolutionary glowworm swarm optimization algorithm with simplex search method for system of nonlinear equations," 2011.

[34] A. Ouyang, L. Liu, G. Yue, X. Zhou, and K. Li, "Bfgs-gso for global optimization problems.," *JCP*, vol. 9, no. 4, pp. 966–973, 2014.

[35] Y. Zhou, G. Zhou, and J. Zhang, "A hybrid glowworm swarm optimization algorithm for constrained engineering design problems," *Appl. Math. Inf. Sci*, vol. 7, no. 1, pp. 379–388, 2013.

[36] Y. Shi, Q. Wang, and H. Zhang, "Hybrid ensemble pso-gso algorithm," in *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, vol. 1, pp. 114–117, IEEE, 2012.

[37] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[38] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of heuristics*, vol. 8, no. 5, pp. 541–564, 2002.

[39] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[40] S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: a nature-inspired algorithm for global optimization," *Neural Computing and Applications*, vol. 27, no. 2, pp. 495–513, 2016.

[41] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*, pp. 169–178, Springer, 2009.

[42] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999.

[43] H. Faris, I. Aljarah, S. Mirjalili, P. A. Castillo, and J. J. Merelo, "Evolopy: An open-source nature-inspired optimization framework in python.," in *IJCCI (ECTA)*, pp. 171–177, 2016.