Modern Applied Science; Vol. 13, No. 1; 2019
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Moth Flame Optimization Based on Golden Section Search and its Application for Link Prediction Problem

Reham Barham[1], Ahmad Sharieh[1] & Azzam Sleit[1,2]

[1] Computer Science Department, King Abdullah II School of Information Technology, University of Jordan, Amman, Jordan

[2] KINDI Center for Computing Research, Qatar University, Qatar

Correspondence: Computer Science Department, King Abdullah II School of Information Technology, University of Jordan, Amman, Jordan. E-mail: rshbarham@gmail.com, sharieh@ju.edu.jo, azzam.sleit@ju.edu.jo

Received: January 27, 2018 Accepted: February 8, 2018 Online Published: December 5, 2018

doi:10.5539/mas.v13n1p10 URL: https://doi.org/10.5539/mas.v13n1p10

Abstract

Moth Flame Optimization (MFO) is one of the meta-heuristic algorithms that recently proposed. MFO is inspired from the method of moths' navigation in natural world which is called transverse orientation. This paper presents an improvement of MFO algorithm based on Golden Section Search method (GSS), namely GMFO. GSS is a search method aims at locating the best maximum or minimum point in the problem search space by narrowing the interval that containing this point iteratively until a particular accuracy is reached. In this paper, the GMFO algorithm is tested on fifteen benchmark functions. Then, GMFO is applied for link prediction problem on five datasets and compared with other well-regarded meta- heuristic algorithms. Link prediction problem interests in predicting the possibility of appearing a connection between two nodes of a network, while there is no connection between these nodes in the present state of the network. Based on the experimental results, GMFO algorithm significantly improves the original MFO in solving most of benchmark functions and providing more accurate prediction results for link prediction problem for major datasets.

Keywords: Benchmark Functions, Golden Ratio, Golden Section Search (GSS), Link Prediction Problem, Meta-heuristic Algorithm, Moth Flame Optimization (MFO)

1. Introduction

In order to improve the performance of Moth Flame Optimization algorithm, golden section search (GSS) strategy is utilized to develop a new version of MFO, which is called Golden Moth Flame Optimization (GMFO). GMFO focuses on enhancing the convergence rate of MFO through supporting the exploration mechanism by increasing the diversification of the search space (population), thus facilitating to get more intensification toward the best solution obtained so far in each iteration.

As in Mirjalili (2015), to explore the search space and to keep away from local solutions, MFO's search agents spend a large number of iterations. By this way, the exploitation of the MFO algorithm slows down and prevents the algorithm from locating a much better approximation of the global solution. So, GMFO is proposed to provide a mechanism that concentrates into direct the search agents towards the more promising region in the search space where the best solution can be found. Accordingly, the number of iterations to reach optimal solution will be reduced significantly and its convergence will be improved.

In order to achieve this improvement; the GSS features are employed for MFO to produce the proposed GMFO algorithm. The employment of GSS for the MFO is concentrated on updating the current best search space generated so far; by applying more exploration on it besides the local search that done by the original MFO such as logarithmic spiral. Performing further search space exploration makes the possibility to return to the same solution much less. Moreover, GSS ensures high diversification. Thus, it prevents solution from getting trapped in local optima and ensuring that direction is toward the global optima solution.

1.1 Moth Flame Optimization (MFO)

MFO, as in Mirjalili (2015), is a novel nature-inspired meta-heuristic and optimization paradigm. MFO optimizer is inspired from the method of moths' navigation in natural world. There is two moth's navigation; the first is called transverse orientation. In this navigation, a moth moves by trying to keep the same angle with regard to the light

resource (as known that moths are attracted by light resources). Given that the light resource is distant from the moth, keeping the same angle with respect to the light guarantees moth flying in a straight line. In addition to keep moving in a straight line, moth usually flies spirally around the region of the light, which is considered as the second navigation. Therefore, moth in the long run will converge on its way to the light. Mirjalili et al. (2015) utilized these navigation methods, transverse and spiral navigation, and developed the Moth Flame Optimization Algorithm. A conceptual model of these two navigation methods is illustrated in Figure 1.

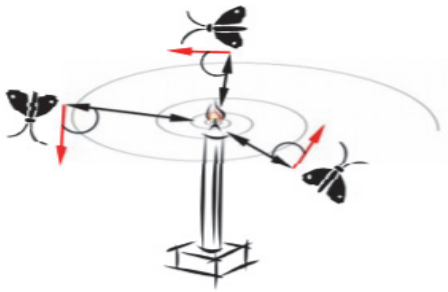


Figure 1. The transverse in a straight line (the red arrow) and the spiral movement of a moth (Mirjalili, 2015)

The general framework of MFO has three main stages. The initialization stage is the first one, where the MFO produces population of moths randomly, and computes their related fitness values. The second stage is the iteration stage, where the main function is carried out and the moths move in the region of the search space. In the final stage, the stop criterion is met. However, false is returned if the stop criterion is not met.

In MFO algorithm (Mirjalili, 2015), the main search space update method of moths is the logarithmic spiral. The movement of this spiral is the core piece of the MFO since it states how the moths update their locations in the region of flames. The movement by this spiral lets a moth wing around a flame and not essentially in the gap among them. For that reason, the exploration and exploitation mechanisms of the search space can be dependable. Logarithmic spiral is defined as in formula (1) (Mirjalili, 2015).

$$M\left(i,j\right) = D \cdot e^{bt} \cdot cos\left(2\Pi t\right) + F\left(i,j\right) \quad (1)$$

Where *M(i,j)* indicates the *j*th position for the *i*th moth, *F*(*i*,*j*) indicates the *j*th position for the *i*th flame, and *D* represents the distance of the *i*th moth from the *j*th flame. *b* is a constant for defining the nature of the logarithmic spiral, and *t* is a random number in [-1, 1]. *D* is calculated as in formula (2). Figure 2 shows the pseudo code of MFO algorithm.

$$D = |\left(F\left(i,j\right) - M\left(i,j\right)\right)| \quad (2)$$

Input: *popSize* // Population size, *Max_iteration*, *dim*// Number of features in the particular dataset,
Output: *Best_fitness* // Best fitness obtained so far

```
// Main loop
while Iteration<Max_iteration+1
    Update flame_no Moth_fitness = CostFunction(dataset.x, dataset.y, P);
    if iteration == 1
     fitness_sorted = sort(Moth_fitness);
     F = sort(M); //sorted population M based on sorted fitness
            //Update the flames
            best_flames=F;
            best_flame_fitness=fitness_sorted;
    Else
```

```
        fitness_sorted = sort(previous_fitness; best_flame_fitness);
       F = sort(previous_population; best_flames);
        //Update the flames
        best_flames=sorted_population;
        best_flame_fitness=fitness_sorted;
    end
    r=-1+Iteration*((-1)/Max_iteration);
    for i = 1: popSize
     for j = 1: dim
          b=1; t=(a-1)*rand+1;
          Calculate D // by using formula (2) with respect to the corresponding moth
          Update Best_Position and Best_fitness//by using formula (1)
     Endfor
    Endfor
end-while
```

Figure 2. The pseudo code of MFO algorithm (Mirjalili, 2015)

The MFO algorithm was compared with other well-known nature-inspired algorithms on 29 benchmark and 7 real engineering problems which are: Welded beam design problem, Gear train design problem, Three-bar truss design problem, Pressure vessel design problem, Cantilever beam design problem, I-beam design problem, and Tension/compression spring design (Mirjalili, 2015). The statistical results on the benchmark functions demonstrate that MFO is capable to offer competitive results. As well, the results of the real problems show the advantages of this algorithm in solving difficult problems with constrained search spaces.

1.2 Golden Section Search (GSS)

The GSS is a search method aims at locating the best maximum or minimum point of a one-dimensional function by narrowing the interval that containing this point iteratively until a particular accuracy is reached (Press et al., 2007).

To understand the concept of GSS search method and how could be applied (Press et al., 2007) on an application, suppose that there are function f, its domain is represented by x, where $x \in [x_{min}, x_{max}]$, x_{min} is the lower bound of domain x and x_{max} is the upper bound. The interval $[x_{min}, x_{max}]$ is an interval where the best maximum or minimum point of f is supposed to be enclosed in. This interval should be divided into three regions. This is done by adding two points from the function domain x and located within the defined interval; such as the internal points x_1 and x_2, where x_1 and $x_2 \in [x_{min}, x_{max}]$ and $x_1 < x_2$. See Figures 3.a and 3.b.

After that, the function needs to be evaluated at these two internal points (Press et al., 2007). In this case, we try to find the promising interval where the best minimum point for function f is believed to be there. This evaluation is carried out as follows. If $f(x_1) < f(x_2)$ is true, then the minimum will be located between x_{min} and x_2. So, x_{min} will still the same while x_2 become the value of x_{max}. But if $f(x_1) > f(x_2)$ becomes true, the point will be located between x_1 and x_{max}. So, x_1 will be the x_{min}, while x_{max} remains the same. Again, the new smaller interval resulted from this evaluation should be divided into three sections.

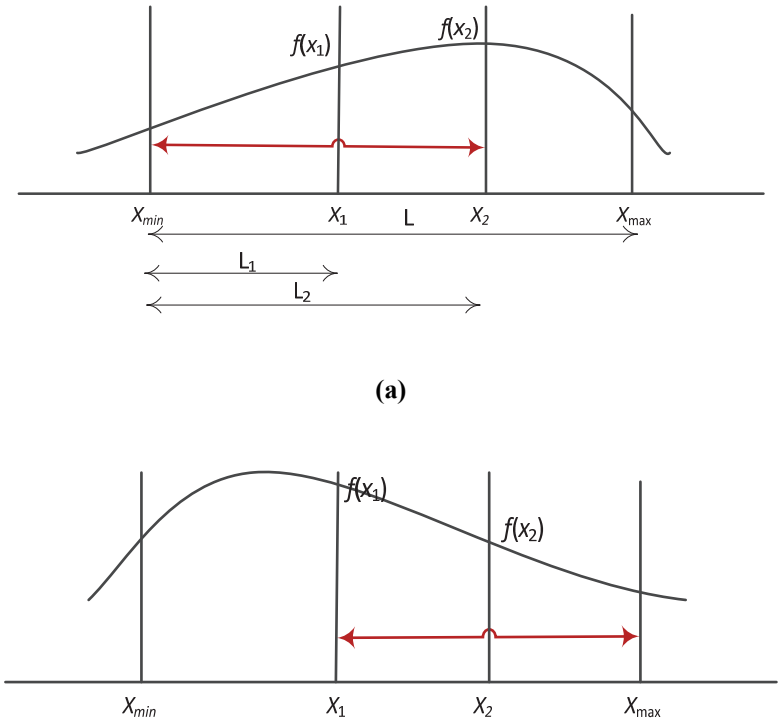


Figure 3. How Golden Section Search evaluates the function f within the interval [x_{min}, x_{max}], and divides this interval into 3 sections. Red arrow represents the new small interval. (a) Represents when f(x_1)<f(x_2), while (b) represents when f(x_1)> f(x_2).

The most important question rises here is how to determine where to divide the interval into sections or where to locate the intermediate points x_1 and x_2. The answer of this question can be found by using the golden ratio.

1.2.1 Golden Ratio (GR)

Golden section search has this name because it depends on the special ratio; namely Golden Ratio. Golden Ratio is used to locate the intermediate points such as x_1 and x_2, which divide the given interval into smaller one to narrow the search space and to determine the next promising smaller interval where the best solution hope to have (Press et al., 2007).

To understand the concept of the Golden Ratio, set the following conditions, based on Figure 3.a, as in formulas (3) and (4) (Press et al., 2007).

$$L = L_1 + L_2 \tag{3}$$

$$GR = L / L_2 = L_2 / L_1 \tag{4}$$

It is noticed that the ratios in formula (4) are equals. These ratios have a special value which is called the Golden Ratio GR. To determine the value of Golden Ration, formula (3) can be substituted into formula (4) to give formula (5).

$$1 + GR = 1 / GR \quad or \; GR^2 + GR - 1 = 0 \tag{5}$$

Solve the quadratic equation in formula (5) for GR and use the positive root, and then a value of GR is given as in formula (6) (Press et al., 2007).

$$GR = (\sqrt{5} - 1) / 2 = 0.61803 \tag{6}$$

Consequently, the intermediate points x_1 and x_2 are located where the ratio of the distance from these points to the ends of the given interval (which represents the search region) is equal to the golden ratio. So, x_1 and x_2 can be

computed as in formulas (7) and (8) which can be interpreted as the second interval will be smaller GR times the previous interval.

$$x_1 = x_{\max} - GR(x_{\max} - x_{\min}) \tag{7}$$

$$x_2 = x_{\min} + GR(x_{\max} - x_{\min}) \tag{8}$$

Accordingly, this study presents, illustrates and tests the Golden Moth Flame optimization algorithm GMFO, which is considered as an improvement of the well-known MFO algorithm (Mirjalili, 2015). The performance of the two algorithms will be investigated when solving the link prediction problem.

1.3 Link prediction Problem

Generally, Graphs provide a natural abstraction to represent interactions between different entities in a network (Srinivas and Mitra, 2016; Sharieh, et al., 2008; Barham, et al., 2016). A real or synthetic network in the world can be represented as a graph of nodes and edges. As in social network, users are represented as nodes, while the interactions between these users whether are associations, collaborations, or influences are presented as edges between these nodes (Liben-Nowell and Kleinberg, 2007). For instance, if there is a snapshot of a social network at time t_0, link prediction looks for predicting the edges that will be added to the network during the interval from time t_0 to a given future time t accurately (Liben-Nowell and Kleinberg, 2007). So, the link prediction problem is related to the problem of inferring missing links from an observed network (Liben-Nowell and Kleinberg, 2007).

Importance of link prediction can be found in its wide variety of applications. Graphs used to represent as social networks, transportation networks, or disease networks (Srinivas and Mitra, 2016; Barham, et al., in press). Link prediction can specially be functional on these networks to analyze and solve interesting problems like predicting occurrence of a disease, managing privacy in networks, discovering spam emails, signifying another routes for probable navigation based on the current traffic models (Srinivas and Mitra, 2016).

More formally, the link prediction task can be formulated as follows (Liben-Nowell and Kleinberg, 2007). An un-weighted and undirected graph G = (V, E), represents a topological structure of a network, such as social networks, where V is the set of nodes in G, and E is the set of existed edges in G. In this G, each edge e = (u, v) $\in$ E and nodes u, v $\in$ V represents an interaction between u and v that took place at a particular time t(e). For two time instances, t and t′, where t′ > t, let G[t, t′] denotes a sub-graph of G consisting of all edges with timestamps between t and t′. And let t_0, t'_0, t_1, and t'_1 be four time instances, where $t_0 < t'_0 \leq t_1 < t'_1$. Then for a given network $G[t_0, t'_0]$, the output is a list of edges does not present in $G[t_0, t'_0]$ that are predicted to appear in the network $G[t_1,t'_1]$.

The rest of the paper is organized as follows. Section 2 is about the related works benefits from utilizing golden section search method. Section 3 illustrates the GMFO methodology. Section 4 presents the experimental results of GMFO and MFO on 15 benchmark functions and their discussions. Conclusion and future work is in Section 5.

2. Related Works

Due to the importance of golden section search and the golden ratio in mathematics, optimization and its applications in different fields, various approaches were proposed and utilized them in their works. This section presents some of these works to prove the successes of using GSS for various researches. This is besides the works deployed the MFO to solve particular problems.

Patel, et al. (2013) applied GSS for tracking the maximum power point (MPPT) for solar photovoltaic system (PV). Their approach benefits from using GSS which reflected in its fast response, robust performance and guaranteed convergence. They conclude that GSS can be a competitive method for PV generation systems because of its good performance.

Djeriou, et al. (2018) applied a maximum power point tracking procedure for stand-alone solar water pumping system to get better overall operating competence. This method is based on golden section search optimization method. GSS method provides two advantages: fastness and perturbation-free which both have an effect on the overall and direct effectiveness of the solar water pumping system.

Liang, et al. (2016) examined the special elements of gait with and without the effect of clothe. They proposed the golden ratio-based segmentation method to decrease the influence of clothing. Their Experimental results are conducted and showed that their proposed method outperforms other approaches of segmentation. They found that the key problem is to find out the exact part of clothe, and discard it. Considering that the individual body is conventional to the golden ratio, and that clothe is designed according to this ratio, the golden ratio segmentation

technique is used to discard the effect of clothing.

Hassaballah, et al. (2013) proposed a new approach for face detection evaluation based on the golden ratio. The new evaluation measure is more practical and precise compared to the existing one for face detection. They conclude that "the golden ratio helps in estimating the face size according to the distance between the centers of eyes".

Böcker (2012) proposed algorithm to solve NP-complete cluster editing problem based on the golden ratio. They presented a search tree algorithm for the problem with repeatedly branch the vertices can be isolated which improves the running time.

Tsai, et al. (2010) proposed to employ the golden section search optimization algorithm to determining a good quality shape parameter of multi-quadrics for the solution of partial differential equations. Experimental results show that the proposed golden section search method is valuable and gives a reasonable shape parameter beside satisfactory precision of the solution.

Koupaei, et al. (2016) proposed a practical version of golden section search algorithm to optimize objective functions. Consequently, their work presented an algorithm takes the benefits and capabilities for both of chaotic maps and the golden section search method in order to solve nonlinear optimization problems. Practically, the proposed algorithm reduces the searching space by chaotic maps and solving an optimization problem on new promising space using the golden section search technique.

Due to the success achieved by GSS in most approaches proposed in literature, this work tends to improve the MFO algorithm based on applying the Golden Section Search in solving the link prediction problem.

3. Methodology: Golden Moth Flame Optimization (GMFO)

As in Mirjalili (2015), to explore the search space and to keep away from local solutions, MFO's search agents spend a large number of iterations. By this way, the exploitation of the MFO algorithm slows down and prevents the algorithm from locating a much better approximation of the global solution. So, GMFO is proposed to provide a mechanism that concentrates into direct the search agents towards the more promising region in the search space where the best solution can be found. Accordingly, the number of iterations to reach the best will be reduced significantly and its convergence will be improved.

In order to achieve this improvement, the Golden Section Search GSS features are employed for MFO to produce the proposed GMFO algorithm. The employment of GSS for the MFO is concentrated on updating the current best search space generated so far; by applying more exploration on it besides the local search that done by the original MFO such as logarithmic spiral. Performing further search space exploration makes the possibility to return to the same solution much less. Moreover, GSS ensures high diversification. Thus, preventing solution from getting trapped in local optima and ensuring that direction is toward the global optima solution.

In other words, applying the Golden Section Search for MFO can be considered as a process for updating the search space, where the search space represents the generated population. Thus, while searching the GSS tries to find the promising region in the search space where the best solution is supposed to have. So, GSS helps to generate more promising population. When the promising region is reached, searching for a much better solution than the later is performed. This search method accelerates the process of finding the best solution by considering the most promising population in iteration. This acceleration can be translated into enhancing the convergence behavior of the MFO algorithm. Table 1 represents the main concepts of applying GSS for GMFO.

Table 1. The main concepts of the GMFO derived from the Golden Section Search

Golden Section Search	GMFO concepts
Function domain	The initial search space or Population
Divide the domain into sections	Explore and Update search space
Find the smaller interval from the domain	Get best region in the search space(promising population)

A good search space exploration mechanism can be provided by the golden section search where the moving toward the global optima solution is ensured. So, the solutions don't return to the same solutions once more. Also, golden section search ensures high diversity so that prevent solution from getting trapped in local optima (Press et al., 2007). The pseudo code for the Golden Section Search function adapted for GMFO and the Golden MFO

algorithm are demonstrated in Figure4 and 5, respectively.

GoldenSectionSearch ( )

Input: x_{min}, x_{max}// Current search space limits, GR=0.61803// The golden ratio, $f(x_{best})$//*Best_fitness* obtained so far

Output: x_{min}*, x_{max}*// The new search space limits

Step 1: Consider x_{min}, x_{max} as the current search space limits

Step2: Compute the intermediate points x_1 and x_2 using formulas (7) and (8) respectively

//create new search space limits

Step 3: Calculate the fitness value using the cost function for x_1 and x_2 so we have $f(x_1)$ and $f(x_2)$

Step4: Ensure that $f(x_1)$ or $f(x_2)$ is less than $f(x_{best})$ and update x_1 and x_2 accordingly

If($f(x_{best}) < f(x_1)$**) then**

$x_1 = x_{best}$;

if $f(x_{best}) < f(x_2)$ **) then**

$x_2 = x_{best}$;

Step5: Update the promising interval according to x_1 and x_2.

if ($f(x_1) < f(x_2)$) **then**

x_{min}*= x_{min}**;**

x_{max}*= x_2 ;

if ($f(x_1) > f(x_2)$) **then**

x_{min}*= x_1**;**

x_{max}*= x_{max} ;

Figure 4. The pseudo code of Golden Section Search Function for GMFO

1. **Initialize** *SearchAgents_no* //number of search agents, *Max_iteration*, and *dim* //number of problem variables, *iteration*=1, *GR*=0.61803// golden ratio
2. **Define** x_{max} // x_{max} =[$x_{max\ 1}$, $x_{max\ 2}$,..., $x_{max\ n}$] where $x_{max\ n}$ is the upper bound of variable *n*, x_{min} // x_{min} =[x_{min1} , x_{min2} ,..., $x_{min\ n}$] where $x_{min\ n}$ is the lower bound of variable *n*
3. **Generate** population of moths *X* based on the initial x_{min} and x_{max}, *SearchAgents_no* and *dim*
4. **Repeat**
5. **Update** *flame_no*
6. **If** *iteration*>1
7. (x_{min}*, x_{max}*) = **GoldenSectionSearch**(x_{min}, x_{max}, *GR*, *f*(x_{best}))// as in Figure 4
8. **Update the search space limits**
9. x_{min}= x_{min}*;
10. x_{max}= x_{max}*;
11. **Check** if moths *X* more than search space x_{max} upper limit or less than x_{min} lower limit, and then normalize them to make their values be within the current search space interval [x_{min}, x_{max}].
12. **Calculate** the fitness of moths
13. **sorted** population *X* based on sorted fitness
14. **Update** the flames
15. **Identify** *Best_solution, Best_fitness*
16. **Compute** *r*=-1+*Iteration**((-1)/*Max_iteration*);
17. **for** *i* = 1*: SearchAgents_no*
18. **for** *j* = 1*: dim*
19. *b*=1*; t*=(*a*-1)**rand*+1*;*
20. **Calculate** *D* // by using Formula (2) with respect to the corresponding moth
21. **Update** moths *X* //by using Formula (1)
22. **end**
23. **end**
24. *iteration=iteration*+1;
24. **Until** (*Max_iteration* is reached)
25. **Return** x_{best}*//Best_solution, f*(x_{best})//Best_fitness

Figure 5. The pseudo code of GMFO algorithm

GMFO algorithm starts by initializing number of parameters such as number of search agents, maximum number of iteration, number of problem variables and setting the golden ratio to 0.61803. See Figure 5, (line 1). After that, the upper limit and lower limit for each variable of the underlying dataset should be specified. See Figure5, (line 2). These limits are considered as the initial domain or search space where the search agents try to find the best solution so far. Based on the defined interval, a population X has been generated. Its number of candidate solutions represented by searchAgents_no parameters. Each candidate solution is represented by a vector of size *dim* (which is the number of problem variables or features). See Figure5, (line 3).

Now, it is the turn of the main loop which considers reaching maximum number of iteration as its stop criteria. See lines 4 and 24 in Figure 5. After that, the flame number is computed. There is no need to call the golden section search function in the first iteration because the initial domain limits x_{min} and x_{max} are available. At the second iteration, if statement (Figure 5, line 6) becomes true, the golden search function will be called as in Figure 4.

Golden search function uses x_{min} and x_{max} which are the current domain limits, *GR* which is golden ratio = 0.68103 and $f(x_{best})$ which is the fitness value for the best solution x_{best} obtained so far. The goal is to update the current interval $[x_{min}, x_{max}]$ into smaller one $[x_{min}*, x_{max}*]$, which can be considered as promising interval because the best solution expected to be within. See line7 Figure 5.

As in Figure 4, the Golden Section Search, which represents the core of GMFO, is performed as follows. Based on the current search space limits x_{min} and x_{max} and *GR* the intermediate points x_1 and x_2 can be computed using formulas (7) and (8), respectively (See steps 1 and 2, Figure 4). Then, calculate the fitness value for each x_1 and x_2 using the cost function to obtain $f(x_1)$ and $f(x_2)$, respectively.

The intermediate points x_1 and x_2 should be verified to make sure they represent the appropriate points and lead to the most promising interval where the best solution is supposed to have and to avoid trapping in local minima. Thus, the fitness values $f(x_1)$ and $f(x_2)$ should be compared with the best fitness obtained so far $f(x_{best})$. Because this work focuses on the minimization problems, the best is the lower. Therefore, if $f(x_{best}) < f(x_1)$, then the x_{best} value will be assigned to x_1; else, x_1 will remain unchanged. The same will happen to x_2 such that if $f(x_{best}) < f(x_2)$ then the x_{best} will be the value of x_2; else, x_2 will remain unchanged. See step 4 Figure4.

Upon updating $x_1$and x_2 in comparison with x_{best}, the promising interval will be computed by conducting a comparison between $f(x_1)$ and $f(x_2)$. If $f(x_1) < f(x_2)$ then the interval is $[x_{min,} x_2]$; otherwise, the interval will become $[x_{1,} x_{max}]$. At the end of this function, the values of x_{min} and x_{max} are updated and returned. See step 5 Figure 4.

Now, we can return to the GMFO steps demonstrated in Figure 5. Line11 represents the most important step in the algorithm after applying the GSS function. This step facilitates updating the population according to the promising interval. This is done by normalizing each candidate solution in the population within the promising interval limits. This updating makes the population more promising and increases the probability to have the best solution. As a result, the number of iterations needs to reach the best solution will be decreased and the algorithm will converge fast.

Next, the fitness value for the updated population is calculated. The population is sorted according to sorted fitness value. Afterward, the best solution is identified. See lines 12-15 Figure5.

Later, generating the next population is performed using the formulas (1) and (2). This new generation will have their own values of promising interval limits, using the GSS function, and will be updated accordingly. Lines 5-20 in Figure5 will be repeated until the maximum iteration is reached. Finally, the overall best solution will be returned (See line 25, Figure 5).

4. Experimental Results and Discussions

In this section, the experimental results of testing 15 benchmark functions on the proposed algorithm GMFO and the original MFO algorithm are presented evaluated and discussed.

4.1 Experiments Platform

All experiments are implemented in Matlab R2017a and conducted using Intel® Core™ i7-5500 CPU @ 2.40 GHz processor, 8.00GB RAM, and Microsoft Windows8 of 64-bit Operating System.

Each experiment is repeated for 30 times independently. Reaching maximum number of iterations is adopted as the stopping criteria. The maximum number of iterations is 1000 and the population size is 50.

4.2 Benchmark Functions

Commonly, the optimization benchmark functions are used to measure the performance of algorithms designed for optimization. These functions are a set of well-known mathematical functions with identified global optima (BenchmarkFcns, 2018). As in most literature such as in Molga and Smutnicki (2005), the proposed GMFO is employed using 15 benchmark functions and a comparison among the original MFO is conducted. These test functions are classified into three sets: uni-modal, multimodal, and composite.

The uni-modal functions (F_1-F_7) with their related number of variables (or dimensions), ranges and minimum return value are illustrated in Table 2. The appropriate way for benchmarking the exploitation of an algorithm is by using the uni-modal test functions. This is because they encompass one global optimum with no local optima.

Table 2. The Uni-modal benchmark functions

Function	Variable number	Range	f_{min}
$F_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	100	[-100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	100	[-10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	100	[-100,100]	0
$F_4(\mathbf{x}) = \max_{i=1,\ldots,n} \lvert x_i \rvert$	100	[-100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	100	[-30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	100	[-100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	100	[-1.28, 1.28]	0

On the other hand, multimodal functions (F8-F13), which are illustrated in Table 3, encompass a considerable number of local optima and they are useful to benchmark the exploration process and avoid trapping in a local optimum.

Table 3. The multi-modal benchmark functions

Function	Variable number	Range	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	-8.9829x5
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0
$F_{10}(\mathbf{x}) = -20.exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	30	[-32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[-600,600]	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_i) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $, where\ y_i = 1 + \frac{x_i + 1}{4}$	30	[-50,50]	0
$F_{13}(x) = 0.1\{\sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)]$ $+(x_n - 1)^2[1 + \sin^2(2\pi x_n)]\}$	30	[-50,50]	0

Ultimately, composite functions are mixture of various rotated, shifted, and biased multimodal benchmark functions. This work employed F_{14} and F_{15} composite functions as illustrated in Table4. Multimodal benchmark functions are helpful for benchmarking the balance among exploration and exploitation.

Table 4. Description of composite benchmark functions

Function	Variable number	Range	f_{min}
$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x^i - a_{ij})^6})^{-1}$	2	[-65,65]	1
$F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_i (b_2 + b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	[-5,5]	0.00030

4.3 Experimental Results of Benchmark Functions and Discussions

Tables 5 through 7 show the experimental results of applying GMFO and MFO for testing the benchmark functions separately. Each experiment is performed for 30 independent runs. The average and standard deviation of the best fitness among these runs are taken. Bold values mean that GMFO algorithm is better, while underline is used to show that MFO algorithm is better. The Ave and STD represent the average fitness value, and standard deviation, respectively.

A comparison between GMFO and MFO over the unimodal functions (F_1-F_7) is demonstrated in Table 5. Results in this table indicate that GMFO perform better than MFO for most unimodal functions such as F_1- F_3 and F_5- F_7. This indication proves that MFO based on using golden section search improve the exploitation of the algorithm; however the unimodal functions is known to encompass one global optimum with no local optima.

Besides Table 5, Table 8 supports this indication by presenting the p values produced by Wilcoxon's rank-sum test. This test is computed for GMFO against MFO. p values less than or equal to 0.05 indicate that there are a significant difference between the results of GMFO and MFO, and thus there is a tangible improvement. Bold values indicate that GMFO is better, while underline represents that MFO is the better.

Table 5. The average (Ave) and standard deviation (STD) of best fitness value over 30 runs of uni-modal benchmark functions.

	GMFO		MFO	
F	***Ave***	***STD***	***Ave***	***STD***
F1	**7.47574E-08**	**2.88194E-07**	10.0002	30.5129
F2	**3.0477E-25**	**3.17343E-25**	1.97518E-19	1.73683E-19
F3	**2.24408E-05**	**7.46131E-05**	166.6666671	912.8709291
F4	4.9622861	3.701332387	1.806507279	3.809898924
F5	**58.46142813**	**114.5859596**	3036.220773	16426.14749
F6	**1.12988E-33**	**5.08491E-33**	1.24377E-29	2.74382E-29
F7	**0.002902571**	**0.001753113**	0.005807052	0.003524263

In addition, a comparison between GMFO and MFO over the multimodal functions (F_8-F_{13}) is demonstrated in Table 6. Results in this table indicate that GMFO perform better than MFO for most multimodal functions such as F_8- F_{12}. This indication proves that GMFO, which based on using golden section search, has the ability to improve the exploration of the search space; however the multimodal functions is known to encompass a considerable number of local optima and they are useful to benchmark the exploration process and avoid trapping in a local optimum.

Besides Table 6, Table 8 supports this indication by presenting the p values produced by Wilcoxon's rank-sum test. This test is computed for GMFO against MFO. p values less than or equal to 0.05 indicate that there are a significant difference between the results of GMFO and MFO, and thus there is a tangible improvement. Bold

values indicate that GMFO is better, while underline represents that MFO is the better.

Table 6. The average (Ave) and standard deviation (STD) of best fitness value over 30 runs of multimodal benchmark functions

	GMFO		**MFO**	
F	***Ave***	***STD***	***Ave***	***STD***
F8	**-3249.046644**	**329.7069718**	-2205.984642	386.8933629
F9	**14.52639033**	**5.892617888**	24.44984768	16.2104948
F10	**3.25665E-15**	1.7034E-15	4.44089E-15	0
F11	**0.189647786**	0.105119607	0.365687673	0.104210132
F12	**0.077793172**	**0.176999958**	0.197112411	0.526561225
F13	0.055079947	0.29133895	0.004394946	0.005474706

Furthermore, a comparison between GMFO and MFO over the composite functions (F_{14} and F_{15}) is demonstrated in Table 7. Results in this table indicate that GMFO perform better than MFO for composite function F_{15}. This indication proves that GMFO, which based on using golden section search, has the ability to provide a balance between the exploration and the exploitation of the search space; thus converge faster besides avoid trapping in local optima.

Table 7. The average (Ave) and standard deviation (STD) of best fitness value over 30 runs of composite benchmark functions

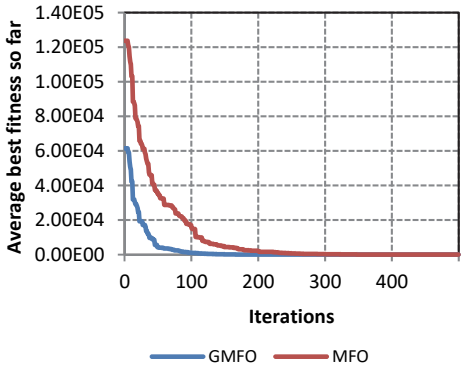
	GMFO		**MFO**	
F	***Ave***	***STD***	***Ave***	***STD***
F14	**1.922873929**	**1.243374734**	3.944772406	2.34073245
F15	**0.000033808**	**0.000199281**	0.000903356	0.000326963

Table 8 supports this indication by presenting the p values produced by Wilcoxon's rank-sum test. p values for F_{14} and F_{15} are less than or equal to 0.05. This indicates that there are significant differences between the results of GMFO and MFO, and thus, there is a tangible improvement.

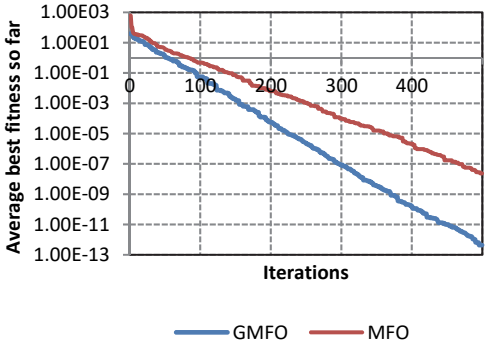
Table 8. P-values of the Wilcoxon rank-sum test over all 30 runs (p>=0.05 have been underlined)

F	**GMFO x MFO**	**F**	**GMFO x MFO**
F1	0.00020	**F9**	0.00670
F2	0.00001	**F10**	0.00120
F3	0.032730	**F11**	0.069780
F4	0.02950	**F12**	0.051950
F5	0.02860	**F13**	0.034830
F6	0.04570	**F14**	0.082900
F7	0.00300	**F15**	0.0107
F8	0.06580		

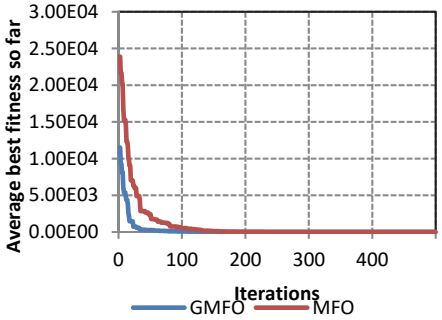
Figures 6 (a) through 6 (o), the convergence curves of the average fitness value for GMFO and MFO for 1000 iterations is plotted. Through these figures, it is noticed that GMFO converge faster than the original MFO algorithm. Thus GMFO has the ability to search the space faster and find the best region in it where the solution is supposed to be there. Accordingly, this notice supports that GMFO significant improvement for MFO algorithm. Based on the experimental results, the indication is GMFO performs better than MFO for most unimodal, multimodal and composite functions. This indication proves that MFO based on using golden section search improve exploration of the search space and hence the convergence to find the best solution. So, GMFO has the ability to search the space and find the best solution more efficiently than MFO.



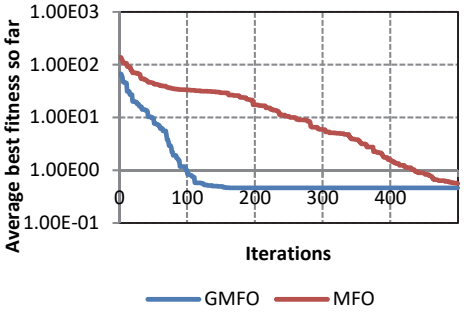
(a) F_1



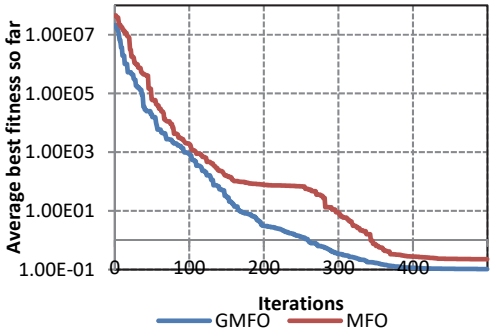
(b) F_2(log scale)



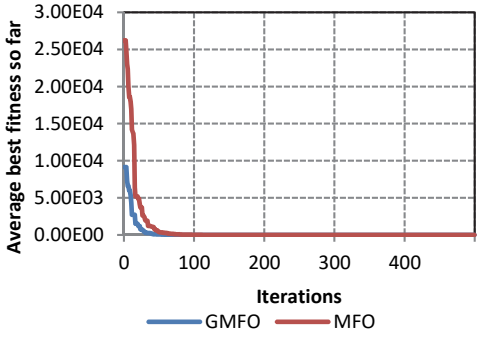
(c) F_3



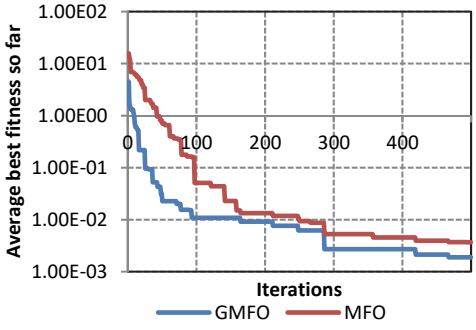
(d) F_4 (log scale)



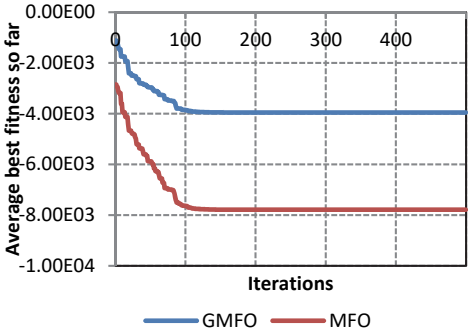
(e) F_5



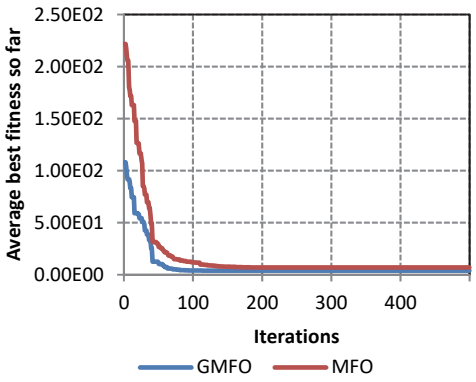
(f) F_6 (log scale)



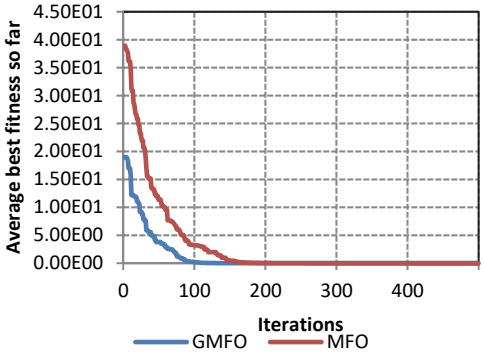
(g) F_7 (log scale)



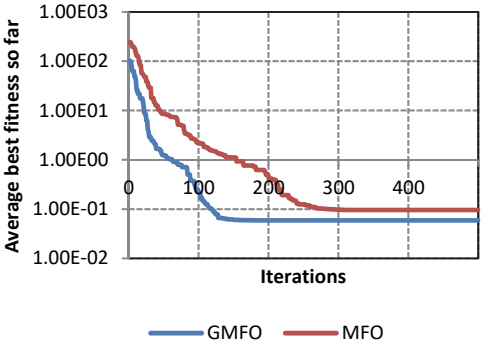
(h) F_8 (log scale)



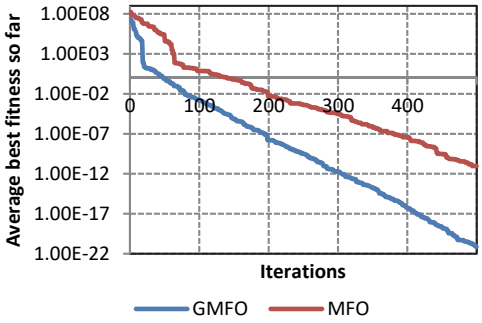
(i) F_9



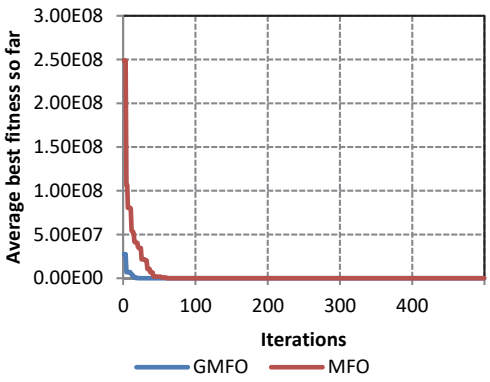
(j) F_{10}



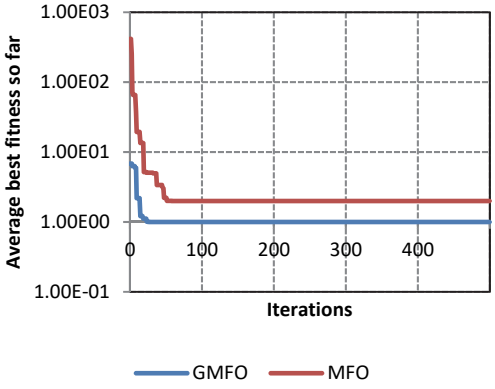
(k) F_{11} (log scale)



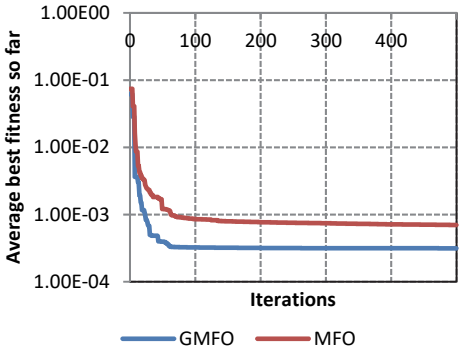
(l) F_{12} (log scale)



(m) F_{13}



(n) F_{14} (log scale)



(o) F_{15} (log scale)

Figure 6. The convergence curves of the average fitness value over 30 independent runs

4.4 Experimental Results for Link Prediction Problem and Discussions

In order to test the performance of GMFO in comparison with other algorithms for link prediction problem, GMFO and other well-known meta-heuristic algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) are carried out on five datasets and a comparison between them is conducted in term of the prediction accuracy measured by Area under Curve (AUC).

To solve link prediction problem using an optimization algorithm, a design for both of candidate solutions and fitness function are required. Group of candidate solutions represent the population. In this study, each candidate solution is represented by an adjacency matrix of zeros and ones. Zeros represent missing links, while ones are for existing links (Barham and Aljarah, 2017).

These matrices are generated randomly and compared to the actual representation of the underlying network. This comparison measure the prediction accuracy based on the AUC metric which represents the fitness function for prediction. To compute the value of AUC, confusion matrix is required. See Table 9.

Table 9. Confusion matrix of a classifier (Lui, 2011)

	Predicted positive	Predicted negative
Actual positive	**TP**	**FN**
Actual negative	**FP**	**TN**

Where:
TP: the number of correct classifications of the positive instances (true positive),
FN: the number of incorrect classifications of positive instances (false negative),
FP: the number of incorrect classifications of negative instances (false positive),
TN: the number of correct classifications of negative instances (true negative).

AUC is usually used to assess the classification results on the positive class in two classes' classification problems. The classifier requires ranking the test results according to their likelihoods of belonging to the positive class with the most likely positive class ranked at the top (Lui, 2011). The true positive rate (TPR) represents recall as in formula (10), while FPR is defined as in formula (11) (Lui, 2011). Greater AUC means better prediction model. AUC could be computed using formula (9).

$$AUC = \frac{1}{2}(1 + TPR)(1 - FPR) \tag{9}$$

$$recall = \frac{TP}{TP + FN} \tag{10}$$

$$FPR = \frac{FP}{FP + TN} \tag{11}$$

4.4.1 Datasets

Table 10 shows the underlying datasets and their details of number of nodes and edges. These datasets are (Link Prediction Group (LPG), 2016): US airport network (USAir), network of the US political blogs (Political blogs), co-authorships network between scientists (NetScience), network or protein interaction (Yeast), and the King James Bible and information about their occurrences (King James) (L¨u, et al., 2016).

Table 10. The datasets details, number of nodes and edges

Dataset Name	Number of nodes	Number of edges
USAir	332	2126
NetScience	1589	2742
Political Blogs	1222	5366
Yeast	2375	6136
King James	1773	9131

4.4.2 Tests on AUC Performance Metric

To evaluate the quality of the GMFO for link prediction, performance metrics such as AUC is computed for each algorithm (GMFO, MFO, PSO and GA) over the five datasets. The best, average, and standard deviation of the AUC results over 30 independent runs are computed as in Table 11 for 30 independent runs.

Greater AUC means better prediction model. Highest AUC for one algorithm means highest prediction quality among other algorithms.

As shown in Table 11, among all algorithms, GMFO has the highest average AUC score. Even for the largest King James dataset, in number of edges, the GMFO still the best with 0.6966, while the other algorithms get average AUC scores from 0.5746 to 0.6013.

Table 11. AUC results for each algorithm: The best, average, and standard deviation of the AUC over 30 independent runs.

Dataset / Algorithm		MFO	GMFO	GA	PSO
USAir	**Best**	0.9084	**0.9878**	0.8561	0.8843
	Average	0.8495	**0.8818**	0.8313	0.8696
	STDEV	0.0503	0.0075	**0.0047**	0.0067
NetScience	**Best**	0.9408	**1.0000**	0.7982	0.8498
	Average	0.8239	**0.9384**	0.7652	0.8229
	STDEV	0.0595	0.0958	0.0496	**0.0230**
Political blogs	**Best**	0.7829	**0.8385**	0.7791	0.7879
	Average	0.7783	**0.7953**	0.7680	0.7664
	STDEV	0.0071	**0.0022**	0.0048	0.0087
Yeast	**Best**	0.7630	**0.7640**	0.7594	0.6139
	Average	0.6965	**0.7098**	0.6739	0.5739
	STDEV	0.0069	0.0076	**0.0003**	0.0063
King James	**Best**	0.6173	**0.6966**	0.6236	0.6910
	Average	0.6016	**0.6176**	0.6013	0.5746
	STDEV	0.0081	0.0030	0.0098	**0.0008**

5. Conclusion and Future Work

In order to improve the performance of Moth Flame Optimization algorithm, golden section search strategy is utilized to develop a new version of MFO, which is called Golden Moth Flame Optimization (GMFO). GMFO focuses on enhancing the convergence rate of MFO through supporting the exploration mechanism by increasing the diversification of the search space (population), thus facilitating to get more intensification toward the best solution obtained so far in each iteration.

Accordingly, this paper presents, illustrates and tests the Golden Moth Flame optimization algorithm GMFO, which is considered as an improvement of the well-known MFO algorithm.

Golden section search (GSS) is a search method aims at locating the best maximum or minimum point of a one-dimensional function by narrowing the interval that containing this point iteratively until a particular accuracy is reached. The experimental results of testing 15 benchmark functions on the proposed algorithm GMFO and the original MFO algorithm are presented evaluated and discussed.

Experimental results indicate that GMFO perform better than MFO for most uni modal, multimodal and composite functions. This indication proves that MFO based on using golden section search improve exploration of the search space and hence the convergence. So, GMFO has the ability to search the space and find the best solution more efficiently than MFO. As a future work, GMFO can be applied to solve different application and problems.

In addition, to evaluate the quality of the link prediction, performance metrics such as AUC is computed for GMFO and compared with MFO, PSO and GA algorithms over five datasets. The best, average, and standard deviation of the AUC results over 30 independent runs are computed. Among all algorithms, GMFO has the highest average AUC score for the prediction results. Future work is focused on applying the proposed GMFO for providing a solution for other significant applications.

References

Barham, R., & Aljarah, I. (2017, October). Link Prediction Based on Whale Optimization Algorithm. In *New Trends in Computing Sciences (ICTCS), 2017 International Conference on* (pp. 55-60). IEEE. https://doi.org/10.1109/ICTCS.2017.41

Barham, R., Sharieh, A., & Sliet, A. (2016). Chemical reaction optimization for max flow problem. (*IJACSA) International Journal of Advanced Computer Science and Applications*, *7*(8). https://doi.org/10.14569/IJACSA.2016.070826

Barham, R., Sharieh, A., & Sliet, A. (in press). A meta-heuristic framework based on clustering and preprocessed datasets for solving the link prediction problem. *Journal of Information Science*, (Accepted in 31 October 2018).

Böcker, S. (2012). A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, *16*, 79-89. https://doi.org/10.1016/j.jda.2012.04.005

Hassaballah, M., Murakami, K., & Ido, S. (2013). Face detection evaluation: A new approach based on the golden ratio $${\Phi} $$. *Signal, Image and Video Processing*, *7*(2), 307-316. https://doi.org/10.1007/s11760-011-0239-3

IEEE. (2018). Benchmark Fcns. Retrieved from http://benchmarkfcns.xyz/about/

Koupaei, J. A., Hosseini, S. M. M., & Ghaini, F. M. (2016). A new optimization algorithm based on chaotic maps and golden section search method. *Engineering Applications of Artificial Intelligence*, *50*, 201-214. https://doi.org/10.1016/j.engappai.2016.01.034

Liang, Y., Li, C. T., Guan, Y., & Hu, Y. (2016). Gait recognition based on the golden ratio. *EURASIP Journal on Image and Video Processing*, *2016*(1), 22. https://doi.org/10.1186/s13640-016-0126-5

Liben-Nowell, D., & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, *58*(7), 1019-1031. https://doi.org/10.1145/956863.956972

Link prediction group (LPG). (15 September 2017). http://www.linkprediction.org/index.php/link/resource/data

Liu B. (2011). *Web Data Mining. Data-Centric Systems and Applications*, Berlin: Springer.

Lu, L., Chen, D., Ren, X. L., Zhang, Q. M., Zhang, Y. C., & Zhou, T. (2016). Vital nodes identification in complex networks. *Physics Reports-Review Section of Physics Letters*, *650*, 1-63.

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, *89*, 228-249. https://doi.org/10.1016/j.knosys.2015.07.006

Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Test functions for optimization needs*, *101*.

Patel A. Kumar V., & Kumar Y. (2013). Golden Section Search Optimization Technique For Maximum Power Point Tracking, *International Journal of Engineering Research and Applications (IJERA), 3*(2), 1204-1209. https://doi.org/10.1109/IICPE.2012.6450384

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (2007). Golden Section Search in one dimension. In: Press, W. Flannery, B. Teukolsky, S. & Vetterling, W. (Eds.), *Numerical Recipes in C: The Art of Scientific*

Computing. (pp. 397-402), USA, Cambridge University Press. https://doi.org/10.1109/BMEI.2009.5304779

Rendón, E., Abundez, I., Arizmendi, A., & Quiroz, E. M. (2011). Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, *5*(1), 27-34.

Saida, I. B., Nadjet, K., & Omar, B. (2014). A new algorithm for data clustering based on cuckoo search optimization. In *Genetic and Evolutionary Computing* (pp. 55-64). Springer, Cham. https://doi.org/10.1007/978-3-319-01796-9_6

Salim, D., Aissa, K., & Adel, M. (2017). Efficiency Improvement in Induction Motor-Driven Solar Water Pumping System Using Golden Section Search Algorithm, Arab J. Sci. Eng. https://doi.org/10.1007/s13369-017-2972-6

Sharieh, A., Al Rawagepfeh, W., Mahafzah, M., & Al Dahamsheh, A. (2008). An algorithm for finding maximum independent set in a graph. *European Journal of Scientific Research*, *23*(4), 586-596.

Srinivas, V., & Mitra, P. (2016). *Link prediction in social networks: role of power law distribution*. Springer International Publishing.

Tsai, C. H., Kolibal, J., & Li, M. (2010). The golden section search algorithm for finding a good shape parameter for meshless collocation methods. *Engineering Analysis with Boundary Elements*, *34*(8), 738-746.