Mutation and memory mechanism for improving Glowworm Swarm Optimization algorithm

Conference Paper · January 2017

DOI: 10.1109/CCWC.2017.7868403

2 authors:

Atheer Bassel
Universiti Kebangsaan Malaysia
10 PUBLICATIONS **19** CITATIONS

SEE PROFILE

Md Jan Nordin
Universiti Kebangsaan Malaysia
130 PUBLICATIONS **1,152** CITATIONS

SEE PROFILE

2017 IEEE 7th Annual Computing and Communication Workshop and Conference (IEEE CCWC2017)
9 - 11 January 2017, Hotel Stratosphere, Las Vegas, USA

Mutation and Memory mechanism for improving Glowworm Swarm Optimization Algorithm

Atheer Bassel
Faculty of Information Science and Technology,
Universiti Kebangsaan Malaysia
atheerbassel@yahoo.com

Md Jan Nordin
Center for Artificial Intelligence Technology,
Universiti Kebangsaan Malaysia
jan@ukm.edu.my

***Abstract*—The Glowworm Swarm Optimization (GSO) is a population-based metaheuristic algorithm for optimization problem. Disadvantages of GSO are low accuracy, convergence speed and weakness in the capability of global search which need to be improved. Thus, Memory Mechanism and Mutation Glowworm Swarm Optimization (MMGSO) are proposed in this study to find a solution for this problem. The proposed method is examined on Unimodal and Multimodal benchmark functions to enhance the GSO algorithm of solution quality, convergence speed and robustness. The results of MMGSO are analyzed and compared with GSO to prove the efficiency of the proposed method.**

Keywords—Glowworm Swarm Optimization; Mutation; Memory less; Metaheuristic algorithm; Optimization

I. INTRODUCTION

Meta-heuristic algorithms are well-recognized approximate algorithms for their ability to resolve the all the problems of optimization with satisfactory outcomes, and thus it is defined as high level strategies for exploring the search space of these problems [1].

Numerous nature-inspired algorithms have been created lately as an attempt to resolve complex and difficult issues. In resolving the real-world problems, the traditional optimization methods are time consuming. Apart from that, the problems were also not effectively solved. Among the available algorithms are the swarm intelligence algorithms which are inspired by nature behaviors of living things [2].

Relatively new, swarm intelligence optimization are population-based algorithms that use stochastic search strategy. They are directly linked to evolutionary algorithms involving procedures that emulate natural evolution [3, 4]. Swarm intelligence algorithms are inspired by the collective behavior and developing intelligence existing in populations that are socially organized.

Optimization comprises a search process to find the optimum solution from a group of potential solutions according to certain criteria of performance. Optimization techniques are numerously available nowadays. These include swarm intelligence and evolutionary computation. As described by Engelbrecht [5], swarm intelligence is a technique of optimization following the processes of natural swarms via the simulation of natural their behavior. Examples of swarms include schools of fishes, flocks of birds, ant colonies and bacterial growth. The behavior of swarm is according to the interactions that take place between members of the swarm through the exchange of local information to accomplish the target. An example of swarm interaction is the search of the food source. Somehow, within the organization of the swarm, rather than employing the concept of centralization, all members of the swarm are engaged in the same way to reach the key objective.

Among the numerous swarm intelligence methodologies is Particle Swarm Optimization (PSO) [6]. The operation of PSO emulates flocks of bird looking for best sources of food. Here, the direction of bird movement is affected by its search experience which represented by local best, and by the global best which resulted from the other member's experience. Meanwhile, Ant Colony Optimization (ACO) [7] mimics ants' behavior when looking for the best shortest path between a source of food and the colony with the pheromone that they leave behind when they travel along the paths. On the other hand, Bee Colony Optimization (BCO)[8] emulates honeybee colonies' behavior of food foraging utilizing a mixture of local and global searches.

Glowworm Swarm Optimization (GSO) [9] follows the lighting worms or glowworms' behavior. These worms regulate their light emission and they use their light for purposes including attracting other worms during the season of breeding. The implementation of GSO is simple and only few parameters are required to be tuned [10, 11]. This makes GSO more applicable in numerous domains including hazard sensing in ubiquitous surroundings [10], robotics and mobile sensor networks [9], and clustering of data [12].

Almost all algorithms of swarm intelligence resolve the problem of optimization using a global solution which is simpler than locating various kinds of solutions. Nevertheless, GSO is unique in itself because it could conduct immediate search for various kinds of solutions. Thus, GSO can solve multimodal functions which according to Barrera and Coello [13] contain many peaks (local maxima) with the same or different objective values. They optimized the multimodal functions to locate all maxima according to certain constraints. Spaces of high dimension increase the count of peaks and this causes the evaluation of each function to necessitate lengthier implementation times in finding the best target peaks. In order

to solve multimodal functions, the swarm has to be able to divide itself into different groups for the sake of sharing extra local information for finding more peaks, the amount of individuals has to be increased [14].

However, GSO is slow regarding convergence. Thus, Zhou and Chen [15] introduced an artificial Glowworm Swarm Optimization algorithm grounded on cloud model. Meanwhile, an improvised version of GSO algorithm was introduced by Tang and Chen [16]. This version is called the parallel hybrid mutation Glowworm Swarm Optimization. Further, a GSO algorithm was employed in the work of Nelson Jayakumar and Venkatesh [17] in identifying the best solution for the problem of multiple-objective environmental economic dispatch.

This paper proposes the usage of mutation in the search process of GSO. This will increase the diversity of the swarm and assist the swarm in discovering the global optimum solutions. The mutation operation's modification increases the population's diversity by the mutation of selected solution. The operation of mutation allows those solutions to be improved. By way of mutation operation, some degree of scattering the solutions in the space of search is retained. This is to reduce the speed of convergence and to find new regions in search space. However, in some problems of optimization, some solutions become infeasible following the operations of mutation and migration. If such situation occurs, it is important that the solutions' feasibility is verified via the addition of other method to fix the solutions. The basic GSO algorithm, to our knowledge, has no memory. Further, in the iteration process, it does not take into account the fitness history of each individual, as well as GSO algorithm also has poor movement stability [18].

Section 2 of the paper explains the basic GSO while Section 3 presents the proposed mutation and memory for GSO. Section 4 introduces the experimental results, while the proposed method is concluded in the last section.

II. THE BASIC OF GSO ALGORITHM

As optimization algorithm, GSO belongs to the domain of swarm intelligence [5, 9]. This algorithm mimics the behavior glowworms or lighting worms in controlling their emission of light. These glowworms emit their light for different purposes such as the attraction of the other worms during the process of breeding. The aim of Swarm intelligence algorithms is to locate the global solution according to the objective function for the specified optimization problem. Further, it is easier to locate one global solution when compared to locating multiple solutions. As it has same or different objective function values, this algorithm is particularly crucial for an immediate search for various kinds of solutions. For this purpose, it is important that a swarm could divide itself into distinct groups.

Krishnan and Ghose [9] introduced GSO in 2005. In this algorithm, the swarm consists of N individuals known as glowworms. Hence, a glowworm *i* has a position $X_i(t)$ at time t in the function search space, emission of light known as the luciferin level $L_i(t)$, and a local decision range $rd_i(t)$. The luciferin level is linked with the objective value of the position of the individual according to the objective function J.

When a glowworm releases more light, this means that it has high luciferin level. This glowworm is closer to a real position and also has a high objective function value. Other glowworms (than its own) inside the local decision range are attracted by a glowworm that has higher luciferin level. The glowworm moves towards other neighboring glowworms when they have higher luciferin level and are within its local range. Eventually, almost all glowworms will be congregated at the multiple peak locations within the search area.

There are of four primary stages in GSO algorithm: initialization of glowworm, luciferin level update, movement of glowworm and update of the range of glowworm local decision. Initialization of glowworm involves the random deployment of *N* glowworms in the specified objective function search area. Hereby, the constants employed for the optimization are initialized, and each glowworm luciferin level is initialized too with the exact value (L_0). Local decision range (r_d) and radial sensor range (r_s) are also initialized with the exact initial value (r_0).

In GSO, the luciferin level update is regarded the most crucial step because the assessment of objective function is performed at the present glowworm position (X_i). For all members of swarm, the modification of the luciferin level is made based on the values of objective function. The equation below is used for the update process of luciferin level:

$$L_i(t) = (1 - \rho)\, L_i(t\text{-}1) + \gamma\, J(X_i(t)) \qquad (1)$$

Based on the above equation, $L_i(t)$ and $L_i(t\text{-}1)$ denote both of the new and old levels of luciferin for glowworm *i*, correspondingly. Meanwhile, ρ is the luciferin decay constant ($\rho \in (0,1)$), γ is the luciferin fraction of enhancement, and $J(X_i(t))$ denotes the objective function value for glowworm *i* at present glowworm position (X_i) at iteration t. Then, and all through the stage of movement, every glowworm makes attempt to attract the neighbor group $N_i(t)$ according to the levels of luciferin and the range of local decision (r_d) based on the rule below:

$$j \in N_i(t), \quad if\ d_{ij} < r_{di}\ and\ L_j(t) > L_i(t) \qquad (2)$$

Where *j* denotes one of the glowworms close to glowworm *i*, $N_i(t)$ represents the neighbor group, d_{ij} denotes the Euclidean distance between glowworm *i* and glowworm, $j < r_{di}(t)$ represents the range of local decision for glowworm *i*, and $L_j(t)$ and $L_i(t)$ denote the levels of luciferin for glowworm *j* and *i*, correspondingly. Then, two operations are used to identify the actual chosen neighbor. These operations include the operation of probability calculation for finding out the direction of movement toward the higher luciferin neighbor. The equation below will be applied:

$$prob_{ij} = \frac{Lj(t) - Li(t)}{\sum_{k \in Ni(t)} L_k(t) - Li(t)} \qquad (3)$$

Based on the equation above, *j* represents one of the neighbor group $N_i(t)$ of glowworm *i*. In the next step, glowworm *i* picks out a glowworm from the neighbor group employing the

method of roulette wheel. Here, glowworm that has higher probability is more likely to be chosen from the neighbor group. Finally, in the glowworm movement's stage, modification is made to the position of the glowworm according to the chosen neighbor position employing the equation below:

$$X_i(t) = X_i(t-1) + s\,\frac{X_i(t)-X_i(t)}{\delta_{ij}} \qquad (4)$$

Based on the above, $X_i(t)$ and $X_i\ (t\ -1)$ denote the new position and the preceding position for the glowworm i, correspondingly, (s) denotes a step size constant, and (δ_{ij}) denotes the Euclidean Distance between glowworm i and glowworm j.

The update of the range of local decision entails the final step of GSO. Here, the range of local decision (rd_i) is updated. This makes the glowworm more flexible to create the neighbor group within the forthcoming iteration. The below formula is employed to update (rd_i) in the forthcoming iteration:

$$rd_i\ (t+1) = min\ \{r_s,\ max\ [0,\ rd_i\ (t-\ 1) + \beta\ (n_t - |\ \mathrm{Ni}\ (t)\ |)]\} \qquad (5)$$

Based on the equation above, $rd_i(t)$ and $rd_i\ (t-\ 1)$ denote the new and the previous ranges of local decision for glowworm i correspondingly, r_s represents the constant radial sensor range, β denotes a model constant, n_t represents a constant parameter employed for controlling the neighbor count, and $|N_i\ (t)|$ represents the real number of neighbors.

Figure 1 provides the summation of the basic GSO algorithm in terms of computational procedure.

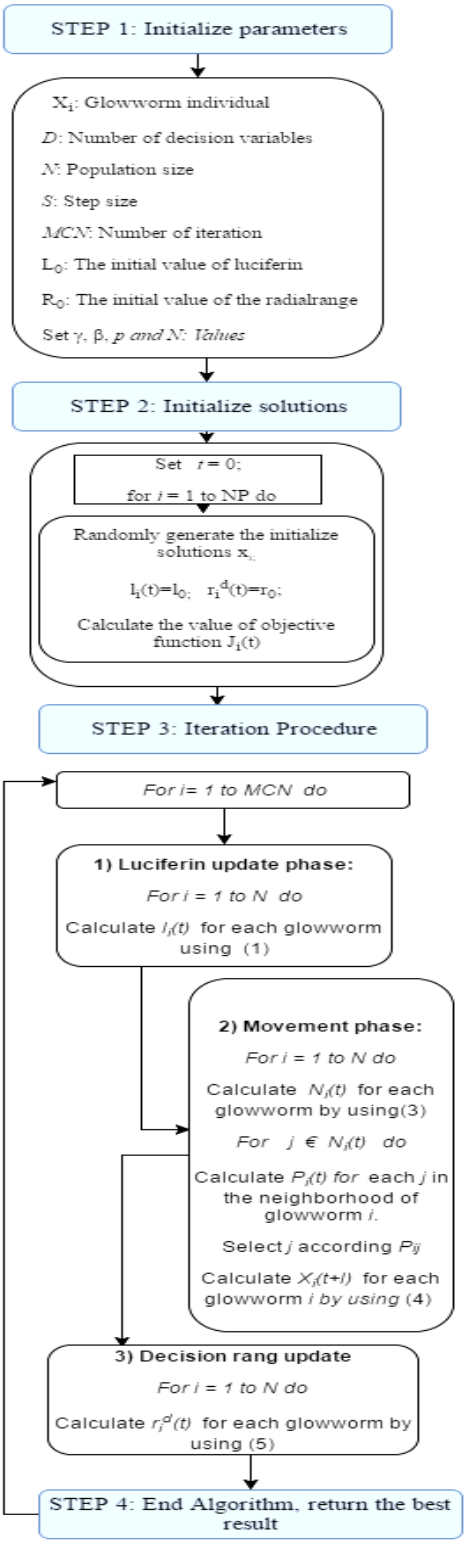

Fig. 1. The basic of GSO

III. Proposed Method

In this paper, first the GSO algorithm has some weakness in the global and slow convergence and low accuracy. Due to the

diversity of the search space for different optimization problems, we propose a model fusion approach, three modules are proposed the first one GSO module the swarm is randomly initialized with the GSO operator being evaluate the fitness and employed to update the swarm. Memory module, we check the new solution inside the memory or not. If the new solution no inside the memory, calculate the fitness and adding the new solution and fitness inside the memory. In the end, mutation module used to update one randomly selected Glowworms. Finally, stopping criterion is satisfied, the algorithm will stop.

A. Mutation

As illustrated in the literature on particle swarm optimization for the rudimentary PSO, every particle flies to the past best particle P_{best} and also to the global best particle g_{best}. As generations increase in number, particles become identical to P_{best} and g_{best}. Worded differently, the value of $P_{best} - X$ and $g_{best} - X$ become small. The moment these best particles become stuck in local optima, every particle in the existing swarm will speedily congregate to the local minima. Here, a number of mutation techniques are used for forming mutant particles rather than g_{best}, X or P_{best}; it is possible that in this case, the mutation could push the trapped particles frontward[19].

PSO can jump out of local optima with the help of numerous mutation operators. Somehow, it should be noted that for some problems, a mutation operator may be more effective than others, while for some other problems, the exact mutation operator may be ineffective. It can also be the same situation at different stages of the optimization process. This means that the best mutation outcomes are not attainable with the use of just one mutation operator. Therefore, for optimal performance, a number of mutation operators may have to be used at different stages. As such, a mutation operator is created in this study. This mutation operator has the ability to adaptively choose the most fitting mutation operator to cater to different problems. With the use of different mutation operators, PSO can be assisted in jumping out of local optima.

This subsection proposes an additional mutation strategy to further improve GSO convergence. Mutation is selected due to several reasons. For instance, the first part technologies such as re-initialization[20] and the jump strategy[21-23] are still basically a variant of mutation. Further, as shown the second part in the literature, a fitting mutation operator could assist swarm/ population in locating global optimum [24, 25].

The mutation in the proposed method expressed by targeting number of solutions selected randomly. Thereafter, the taken population will be mutated and evaluated. Finally, the mutated solutions will take the place of worse solutions in the population of GSO algorithm.

B. Memory less

Based on the basic GSO, it is obvious that GSO does not keep any search space information during the search process. Also, GSO algorithm does not keep the history of any situation for each Glowworm. This causes the Glowworm to move irrespective of its prior situation, and it is also possible that the glowworm will miss their history information at the end. Thanks to memory inclusion the solution would be evaluated if it is not met in previous search iterations. Thereafter, new solution with its objective value will be inserted into the memory to prevent cyclic evaluation. Also, new absorption parameter should be added, while the parameter should be changed with time and the situation history recalled.

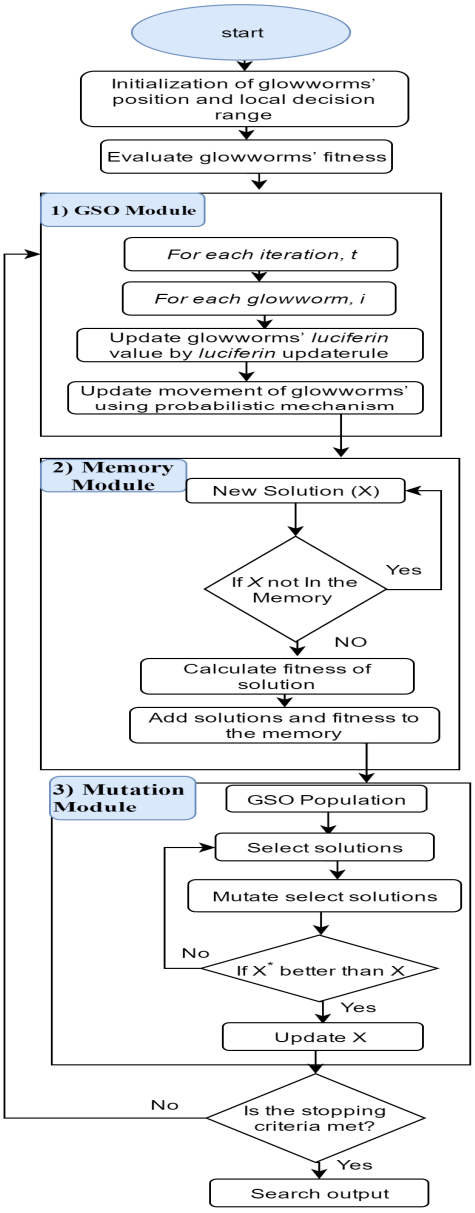

Fig. 2. MMGSO

IV. EXPERIMENTS AND DISCUSSION

This section highlights the completed experiment. In particular, the performance of the MMGSO algorithm proposed alongside its variants for certain functions of analytical

benchmark are explained. The experiments are all performed in a Windows 7 professional system with Intel core i3, 2.67 GHz, 2G RAM while the execution of the codes conducted by Matlab 2013b. To assess and test the performance of the proposed optimization method, we considered mathematical functions which are commonly known in the optimization field as benchmark test functions. These equations can be categorized into two groups which are unimodal, and multimodal, and thus we organized the first four functions f1-f4 to be unimodal and the last four f5-f8 are multimodal as shown in Table 1. The simulation parameters of these functions are set as follow: 10 as the problem dimension, 5000 which is the number of iterations, and 12 times as number of runs. To prove the effectiveness of the proposed mutation and memory-less to GSO, the experimental results were applied in three levels, i.e., Mutation GSO alone, Memory GSO and MMGSO. The experimental results are conducted based on number of metrics which are minimum cost (min), maximum cost (max), average cost (mean), and standard deviation (dev). Tables 3, present the experimental results of the proposed three levels model.

Table 1. The parameters are set at bellow in the experiments

P	B	Γ	L_0	n_t	r_s
0.8	0.6	0.05	0.5	0.9	0.5

Thus Figure 3 shows the result for 12 runs of different types of functions based on number of iterations and the best fitness.

V. Conclusion

This paper focused on the weakness points in the search process of GSO. Three types of limitations have been determined in GSO which are low accuracy, convergence speed and weakness in the capability of global search. To tackle these problems, we improved the basic GSO via adopting mutation and memory mechanisms. For the purpose of showing the effect of each mechanism, we presented the results of each of them separately, and then we showed the results of the hybrid method. The proposed method was experimented on benchmark of test functions that include Uni-modal and Multi-modal. The results show the effectiveness of the proposed method over the basic GSO.

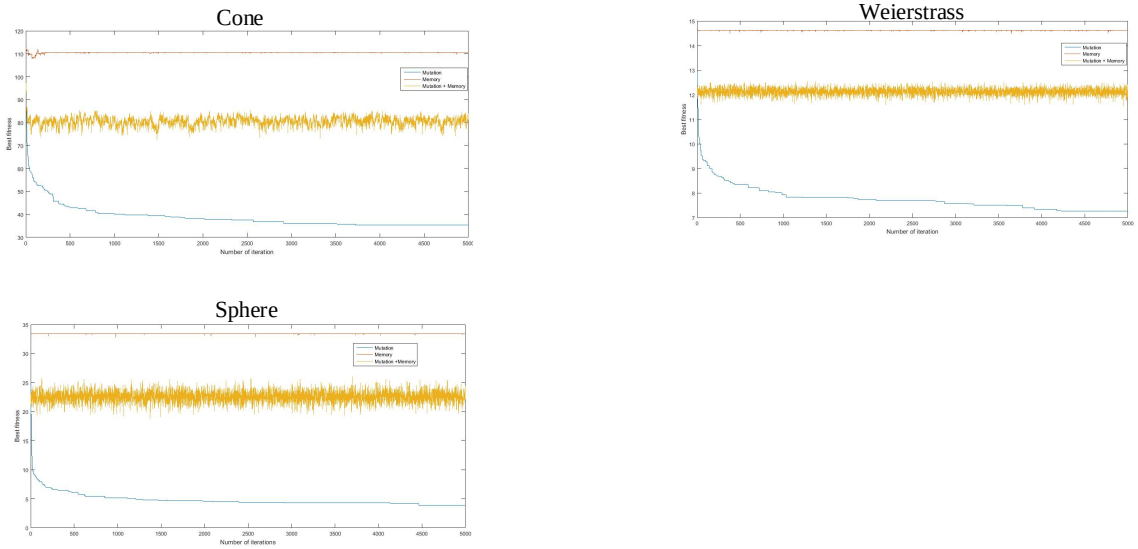

Fig. 3. Comparison among the investigated methods over uni-modal and multi modal problems

Table 2. Benchmark functions

Property	Peak function	Formula	Search space
Uni-modal	Cone	$f(x)=\sqrt{\sum_{i=1}^{n} x_i^2}$	[-100,100]
Uni-modal	Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	[-5.12, 5.12]
Uni-modal	Schwefel 2.22	$f(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10, 10]
Uni-modal	Quadric	$f(x)=\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	[-1.28, 1.28]
Multi-modal	Rastrigin	$f(x)=\sum_{i=1}^{n} (x_i^2 - 10 \cos 2\pi x_i + 10)$	[-5.12, 5.12]
Multi-modal	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	[-600, 600]
Multi-modal	Ackley	$f(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i + 20 + exp$	[-32, 32]
Multi-modal	Weierstrass	$f(x)=\sum_{i=1}^{n} (\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (xi + 0.5))]) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$	[-0.5, 0.5]

Table.3. Comparison between Mutation GSO, Memory GSO, Mutation and Memory GSO

Test Function	Mutation GSO				Memory GSO				Mutation &Memory GSO			
	mean	Std	min	Max	mean	Std	min	max	mean	std	min	max
Cone	37.047	6.631	34.499	99.480	104.804	0.587	96.191	104.932	76.940	5.982	43.239	94.201
Sphere	3.751	1.509	3.144	28.135	26.712	0.000	26.712	26.712	21.084	1.441	11.496	21.681
Schwefel2.22	10.805	1.154	10.415	45.916	62.934	0.000	62.934	62.934	34.478	7.678	17.376	51.865
Quadric	0.262	0.108	0.162	1.667	3.154	0.006	2.744	3.154	1.517	0.241	0.522	2.354
Rastrigin	34.670	5.490	33.705	83.483	119.504	0.000	119.504	119.504	94.009	6.609	59.428	103.743
Griewank	13.732	5.305	12.270	99.359	114.312	0.000	114.312	114.312	81.994	12.351	17.244	108.284
Ackley	13.598	0.898	13.114	19.220	19.028	0.278	15.028	19.275	15.649	0.076	13.190	15.653
weierstrass	7.916	0.664	6.965	12.988	15.443	0.033	14.067	15.445	12.419	0.612	9.496	13.494

REFERENCES

[1] Blum and A. Roli, "Hybrid metaheuristics: an introduction," in Hybrid Metaheuristics: Springer, 2008, pp. 1-30.

[2] H. Haklı and H. Uğuz, "A novel particle swarm optimization algorithm with Levy flight," Applied Soft Computing, vol. 23, pp. 333-345, 2014.

[3] T. Bäck, D. Fogel, and Z. Michalewicz, "Handbook of evolutionary computation," Release, vol. 97, no. 1, p. B1, 1997.

[4] H.-P. P. Schwefel, Evolution and optimum seeking: the sixth generation. John Wiley & Sons, Inc., 1993.

[5] A. P. Engelbrecht, Computational intelligence: an introduction. John Wiley & Sons, 2007.

[6] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, 1997, vol. 5, pp. 4104-4108: IEEE.

[7] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Transactions on evolutionary computation, vol. 1, no. 1, pp. 53-66, 1997.

[8] L.-P. Wong, C. Y. Puan, M. Y. H. Low, and C. S. Chong, "Bee colony optimization algorithm with big valley landscape exploitation for job shop scheduling problems," in 2008 Winter Simulation Conference, 2008, pp. 2050-2058: IEEE.

[9] K. Krishnanand and D. Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," in Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005., 2005, pp. 84-91: IEEE.

[10] K. Krishnanand and D. Ghose, "Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations," Robotics and Autonomous Systems, vol. 56, no. 7, pp. 549-569, 2008.

[11] K. Krishnanand and D. Ghose, "Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions," Swarm intelligence, vol. 3, no. 2, pp. 87-124, 2009.

[12] I. Aljarah and S. A. Ludwig, "A new clustering approach based on glowworm swarm optimization," in 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 2642-2649: IEEE.

[13] J. Barrera and C. A. C. Coello, "A review of particle swarm optimization methods used for multimodal optimization," in Innovations in swarm intelligence: Springer, 2009, pp. 9-37.

[14] I. Aljarah and S. A. Ludwig, "A Scalable MapReduce-enabled Glowworm Swarm Optimization Approach for High Dimensional Multimodal Functions," International Journal of Swarm Intelligence Research (IJSIR), vol. 7, no. 1, pp. 32-54, 2016.

[15] Q. Zhou, Y. Zhou, and X. Chen, "Cloud model glowworm swarm optimization algorithm for functions optimization," in International Conference on Intelligent Computing, 2013, pp. 189-197: Springer.

[16] Z. Tang, Y. Zhou, and X. Chen, "An improved glowworm swarm optimization algorithm based on parallel hybrid mutation," in International Conference on Intelligent Computing, 2013, pp. 198-206: Springer.

[17] D. N. Jayakumar and P. Venkatesh, "Glowworm swarm optimization algorithm with topsis for solving multiple objective environmental economic dispatch problem," Applied Soft Computing, vol. 23, pp. 375-386, 2014.

[18] H. Cui, J. Feng, J. Guo, and T. Wang, "A novel single multiplicative neuron model trained by an improved glowworm swarm optimization algorithm for time series prediction," Knowledge-Based Systems, vol. 88, pp. 195-209, 2015.

[19] Y. Zhang, D.-W. Gong, X.-Y. Sun, and N. Geng, "Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis," Soft Computing, vol. 18, no. 7, pp. 1337-1352, 2014.

[20] H. Wang, I. Moon, S. Yang, and D. Wang, "A memetic particle swarm optimization algorithm for multimodal optimization problems," Information Sciences, vol. 197, pp. 38-52, 2012.

[21] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or Cauchy jumps," in 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 3285-3291: IEEE.

[22] T. Blackwell, "A study of collapse in bare bones particle swarm optimization," IEEE Transactions on Evolutionary Computation, vol. 16, no. 3, pp. 354-372, 2012.

[23] M. M. Al-Rifaie and T. Blackwell, "Bare bones particle swarms with jumps," in International Conference on Swarm Intelligence, 2012, pp. 49-60: Springer.

[24] M. Hu, T. Wu, and J. D. Weir, "An intelligent augmentation of particle swarm optimization with multiple adaptive methods," Information Sciences, vol. 213, pp. 68-83, 2012.

[25] H. Gao and W. Xu, "A new particle swarm algorithm and its globally convergent modifications," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 41, no. 5, pp. 1334-1351, 2011.