Improved Simplified Glowworm Swarm Optimization Algorithm

Srinivasa Acharya[1], Dr.D.Vijaya Kumar[2]
[1]Assistant Professor, EEE Department, AITAM Tekkali
[2]Professor, EEE Department, AITAM Tekkali

Abstract- *Aimed at the poor optimizing ability and the low accuracy of the glowworm swarm optimization algorithm (GSO), a simplified glowworm swarm optimization algorithm (SGSO) was put forward in this paper, which omitted the phases of seeking dynamic decision domain and movement probability calculation, and meanwhile simplified the location updating process. Moreover, elitism was introduced to improve the capacity of searching optimal solution. It was applied to the unimodal and multimodal benchmark function optimization problems. The improved SGSO algorithm is compared with the basic GSO and other swarm intelligent optimization algorithms to demonstrate the performance. Experimental results showed that SGSO improves not only the precision but also the efficiency in function optimization.*

Keywords- glowworm swarm optimization; swarm intelligence; function optimization

I. INTRODUCTION

META-HEURISTIC algorithms provide a new perspective for solving complex problems by mimicking the biological behaviors and nature phenomenon, with the characteristics of high robustness, low complexities, excellent efficiency and superb performance, compensate the lack of searching and calculation for finite solutions and high complexity in traditional algorithms.

As a significant branch of meta-heuristic research, swarm intelligence algorithms, which inspired by the behavior of birds, fish, ants, and bee colonies and so on, is applied to search global optimum of many problems. Besides the characteristics of the meta-heuristic algorithms, swarm intelligent algorithms have the advantages of easy operation and good parallel architecture. In recent years, novel swarm intelligent algorithms for optimization have sprung up continually and have driven a high tide of researches on swarm intelligence. For example, particle swarm optimization algorithm (PSO), proposed by Kennedy J, Eberhard R.C. [1] in 1995, imitated the behavior of birds; bacterial foraging optimization algorithm (BFO) [2], introduced in 2002, simulated the foraging of bacteria; artificial bee colony algorithm (ABC) [3], introduced in 2005, mimicked the behavior of bee colonies for searching honey. Swarm intelligence optimization algorithms are widely applied in many scientific field including function optimization and combination optimization [4- 6], NP-hard problems [7, 8], data mining [9- 11], engineering and process [12, 13], biotechnology [14] and other fields.

Glowworm swarm optimization algorithm (GSO) is a nature inspired heuristic intelligent algorithm, proposed by Krishnan and K.N. and Ghose D. in 2005 [15], which simulated behavior of glowworm group in moving by using luciferin to attract other glowworms around or foraging. The greater value of luciferin, the brighter of the glowworm, the more attractive will be.

Glowworm swarm optimization algorithm has been applied to many fields, such as multimodal function and combination optimization [16, 17], robotics applications [18-20], and wireless sensor networks [21, 22]. Also, it is widely used in some NP-Hard problems like TSP [23] and 0-1 knapsack issues [24]. Glowworm swarm optimization algorithm has some shortcomings, such as low accuracy in later iterations, slow convergence speed and easy to be trapped into local optimal solutions.

A simplified glowworm swarm optimization algorithm (SGSO) is proposed to improve the performance of the original GSO algorithm. Comparison shows good performance in the field of function optimization problems with the basic GSO, which embodies the ability of fast convergence speed and strong searching ability in contrast to PSO, BFO, ABC [25] and the fruit fly optimization algorithm (FOA) [26, 27] which is a novel swarm intelligent algorithm proposed by Pan in 2011, mimicking the foraging behavior of fruit flies for searching global optimum.

The rest of this paper is organized as follows. Section II introduces the basic concepts and principles of glowworm swarm intelligent optimization algorithm. Section III provides the simplified glowworm swarm optimization algorithm and corresponding principles, location update, elitism, boundary control, procedure of SGSO and computation complexity. Results from experiments are described in Section IV, where we test two groups of experiments for SGSO. One is the comparison between the basic GSO and SGSO in different dimensions, and the other is the comparison among other intelligent algorithms in 30 dimensions. Finally, section V concludes the paper and illustrates the future research.

II. GLOWWORM SWARM OPTIMIAZTION ALGORITHM

In GSO algorithm, the glowworm is more attractive when the luciferin value is greater, which guides other glowworms to move towards it. Each glowworm has its dynamic decision space, which contains glowworms with both values of luciferin higher than itself and distance within its dynamic decision radius. Glowworm updates its location to a glowworm in its dynamic decision space in the light of probability, and then, renews its decision space radius.

A. Procedure of GSO

The value of luciferin is related to both the value of luciferin in the former iteration and objective function in the current iteration.

Let

$x_i(t)$represents the location of glowworm *I* iteration, $J(x_i(t))$ denotes the value of objective function, which is transferred to the value of luciferin denotes by $l_i(t)$ follows as:

$$l_i(t)=(1-\rho)l_i(t-1)+\gamma J(x_i(t)) \quad (1)$$

where

ρ and γ Are the luciferin decay constant and enhancement constant respectively. Both of them range from 0 to 1. Each glowworm has a dynamic decision space, which contains its neighbors with higher luciferin values than its own value and The distance between them within the decision space radius. $N_i(t)$ is the set of neighborhood of glowworm *i* at the *t* iteration, given by (2). $pij(t)$ is deemed to be the probability of glowworm *i* moving toward a neighbor *j* in the *t* iteration, calculated by (3).

$$N_i(t) = \{j : d_{ij}(t) < r_d^i(t) ; l_i(t) < l_j(t)\} \quad (2)$$

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \quad (3)$$

where $k \in N_i(t)$ $r_d^i(t)$denotes the dynamic decision space radius of glowworm *i* in the *t* iteration, and $lj(t)$ is the luciferin value of glowworm *j* after the phase of probabilistic mechanism in the *t* iteration. Each glowworm updates its location according to (4).

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right) \quad (4)$$

where *s* is the step-size, represents the Euclidean norm operator. The radius of each glowworm dynamic decision space not only depends on the current radius of dynamic decision space, but also associates with the radial range of the luciferin sensor deemed by r_s . The update rule of each glowworm dynamic decision space radius is given by:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\} \quad (5)$$

where β denotes the dynamic decision space parameter, t n is a control parameter for neighbors in the space. Ni(t) is the number of neighbors in the dynamic decision space.

III. THE SIMPLIFIED GLOWWORM SWARM OPTIMIZATON ALGORITHM

A. Principle of SGSO

In GSO, the running time is very long because of the complex computing of decision space and probabilistic chosen mechanism. What's more, the location update of glowworm is based on the dynamic decision space, which concerned with luciferin. Hence, SGSO is proposed in this paper, the location update of glowworm is simplified only based on the luciferin, which reduces the running time.

Meanwhile, the dynamic decision space transferres its local search to global search using elitism, which enhances the efficiency and the searching ability. The procedure of SGS adopts new policies including luciferin update, location Update and. elitism

B. Location Update

As noted above, s location updating policy depends on the dynamic decision spaces consisting of the neighbors, which Leads to the local optimum as well. Furthermore, location update also relates to both the radius of dynamic decision space and radial range of the luciferin sensor, which takes much time. We modify the movement of glowworm individuals and simplify the movement of glowworms by adopting the thought of probabilistic selection in simulated annealing algorithm [28]. In other words, glowworms move towards to the best glowworm with a probability, otherwise moves to another direction.

Each glowworm moves to the best location, in which luciferin value is the minimum in the group. The location update of each glowworm follows (6).

$$x_i(t+1) = wx_i(t) + \alpha(x_{best}(t) - x_i(t)) \quad (6)$$

where α is the speed parameter. *w* is deemed to be the inertia weight, $x_{best}(t)$ is the best location in the *t* iteration.

The calculation of α is given by:

$$\alpha_i = \begin{cases} \alpha_{i-1} r_1 & if \;\; r_2 < 0.5 \\ \alpha_{i-1} \dfrac{J(x_j(t)) - \min J(x_j(t))}{\max J(x_j(t)) - \min J(x_j(t))} & otherwise \end{cases} \quad (7)$$

where $r1$ and $r2$ are random number generated in each iteration. α is a constant variable in the inception of the algorithms.

In each iteration, α relates to the former value of itself in the iteration in advance. When $r2$ is less than 0.5, α is connected with a random number changing in each iteration.

Otherwise, it relates to the best and worst fitness values in the last iteration see (7).

As noted above, new location update can ensure the location update which helps glowworms jump out of the local optimum superiorly. Furthermore, it reduces the calculation times which promotes the moving speeding of glow warms.

For the purpose of promoting the searching ability of the optimum, each glowworm moves to the best location in all iterations. The worst *m* glowworm locations with the largest value of luciferin are instead of the best *m* glowworm locations possessing the smallest value of luciferin in each iteration

D. Boundary Control

Locations of glowworms will go beyond the domain of the problems, so we need to limit them in the domain problems to ensure the legality of the locations. Hence, when the location exceeds the upper bound, the value is assigned to the upper bound value, similarly when the location value low than the lower bound, the value is assigned to the lower bound value.

E. Procedure of SGSO

Step1 : Initialize parameters including the scale of glow worm group *n*, the dimension *d*, the maximal iteration $iter_{\max}$,$^{\rho}$,$^{\gamma}$, the initial luciferin value $l_i(0)$,w,$^{\alpha}$,the boundary value of objective function *upbnd* and *lwbnd*, etc.

Step2 : Luciferin update: transform objective function value to luciferin value using (1).

Step3 : Update the location of each glowworm based on (6) and (7).

Step4 : Calculate the value of objective function after location update, replace the *m* worst locations using the *m* best locations to accomplish the elitism mechanism.

Step5 : Compare the value of the optimum and objective function, if objective function value better than the optimal exists, update the optimal value using the objective function value.

Step6 : If the t iteration is equal to $iter_{\max}$, the algorithm come to the end, else t=t+1, go to **Step2**.

III. EVALUATION AND ANALYSIS OF EXPERIMENTAL RESULTS

The algorithms are coded in Matlab7.13 and experiments were executed on Pentium dual-core processor 3.10 GHz PC with 4G RAM.

IV.

Two experiments are tested in this section. Comparison between SGSO and the basic GSO to testify the performance of SGSO for benchmark functions in 10, 20, 30 dimensions, respectively. After that, we compare SGSO with some famous and recent swarm intelligent algorithms, such as PSO, BFO.

A. Parameter Discussion

In all algorithms, *upbnd* and *lwbnd* are equal to the upper and lower bounds of the objective function domain respectively. 9 benchmark functions used for experiments are shown in Table I. Each benchmark function has the optimum of 0. Functions *f*1 - *f*4, *f*9 are unimodal functions, while the others are multimodal functions. The parameters setting for the swarm intelligent algorithms see Table II.

B. Comparison between GSO and SGSO

We compare SGSO with the basic GSO to testify the Performance. The optimal value, average value and average running time are calculated after 300 independent experiments with maximal iteration 300. ABC and FOA. Taking the best value, mean value and running time into consideration for the experiments.

Performances of running in 30 dimensions and 300 iterations are shown respectively in Fig.1. Due to the objective function values close to 0, which cannot distinguish clearly, logs base e are deal with the vertical function values. Moreover, considering on either the convergence speed or accuracy, SGSO performs demonstrably superior to GSO. The four stages and probabilistic mechanism in dynamic decision space of GSO takes a long time to give rise to the longest running time and the worst accuracy for solutions; SGSO omits the calculation for dynamic decision space and probabilistic mechanism and meanwhile adopts elitism with little complexity, which lead to the high accuracy and searching speed of the optimum.

From Table III to Table V, we can see that SGSO shows good performance in both precision and efficiency while the basic GSO performs worse. From the perspective of values of objective functions, SGSO shows good performance in both running time and the solutions. GSO changes obviously in different dimensions, but the running time remains approximately in different dimensions; SGSO reaches the real optimum for *f*2 to *f*8 regardless of dimensions. Although dimensions changing from 10 to 30, the optimum of SGSO varies tiny, as well as the running time.

TABLE I. BENCHMARK FUNCTIONS

ID	Function equation	Domain
f_1	$\sum_{i=1}^{d-1}(100(x_{i+1}-x_i^2)^2 + (x_i-1)^2)$	±50
f_2	$\sum_{i=1}^{d} x_i^2$	±5.12
f_3	$\sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	±100
f_4	$10^6 x_1^2 + \sum_{i=2}^{d} x_i^2$	±100
f_5	$\sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i) + 10)$	±5.12
f_6	$20 + e - 20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$	±32
f_7	$\frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos(\frac{x_i}{i}) + 1$	±5.12
f_8	$\sum_{i=1}^{d}(x_i^2+x_{i+1}^2)^{0.25}[\sin(50(x_i^2+x_{i+1}^2)^{0.1})+1]$	±100
f_9	$\sum_{i=1}^{d} i x_i^4 + rand[0,1)$	±1.28

TABLE II. PARAMETERS OF ALGORITHMS

Algorithm	Parameters
PSO	$n = 50$, $w = 0.8$, $c_1=2$, $c_2=2$
BFO	$n = 20$, $n_c = 10$, $n_s = 5$, $n_r = 2$, $c_r = 0.025$
GSO	$n = 20$, $\rho = 0.4$, $\gamma = 0.6$, $l_i(0) = 4$, $n_t = 4$, $r_d = 50$, $r_s = 50$
FOA	$n = 20$
SGSO	$w=0.8$, $\alpha = 0.4$, $m=3$

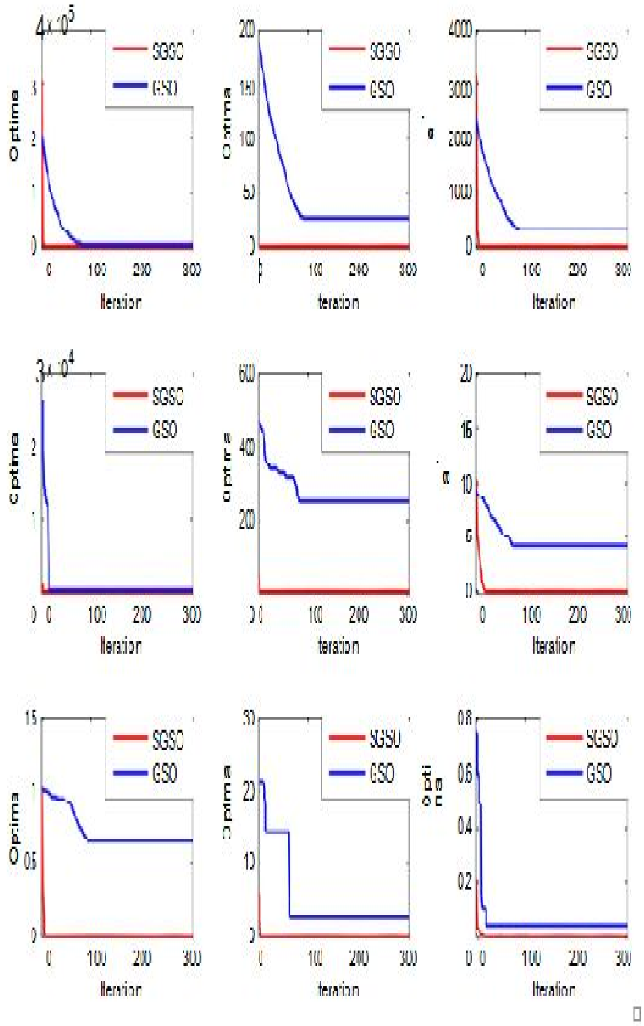

Fig. 1. Comparison of searching curves for $f1$ to $f9$ between GSO and SGSO

TABLE III. COMPARISON OF GSO AND SGSO IN 10 DIMENSIONS

f	Algorithm	Best	Mean	Running time(s)
f_1	GSO	1.4456e+003	1.4982e+003	0.8680
	SGSO	0.0204	0.0361	0.0575
f_2	GSO	8.3816	10.3736	0.8597
	SGSO	0	1.1254e-310	0.0555
f_3	GSO	31.8760	53.8612	0.8674
	SGSO	0	3.2784e-315	0.0761
f_4	GSO	44.4378	48.3262	0.7421
	SGSO	0	7.1634e-299	0.0586
f_5	GSO	38.4087	41.6258	0.7428
	SGSO	0	0.0017	0.0508
f_6	GSO	3.9220	4.2801	0.8174
	SGSO	-8.8818e-016	-8.8818e-016	0.1063
f_7	GSO	0.2528	0.6919	1.3480
	SGSO	0	1.7718e-008	0.2160
$f8$	GSO	0.1298	0.3982	1.3463
	SGSO	0	1.0495e-079	0.2481
$f9$	GSO	0.0158	0.0454	1.3201
	SGSO	3.1842e-005	2.8543e-004	0.1842

TABLE IV. COMPARISON OF GSO AND SGSO IN 20 DIMENSIONS

f	Algorithm	Best	Mean	Running time(s)
f_1	GSO	3.7212+003	4.0139+003	0.9650
	SGSO	1.3793	1.4021	0.0539
f_2	GSO	11.7586	32.8547	1.0261
	SGSO	0	2.0543-309	0.0543
f_3	GSO	140.7542	228.8223	0.9390
	SGSO	0	3.2527e-311	0.0992
f_4	GSO	36.1046	44.1825	0.9009
f5				
f6	SGSO	0	4.2742e-310	0.0603
	SGSO	-8.8818e-016	-8.8818e-016	0.1027
f_7	GSO	0.5904	0.8217	1.3217
	SGSO	0	2.2561e-009	0.2198
f_8	GSO	0.5062	1.2323	1.3945
	SGSO	0	4.4237e-078	0.2171
f_9	GSO	0.0298	0.0445	1.3001
	SGSO	1.2016e-005	1.4392e-004	0.1765

TABLE V. COMPARISON OF GSO AND SGSO IN 30 DIMENSIONS

f	Algorithm	Best	Mean	Running time(s)
f_1	GSO	3.9215e+003	9.0725+003	0.9252
	SGSO	8.2845	9.8914	0.0575
f_2	GSO	27.1790	35.7654	0.9617
	SGSO	0	8.7145e-310	0.0572
f_3	GSO	327.2692	452.2723	0.9639
	SGSO	0	8.1278e-310	0.1451
f_4	GSO	58.9423	71.2306	0.9821
	SGSO	0	4.2748e-310	0.0627
f_5	GSO	244.2684	248.3783	0.8823
	SGSO	0	0.0071	0.0574
f_6	GSO	5.7349	7.2853	0.9696
	SGSO	-8.8818e-016	-8.8818e-016	0.1173
f_7	GSO	0.5985	0.8711	1.1627
	SGSO	0	2.9032e-008	0.2231
f_8	GSO	2.0431	8.3309	1.3979
	SGSO	0	6.4709e-078	0.2679
f_9	GSO	0.0423	0.0692	1.3789
	SGSO	8.0662e-006	4.3982e-005	0.1726

C. Comparison with Other Swarm Intelligent Algorithms

We compare SGSO with PSO, BFO, ABC and FOA for the 9 benchmark functions. Due to the convergence speed is very slow in FOA, here we set the maximal iteration to 1000 and run 300 times to observe the results for *f*1 to *f*9 benchmark functions in 30 dimensions.

Table VI shows the best and mean values of the algorithms, as well as the running time for benchmark functions *f*1 to *f*9 in 30 dimensions. As we know, PSO is an excellent algorithm which applies to almost every scientific field. Results show that PSO performs worse than FOA and SGSO from Table VI in respect of both precision and running time; BFO performs seldom well for the values, nevertheless, the running time is the highest because of the three behavior in foraging; ABC takes less time than BFO but still a little longer than others, the optimum is better than BFO, but worse than the others; FOA takes the lowest running time on account of its simple mechanism. Although FOA performs better than PSO, BFO and ABC taking the least running time, as well as the smallest deviation by comparison with others, the values of the best and mean are larger than SGSO; SGSO performs well for the best, mean values and the running time in contrast to other algorithms. With the tiny deviation, SGSO owns highest robustness. And furthermore, SGSO has the low time cost comparing with others in functions *f*1 to *f*9 although the running time is a bit longer than FOA, but far less than the other algorithms.

Table VI. COMPPARISION WITH OTHER ALGORITHMS IN 30 DIMENSIONS

f	Algorithm	Best	Mean	Running time(s)
f_1	PSO	31.0285	49.4807	0.7931
	BFO	43.4631	1.4331e+003	7.1552
	ABC	27.7962	28.6873	2.9437
	FOA	28.2173	28.7338	0.2338
	SGSO	8.0789	16.2837	0.3779
f_2	PSO	0.2219	0.6284	0.7513
	BFO	27.8451	29.8431	5.9817
	ABC	3.3128	4.8721	2.5912
	FOA	1.0150e-004	2.0982e-004	0.2098
	SGSO	0	1.0905e-007	0.3651
f_3	PSO	2.0701	15.9722	1.2205
	BFO	0.5419	108.6321	14.1602
	ABC	0.0203	0.0497	7.1757
	FOA	0.0012	0.0015	0.4081
	SGSO	0	8.3895e-007	0.6003
f_4	PSO	4.5567	20.9914	0.8457
	BFO	3.4542e-005	40.0019	7.6265
	ABC	0.0032	0.0038	2.7734
	FOA	2.6782	2.6976	0.2697
	SGSO	0	1.2861e-004	0.3982
f_5	PSO	49.2173	77.6424	0.8698
	BFO	102.8421	137.9542	7.9631
	ABC	0.1692	0.4624	2.6402
	FOA	0.0210	0.0231	0.2627
	SGSO	0	1.2536e-004	0.3740
f_6	PSO	0.8998	1.5890	1.2047
	BFO	-5.2288e-004	2.5243	14.6496
	ABC	2.0182	3.1521	7.4484
	FOA	0.0072	0.0079	0.3202
	SGSO	-8.8818e-016	-8.8818e-016	0.5754
f_7	PSO	0	0	1.1103
	BFO	0.0414	2.0124	39.6883
	ABC	0	1.0218	2.6551
	FOA	7.2976e-006	7.6892e-006	0.4339
	SGSO	0	0	0.7038
f_8	PSO	0.0754	0.1358	1.6925
	BFO	67.5995	79.4626	38.9332
	ABC	17.6547	27.3869	3.5812
	FOA	2.0480	2.3768	0.6536
	SGSO	0	0.0297	0.5472
f_9	PSO	8.3267e-005	1.1154e-004	1.0839
	BFO	7.4369e-004	0.0025	12.2972
	ABC	5.7077e-006	0.0001	2.6599
	FOA	1.2551e-007	1.0227e-005	0.3473
	SGSO	0	5.0498e-006	0.4950

Comparisons of searching process among BFO, ABC, FOA and SGSO for *f*1 to *f*9 in 30 dimensions are shown in Fig. 2 to Fig. 10. Because of the similar values for the optimum, we plot the searching curves by evaluating the logarithm of function values for better observing. SGSO and PSO converge fast and the value comes to 0 before the 40 iterations, hence its curve is broken in Fig.8. Due to the characteristic of *f*9

PSO converges fast but the optimum is much worse than FOA and SGSO; BFO converges slower than PSO, the optimum is as worse as PSO; ABC gains the best optimum to 0 in *f*7, the searching speed is less than SGSO; FOA have the worst convergence, its searching curves are not convergent finally, although the optimum is smaller than it in PSO, BFO and ABC taking the second place; SGSO performs best in convergence speed. Its optimum comes to the minimum which shows its good searching ability. In short, SGSO is a brilliant algorithm for searching the optimum with fast convergence speed.

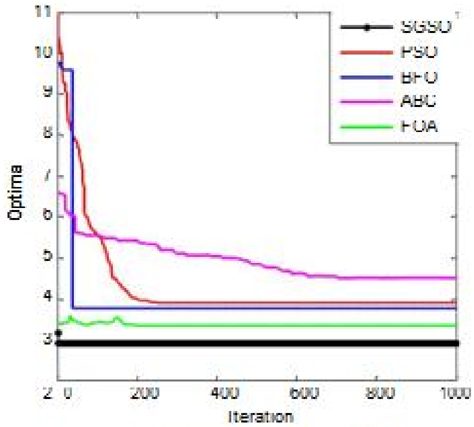

Fig. 2. Searching curves for f_1

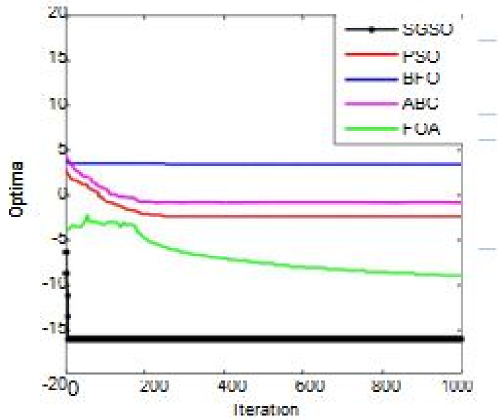

Fig. 3. Searching curves for f_2

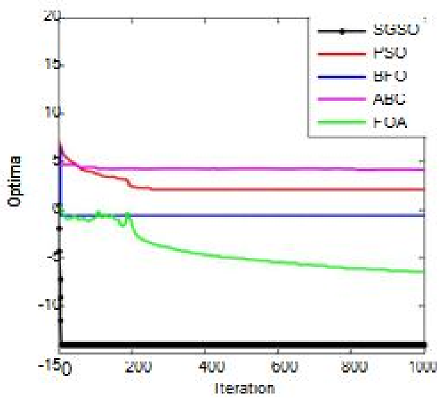

Fig. 4. Searching curves for f_3

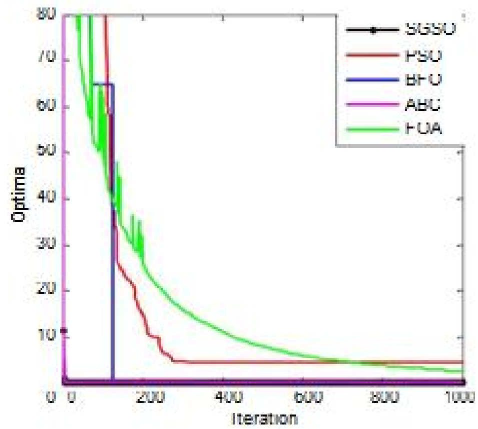

Fig. 5. Searching curves for f_4

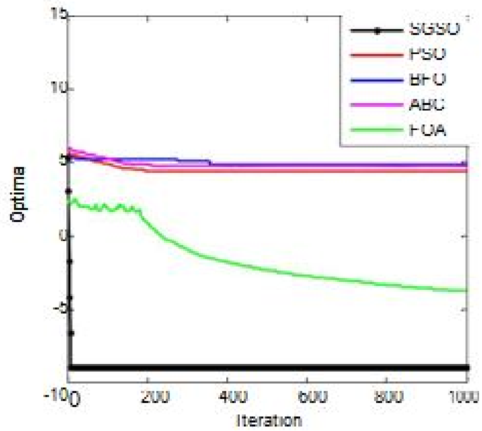

Fig. 6. Searching curves for f_5

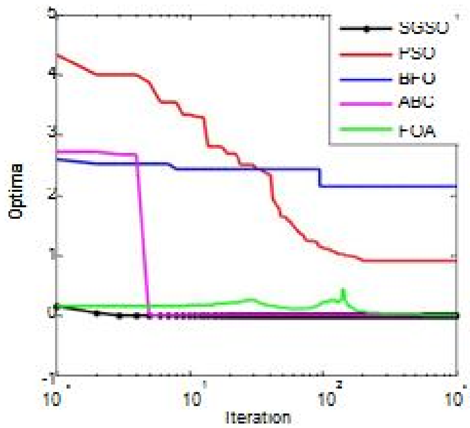

Fig. 7. Searching curves for f_6

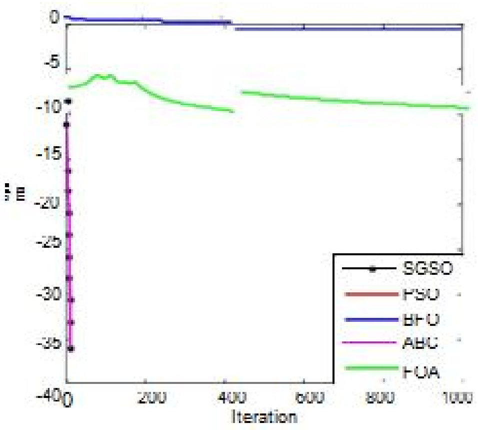

Fig. 8. Searching curves for f

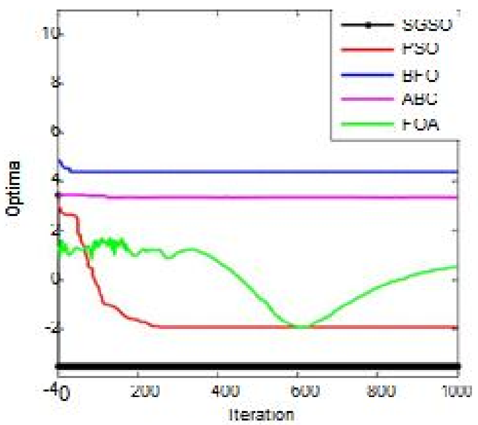

Fig. 9. Searching curves for f

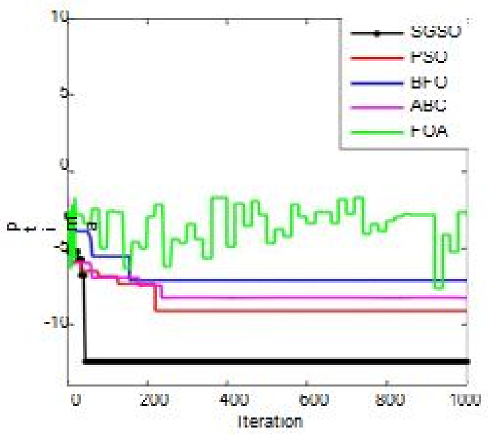

Fig. 10. Searching curves for f

IV. CONCLUSION

A simplified glowworm swarm optimization algorithm was proposed in this paper, which improved the basic GSO in that: (1) omitted the dynamic decision space and probabilistic mechanism for selection, reduced the time complexity of the algorithm and enhanced the efficiency; (2) modified the location update mechanism, the location update transferred based on dynamic decision space to both the optimal and the searching space for solutions; (3) adopt the elitism strategy which makes the excellent glowworms to the next iteration. Results on benchmark function optimization experiments show that SGSO performs much better than the basic GSO in searching ability and the running time, but also than other recent swarm intelligent algorithms such as PSO, BFO, ABC and FOA. As can be seen from the experimental results, SGSO owns the best mean values but not the best minimum values for all problems. We intend to ameliorate the performance and apply SGSO to other problems like clustering to test the performance in future.

V. REFERENCES

[1] K.M. Passino, “Biomimicry of bacterial foraging for distributed optimization and control,” IEEE Control Systems Magazine, vol.22, 52-67, 2002.

[2] J. Kennedy, R.C. Eberhard, “Particle swarm optimization,” in Proceedings of the First IEEE International Conference on Neural Networks, Perth, Australia, IEEE Press, 1995, pp.1942-1948.

[3] D. Karaboga, “An idea based on honey bee swarm for numerical optimization,” Erciyes University, Engineering Faculty, Computer Engineering Department, Erciyes, Turkey, Tech. Rep.TR06, 2005.

[4] Manoj Thakur, “A new genetic algorithm for global optimization of multimodal continuous functions,” Journal of Computational Science, vol. 5, no. 2, pp. 298-311, March 2014.

[5] Ramin Barati, “A novel approach in optimization problem for research reactors fuel plate using a synergy between cellular automata and quasi-simulated annealing methods,” Annals of Nuclear Energy, vol. 70, pp. 56-63, August 2014.

[6] Anupam Yadav, Kusum Deep, “An efficient co-swarm particle swarm optimization for non-linear constrained optimization,” Journal of Computational Science, vol. 5, no. 2, pp. 258-268, March 2014.

[7] Michalis Mavrovouniotis, Shengxiang Yang, “Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors”, Applied Soft Computing, vol. 13, no. 10, pp. 4023-4037, October 2013.

[8] Kaushik Kumar Bhattacharjee, S.P. Sarmah, “Shuffled frog leaping algorithm and its application to 0/1 knapsack problem,” Applied Soft Computing, vol. 19, pp. 252-263, June 2014.

[9] Satyasai Jagannath Nanda, Ganapati Panda, “A survey on nature inspired metaheuristic algorithms for partitional

clustering," Swarm and Evolutionary Computation, vol. 16, pp. 1-18, June 2014.

[10] Ruochen Liu, Yangyang Chen, Licheng Jiao, Yangyang Li, "A particle swarm optimization based simultaneous learning framework for clustering and classification, " Pattern Recognition, vol. 47, no. 6,2143-2152, June 2014.

[11] Vi jay Kumar, Jitender Kumar Chhabra, Dinesh Kumar, "Automatic cluster evolution using gravitational search algorithm and its application on image segmentation," Engineering Applications of Artificial Intelligence, vol. 29, pp. 93-103, March 2014.

[12] Mostafa Z. Ali, Noor H. Awad, "A novel class of niche hybrid Cultural Algorithms for continuous engineering optimization," Information Sciences, vol. 267, pp. 158-190, May 2014.