2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Self-adaptive Group Search Optimizer with Elitist Strategy

Xiang-wei Zheng, Dian-jie Lu, Zhen-hua Chen
School of Information Science and Engineering, Shandong Normal University, Jinan, China
Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan, China
xwzhengcn@ gmail.com

Abstract—**To deal with the disadvantages of Group Search Optimizer (GSO) as slow convergence, easy entrapment in local optima and failure to use history information, a Self-adaptive Group Search Optimizer with Elitist strategy (SEGSO) is proposed in this paper. To maintain the group diversity, SEGSO employs a self-adaptive role assignment strategy, which determines whether a member is a scrounger or a ranger based on *ConK* consecutive iterations of the producer. On the other hand, scroungers are updated with elitist strategy based on simulated annealing by using history information to improve convergence and guarantee SEGSO to remain global search. Experimental results demonstrate that SEGSO outperform particle swarm optimizer and original GSO in convergence rate and escaping from local optima.**

Keywords—group search optimizer; simulated annealing; elitist strategy; role assignment

I. Introduction

Evolutionary Algorithms (EAs) are popular in recent years for its easy implementation and its ability to efficiently solve a lot of complex problems which are difficult to deal with traditional optimization algorithms [1]. Inspired by animal searching behavior and group living theory, a new optimization approach named as Group Search Optimizer (GSO) is proposed by He S, Wu Q H and Saunders J R in 2006 [2]. The population of GSO is called a *group* and each individual in the population is called a *member*. In GSO, members are classified as a producer, scroungers and rangers. Producer is responsible for finding food and is the best member in the group. The framework of GSO is mainly based on the producer–scrounger (PS) model where scroungers join producers to find food while rangers perform random walk motions to avoid entrapments. Comparing with ACO and PSO, GSO emphasizes more on imitating searching behavior of animals. In original GSO algorithm, about 80% individuals are chosen as scroungers, and the producer is the one and only destination of them. In contrast with other EAs, GSO is conceptually simple and easy to be implemented. Original GSO and some improved GSO are proved to be an efficient method for solving function optimization problems [3].

However, GSO has the disadvantages as slow convergence, easy entrapment in local optima and failure to use history information. To deal with these disadvantages of GSO, a Self-adaptive Group Search Optimizer with Elitist strategy (SEGSO) is proposed in this paper. First, SEGSO employs a self-adaptive role assignment strategy to determine whether a member is a scrounger or a ranger based on *ConK* consecutive iterations of the producer. This dynamic role change will maintain group diversity. Second, scroungers are updated with elitist strategy based on Simulated Annealing (SA) by using history information to improve convergence and guarantee SEGSO to remain global search. At last, experimental results demonstrate that SEGSO outperform particle swarm optimizer and original GSO in convergence rate and escaping from local optima.

This work was supported in part by the National Natural Science Foundation of China (61373149, 61272094) and the Promotive Research Fund for Excellent Young and Middle-aged Scientists of Shandong Province (BS2010DX033).

The rest of this paper is organized as follows. Section II briefly reviews related work of GSO and Section III describes the proposed SEGSO algorithm in our study. Section IV presents experimental results and discussion. Section V concludes the paper and points out future work.

II. Related work

To deal with the disadvantages of GSO, some improved GSOs are proposed and then applied to some practical application problems.

Firstly, some improved GSOs incorporated other EAs' operators. D Chen, J Wang and et al proposed an improved GSO optimizer with quantum-behaved operator for scroungers to escape from local optima [4]. The scroungers are divided into two parts, the scroungers in the first part update their positions with the operators of QPSO to improve diversity of population, and the remainders keep searching for opportunities to join the resources found by the producer. Q Kang, T Lan, Y Yan and et al proposed an improved group search optimizer (iGSO) by incorporating particle swarm optimization for optimal setting of distributed generations [5].

Secondly, some improved GSOs modified GSO structure. M Junaed, M Akhand and K Murase studied model of multiple producers in nature and extended GSO with multiple producers [6]. M Moradi-Dalvand, B Mohammadi-Ivatloo, and et al proposed a continuous version of quick group search optimizer (QGSO) [7] and is applied for solution of non-convex and large scale economic dispatch problems. Based on cooperative evolutionary mechanism and divide-and-conquer paradigm, L Pacifico and T Ludermir designed a Cooperative Group Search Optimizer (CGSO) [8], which improves the performance of standard GSO. Experiments on some benchmark functions show that CGSO is able to achieve better results than standard GSO in most of the tested problems.

At last, some improved GSOs were employed to practical applications. L Wang, X Zhong and M Liu applied group search optimizer to solve multi-objective optimization problems [9]. The scanning strategy of the original GSO is replaced by the limited pattern search procedure and a special mutation is designed to balance the exploration and exploitation. In addition, the non-dominated sorting scheme and multiple producers are used in the algorithm. S He, Q Wu and J Saunders applied a group search optimizer (GSO) to train a three-layer feed-forward artificial neural network used for diagnosis of breast cancer [10], including connection weights and bias. The comparison experiments show that GSO for artificial neural network has a better convergence rate and generalization performances for the breast cancer diagnosis problem.

In summary, these improved and extended GSOs overcame the disadvantages of GSO in certain degree. However, GSO still has the disadvantages as slow convergence, easy entrapment in local optima and failure to use history information, therefore, this paper presents a new SEGSO to enhance its capability for search and optimization.

III. A Self-adaptive Group Search Optimizer with Elitist Strategy

A. *Self-adaptive role assignment*

As mentioned above, the members in GSO group are classified as a producer, scroungers and rangers. This classification is similar to multi-population algorithms where the individuals of a sub-population can migrate from one subpopulation to another. However, when to migrate individuals and how many individuals should be migrated are two key issues in multi-population algorithms.

By referring efficiency population utilization strategy for particle swarm optimizer (EPUS-PSO) [11], a self-adaptive role assignment method, namely how to determine whether a member belongs to scroungers or rangers, is designed to improve the convergence rate and accuracy of original GSO. The dynamic assignment model of scroungers and rangers defined in our study is different from the original GSO. At the initial stage, after selecting a producer, scroungers are the top 80% members wth best fitness value and the remainders are rangers. With evolution progress, scrounger and rangers are interchanged according to optimization status. By borrowing idea from EPUS-PSO, a group manager is introduced to SEGSO to enhance its searching ability. The group manager will increase or decrease numbers of scroungers according to the producer status. The numbers of scroungers and rangers are updated according to the following rules.

(1) If the producer has not been updated in *ConK* consecutive iterations, and if the current number of the scroungers doesn't exceed the predefined number, a ranger will be designated as a scrounger. Here the random selection is employed to designate a ranger to be a scrounger.

(2) If the producer is not updated within [round(*ConK* /2), *ConK*] consecutive iterations, and if the current number of the ranger doesn't exceed the predefined number, a scrounger will be designated as a ranger and is updated according to the motion rule. Here the roulette wheel selection is employed to designate a worse scrounger to be a ranger.

(3) Otherwise, the numbers of scroungers and rangers are not changed.

Figure 1 is the variation of scrounger number and unchanged iteration of the producer based on dynamic self-adaptive role assignment. Here, *ConK*=6, *group size*=75.

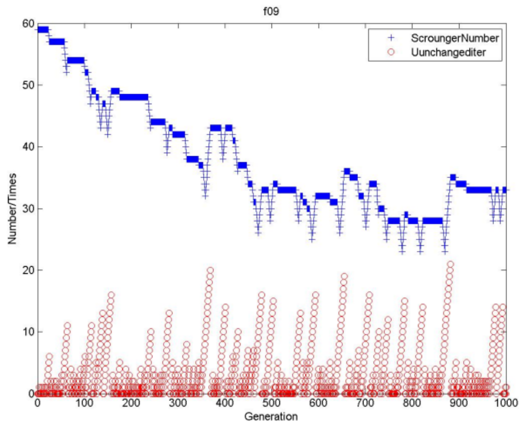

Fig. 1. the variation of scrounger number and unchanged iteration of the producer

B. *Scrounger update with elitist strategy based on SA*

In original GSO, once a producer is determined, scroungers will be randomly selected according to predefined ratio. However, scroungers may walk to a bad position and this will slow the convergence rate. Therefore SEGSO introduces elitist strategy to save the local optimal positions. Sometimes, the convergence rate is improved but the algorithms may be trapped in local optima. In our study, the idea of SA is employed to construct the elitist strategy of scrounger. SA was selected based on the following considerations. On the one hand, SA is a global search algorithm and is theoretically proved effective. On the other hand, SA is simple and successfully applied in many fields. At *(k+1)th* iteration, the behavior of *ith* scrounger X_i^{k+1} is modeled and updated as the following.

```
$X_i^{k+1} = X_i^k + r_3 \circ (X_p^k - X_i^k)$
If $f(X_i^{k+1}) < f(X_i^k)$
Then the scrounger moves to $X_i^{k+1}$
Else If $r_1 < p_{k+1}$
    Then the scrounger moves to $X_i^{k+1}$
    Else the scrounger don't move i.e. $X_i^{k+1} = X_i^k$
    End
End
```

Where r_1, r_3 is random between (0,1); $f(x)$ is the fitness function; p_{k+1} is the annealing rate and p_{k+1} is defined in (10).

$$p_{k+1} = \mathrm{A}\frac{\mathrm{k}}{\mathrm{Maxiteration}} \tag{10}$$

Maxiteration is the maximum iteration time of the algorithm. *A* is the annealing constant.

The benefits of our strategy are as follows. At the beginning, a scrounger joins the producer and retains its optimal position found by then because the initial search range is big and the search behavior is blind. With the progress of the search, some local optima or global optima are found, the producer will retain the search result and scroungers are inspired to search more available positions.

C. *Algorithm description*

The proposed SEGSO algorithm is described as follows.

```
Define and initialize algorithm parameters;
Randomly initialize positions and head angles of all members;
For i = 1 : MaxIteration
    Calculate the current fitness values of all members;
    Sort the members according to their fitness values in ascending
        order (for minimum);
    Select a producer, namely the member with best fittness value;
    Calculate the unchangediter of the producer;
    If unchangediter > ConK and scroungernumber <=
    predefinedscroungernumber
        Randomly select a ranger and change it as a scrounger;
    Elseif unchangediter >= round(ConK/2) and unchangediter < ConK
    and srangernumber <= predefinedsrangernumber
        Employ roulette wheel selection to select a scrounger and
        designate it as a ranger;
      End
    End
    Update scroungers with elitist strategy based on SA;
    Update rangers;
    Adjust feasible bound;
End
Output the results
```

IV. Simulation Experiments

A. *Benchmark functions*

In order to verify effectiveness of the proposed SEGSO, thirteen benchmark functions were used as listed in Table I which had local optima in their search spaces. The first seven functions, namely f_{01} to f_{07}, are unimodal functions while the others are multimodal functions [3].

TABLE I BENCHMARK FUNCTIONS

No.	Benchmark Functions	D	S
f_{01}	$f_{01}(x)=\sum_{i=1}^{n} x_i^2$	30	[-100,100]
f_{02}	$f_{02}(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]
f_{03}	$f_{03}(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	30	[-100,100]
f_{04}	$f_{04}(x)=\max_i\{\lvert x_i, 1\le i\le n\rvert\}$	30	[-100,100]
f_{05}	$f_{05}(x)=\sum_{i=1}^{n-1}(100(x_{i-1}-x_i^2)^2+(x_i-1))^2$	30	[-30,30]
f_{06}	$f_{06}(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	30	[-100,100]
f_{07}	$f_{07}(x)=\sum_{i=1}^{n} ix_i^4+\mathrm{random}[0,1)$	30	[-1.28,1.28]
f_{08}	$f_{08}(x)=-\sum_{i=1}^{n}(x_i\sin(\sqrt{\lvert x_i\rvert}))$	30	[-500,500]
f_{09}	$f_{09}(x)=-\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)^2$	30	[-5.12,5.12]
f_{10}	$f_{10}(x)=-20\exp(-0.2\sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}x_i^2\right)})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)$	30	[-32,32]
f_{11}	$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{30}(x_i-100)^2-\prod_{i=1}^{n}\cos(\frac{x_i-100}{\sqrt{i}})+1$	30	[-600,600]
f_{12}	$f_{12}(x)=\frac{\pi}{n}\Big\{10\sin^2(\pi y_1)+\sum_{i=1}^{29}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\Big\}+\sum_{i=1}^{30}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_1+1)$ $u(xi,a,k,m)=\begin{cases}k(x_i-a)^m, x_i>a\\ 0, -a\le x_i\le a\\ k(-x_i-a)^m, x_i<-a\end{cases}$	30	[-50,50]

*f*13	$$f_{13}(x) = 0.1\left\{\sin^2(\pi 3x_1 + \sum_{i=1}^{29}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + (x_n-1)^2[1+\sin^2(2\pi x_{30})]\right\} + \sum_{i=1}^{30} u(x_i,5,100,4)$$	30	[-50,50]

B. *Simulation platform and setting*

Original GSO and PSO are selected as comparative algorithms in the following experiments. All the algorithm codes were written and executed in MATLAB 2008Ra in our study. Some codes in Matlab were downloaded from the authors' homepages. The experiments were carried out on a PC with an i5 3.2 GHz Intel Processor and 4.0 GB RAM. The operating system is Microsoft Windows 7.

The function evaluation times (FEs) for all the algorithms are the same. Here, for f_{03} and f_{09}, the FEs are 250000 and for the others, the FEs are 150000 respectively.

For PSO, experimental parameters are given as follows.

- Population size: 75 or 125(for f_{03} and f_{09})
- *Maxiteration*: 2000
- c_1, c_2: 2
- ω: 1.0~0.4
- End times of ω: *Maxiteration*/2
- Mutation probability: 0.1
- Running times for each benchmark function: 50

For GSO and SEGSO, experimental parameters are given as follows.

- Group size: 75 or 125(for f_{03} and f_{09})
- *Maxiteration*: 2000
- Initial ratio of Scroungers to Rangers : 4:1
- Initial angle: $\pi/4$
- Max pursuit angle: $\pi/\lfloor\sqrt{D+1}\rfloor$
- Max turning angle: Max pursuit angle/2
- *ConK*: 6 (only for SEGSO, *ConK* can be tuned according the benchmark functions)
- Running times for each benchmark function: 50

C. *Experimental results and discussion*

The experimental results on 13 benchmark functions are presented in Table II and Figure 2~14. The rank values were calculated based on the mean value of selected algorithms.

It was found that the SEGSO has a better performance compared with original GSO and PSO in terms of accuracy and convergence speed. For unimodal functions (f_{01} to f_{07}), experimental results show the better explorative ability of SEGSO. According to the ranks, SEGSO outperformed PSO and GSO on most benchmarks. For multimodal functions (f_{08} to f_{13}), SEGSO demonstrates its strong ability in escaping from local optima. However, for f_{05}, SEGSO doesn't have good performance.

TABLE II STATISTIC RESULTS FOR 13 BENCHMARK FUNCTIONS

Benchmark	Algorithm	Mean	Std	Best	Worst	Median	Rank
*f*01	PSO	2.1436e-002	3.1569e-002	6.0356e-004	1.3842e-001	9.0447e-003	3
	GSO	1.6006e-009	3.1332e-009	6.1693e-011	1.9639e-008	5.2713e-010	2
	SEGSO	1.4666e-010	1.9461e-010	8.4766e-012	8.6182e-010	8.2305e-011	1
*f*02	PSO	3.1342e-002	2.1054e-002	3.4528e-003	1.0195e-001	3.1808e-002	3
	GSO	9.3213e-006	8.5055e-006	1.7932e-006	4.6436e-005	6.1897e-006	2
	SEGSO	3.5041e-006	3.3735e-006	7.1413e-007	2.0341e-005	2.3744e-006	1
*f*03	PSO	3.9197e+003	3.9189e+003	3.0487e+002	1.6839e+004	1.9128e+003	3
	GSO	1.2313e+001	6.9305e+000	2.6252e+000	3.4459e+001	1.0960e+001	2
	SEGSO	2.8208e+000	2.0924e+000	2.3930e-001	9.7607e+000	2.3269e+000	1
*f*04	PSO	3.3875e+000	8.4892e-001	1.9370e+000	5.2556e+000	3.2230e+000	3
	GSO	5.4923e-002	1.9654e-002	2.1465e-002	1.2235e-001	5.3978e-002	2
	SEGSO	2.8155e-002	1.2606e-002	6.8287e-003	6.4056e-002	2.6930e-002	1
*f*05	PSO	3.1793e+002	7.1733e+002	1.7360e+001	3.0903e+003	1.0674e+002	3
	GSO	4.0689e+001	2.9032e+001	1.5961e+000	9.0766e+001	2.6631e+001	1
	SEGSO	5.2254e+001	3.0211e+001	5.4484e+000	9.6970e+001	7.0649e+001	2
*f*06	PSO	1.6000e-001	5.0950e-001	0	3.0000e+000	0	3
	GSO	2.0000e-002	1.4142e-001	0	1.0000e+000	0	2
	SEGSO	0	0	0	0	0	1
*f*07	PSO	2.3811e-002	9.0801e-003	1.2232e-002	5.4011e-002	2.0028e-002	2

	GSO	4.2654e-002	2.2598e-002	7.4748e-003	9.7094e-002	3.7339e-002	3
	SEGSO	1.7714e-002	9.3533e-003	4.3225e-003	4.6161e-002	1.6536e-002	1
f_{08}	PSO	-1.1369e+004	3.8375e+002	-1.2331e+004	-1.0543e+004	-1.1367e+004	3
	GSO	-1.2569e+004	2.7258e-002	-1.2569e+004	-1.2569e+004	-1.2569e+004	1
	SEGSO	-1.2561e+004	3.7918e-003	-1.2561e+004	-1.2569e+004	-1.2569e+004	2
f_{09}	PSO	1.2053e+002	5.6978e+001	4.1632e+001	3.0055e+002	1.0988e+002	3
	GSO	1.6435e+000	1.3207e+000	1.6088e-018	5.9408e+000	1.4851e+000	2
	SEGSO	5.3457e-001	6.0696e-001	4.2783e-018	1.9799e+000	2.0495e-008	1
f_{10}	PSO	8.3420e-001	3.9389e+000	6.6982e-003	1.9941e+001	3.0045e-002	3
	GSO	6.2292e-006	4.0929e-006	1.5045e-006	2.0800e-005	5.1193e-006	2
	SEGSO	3.9064e-006	5.0246e-006	8.8179e-007	2.8644e-005	2.5856e-006	1
f_{11}	PSO	9.3817e-002	1.1698e-001	2.1020e-004	5.8898e-001	4.5348e-002	3
	GSO	2.6621e-002	2.9299e-002	4.7978e-011	1.3977e-001	2.3365e-002	2
	SEGSO	2.2775e-002	2.8085e-002	8.1196e-011	1.3699e-001	1.2321e-002	1
f_{12}	PSO	6.1239e-004	2.4898e-003	2.2953e-007	1.7424e-002	1.2008e-004	3
	GSO	2.2822e-012	3.7439e-012	5.5319e-014	1.8088e-011	9.9018e-013	2
	SEGSO	6.1510e-013	2.4722e-012	1.0988e-014	1.7586e-011	1.6164e-013	1
f_{13}	PSO	5.7231e-003	7.9886e-003	2.6358e-005	3.3631e-002	2.3278e-003	3
	GSO	2.5332e-010	1.5534e-009	1.2818e-012	1.1012e-008	1.5916e-011	2
	SEGSO	6.9578e-012	1.0514e-011	4.1858e-013	5.1822e-011	3.0661e-012	1

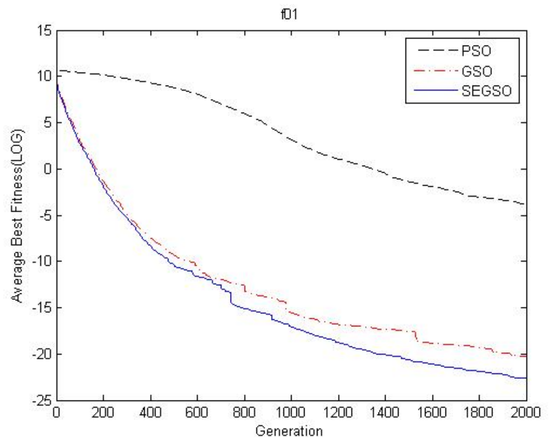

Fig. 2. the average fitness of f_{01}

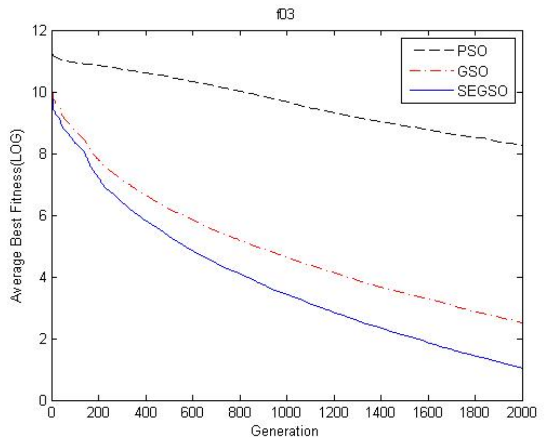

Fig. 4. the average fitness of f_{03}

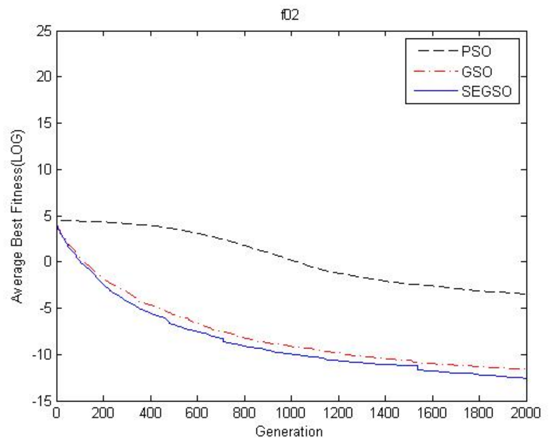

Fig. 3. the average fitness of f_{02}

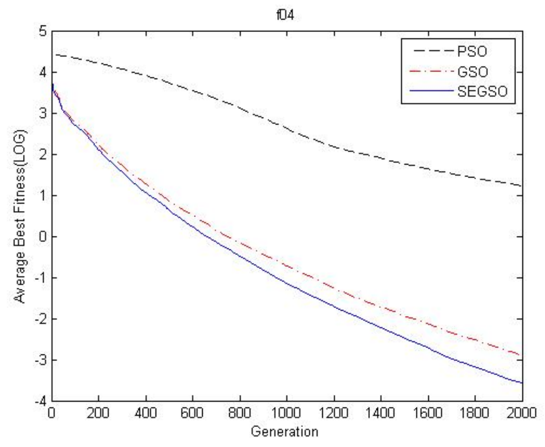

Fig. 5. the average fitness of f_{04}

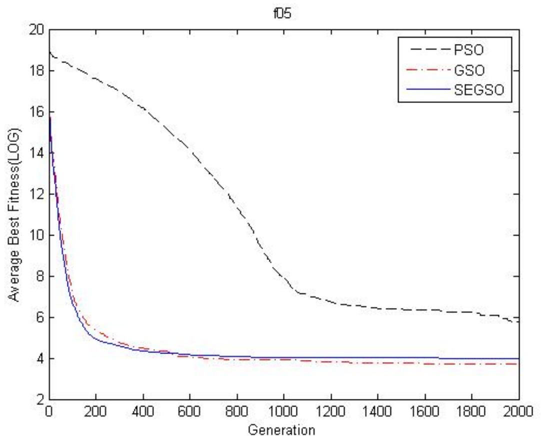

Fig. 6. the average fitness of f_{05}

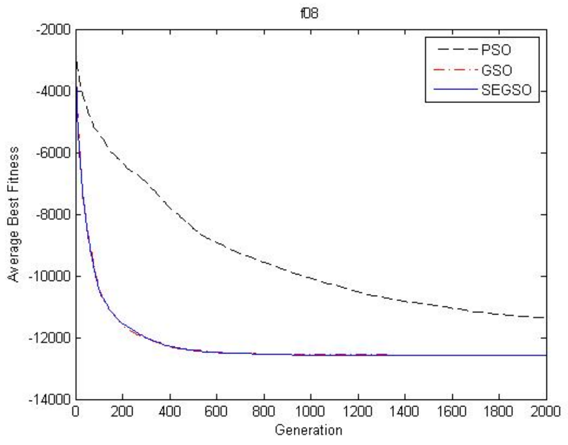

Fig. 9. the average fitness of f_{08}

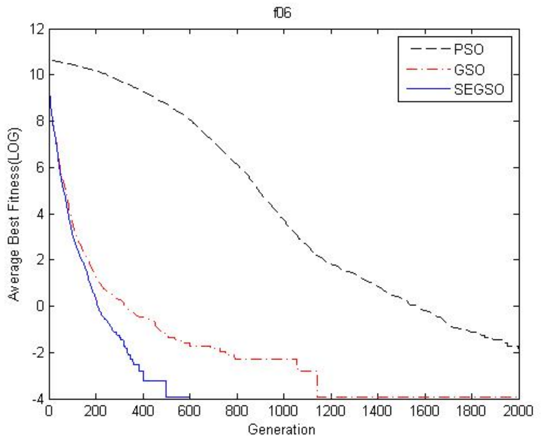

Fig. 7. the average fitness of f_{06}

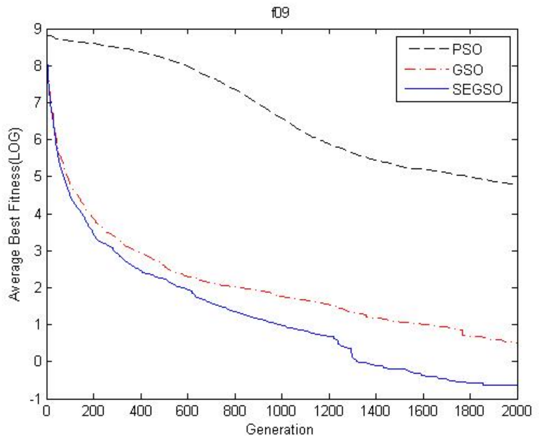

Fig. 10. the average fitness of f_{09}

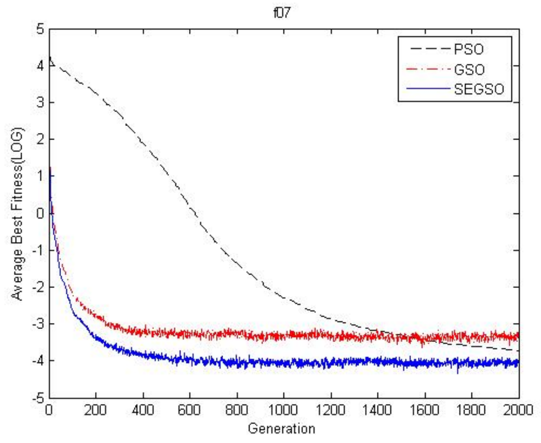

Fig. 8. the average fitness of f_{07}

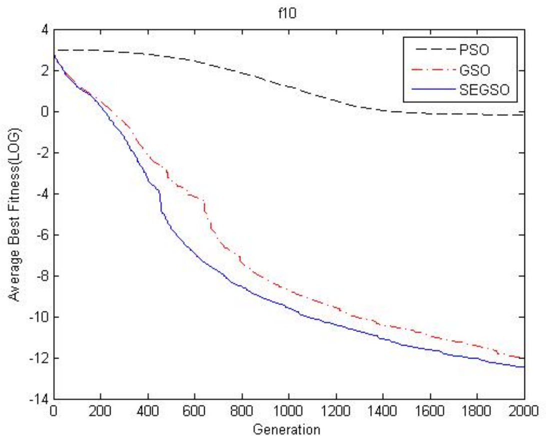

Fig. 11. the average fitness of f_{10}

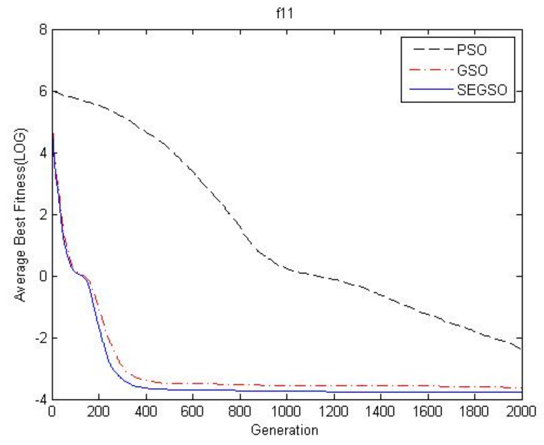

Fig. 12. the average fitness of f_{11}

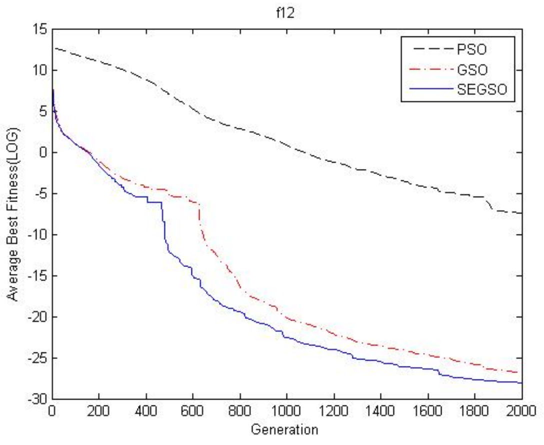

Fig. 13. the average fitness of f_{12}

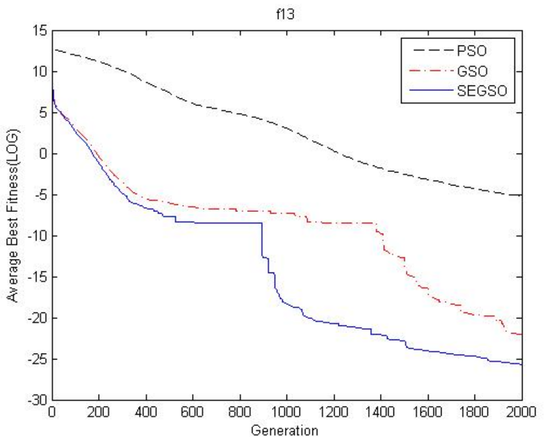

Fig. 14. the average fitness of f_{13}

V. CONCLUSION AND FUTURE WORK

This paper presents a self-adaptive group search optimizer with elitist strategy to deal with the disadvantages of GSO. Based on *ConK* consecutive iterations of the producer, SEGSO employs a self-adaptive role assignment strategy to determine whether a member is a scrounger or a ranger to maintain group diversity. Scroungers are updated with elitist strategy based on SA by using history information to improve convergence and guarantee SEGSO to remain global search. Experimental results demonstrate the effectiveness of SEGSO. However, improvement of SEGSO to GSO is still limited and some parameters have to be determined on our experiences. In the future, new dynamic role assignment strategy and search methods will be elaborated to suit GSO and applied to more fields.

REFERENCES

[1] B. Akay and Y. Xin, "Recent Advances in Evolutionary Algorithms for Job Shop Scheduling," *Automated Scheduling and Planning*, Springer Berlin Heidelberg, pp. 191-224, 2013.

[2] S. He, Q. H. Wu and J. R. Saunders, "A novel group search optimizer inspired by animal behavioural ecology," in *Proceeding of 2006 IEEE Congress on Evolutionary Computation*, vol. 4, pp.16-21, 2006.

[3] S. He, Q. H. Wu and J. R. Saunders, "Group search optimizer: an optimization algorithm inspired by animal searching behavior," *IEEE Transactions on Evolutionary Computation,* vol. 13, no.5, pp. 973-990, 2009.

[4] D. Chen, J. Wang, F. Zou, W. Hou and C. Zhao, “An improved group search optimizer with operation of quantum-behaved swarm and its application,” *Applied Soft Computing*, vol. 12, no. 2, pp. 712–725, February 2012.

[5] Q. Kang, T. Lan, Y. Yan and et al, “Group search optimizer based optimal location and capacity of distributed generations,” *Neurocomputing*, vol. 78, no. 1, pp. 55-63, 2012.

[6] A. Junaed, M. Akhand and K. Murase, “Multi-producer group search optimizer for function optimization,” in *IEEE International Conference on Informatics, Electronics & Vision (ICIEV)*, pp. 1-4, 2013.

[7] M. Moradi-Dalvand, B. Mohammadi-Ivatloo, A. Najafi and et al, “Continuous quick group search optimizer for solving non-convex economic dispatch problems,” *Electric Power Systems Research*, vol. 93, pp. 93-105, 2012.

[8] L. Pacifico and T. Ludermir, “Cooperative Group Search Optimization,” in *2013 IEEE Congress on Evolutionary Computation (CEC’ 2013)*, pp. 3299-3306, 2013.

[9] L. Wang, X. Zhong and M. Liu, “A novel group search optimizer for multi-objective optimization,” *Expert Systems with Applications*, vol. 39, no. 3, pp. 2939–2946, February 2012.

[10] S. He, Q. H. Wu, and J. R. Saunders, "Breast cancer diagnosis using an artificial neural network trained by group search optimizer," *Transactions of the Institute of Measurement and Control*, vol. 31, no. 6, pp. 517-531, 2009.

[11] S. Hsieh, T. Sun, C. Liu and S. Tsai, “Solving large scale global optimization using improved particle swarm optimizer,” in *IEEE Congress on Evolutionary Computation*, pp. 1777−1784, 2008. [doi: 10.1109/CEC.2008.4631030]