Journal of Experimental & Theoretical Artificial Intelligence

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/teta20

Success rate group search optimiser

Mohammad Hasanzadeh[ab], Sana Sadeghi[c], Alireza Rezvanian[ad] & Mohammad Reza Meybodi[a]

[a] Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran

[b] ICT Research Institute, Academic Center for Education, Culture and Research (ACECR), Tehran, Iran

[c] Department of Engineering, Payame Noor University (PNU), Tehran, Iran

[d] Department of Computer Engineering and Information Technology, Hamedan University of Technology, Hamedan 65155-579, Iran

Published online: 06 Nov 2014.

To cite this article: Mohammad Hasanzadeh, Sana Sadeghi, Alireza Rezvanian & Mohammad Reza Meybodi (2014): Success rate group search optimiser, Journal of Experimental & Theoretical Artificial Intelligence, DOI: 10.1080/0952813X.2014.971467

To link to this article: http://dx.doi.org/10.1080/0952813X.2014.971467

Journal of Experimental & Theoretical Artificial Intelligence, 2014
http://dx.doi.org/10.1080/0952813X.2014.971467

Success rate group search optimiser

Mohammad Hasanzadeh[a,b1], Sana Sadeghi[c2], Alireza Rezvanian[a,d]*
and Mohammad Reza Meybodi[a3]

[a]*Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran;* [b]*ICT Research Institute, Academic Center for Education, Culture and Research (ACECR), Tehran, Iran;* [c]*Department of Engineering, Payame Noor University (PNU), Tehran, Iran;* [d]*Department of Computer Engineering and Information Technology, Hamedan University of Technology, Hamedan 65155-579, Iran*

(*Received 18 March 2014; accepted 21 September 2014*)

The group search optimiser (GSO) algorithm is a newly found evolutionary algorithm that is inspired by animal-searching behaviour and group living theory. The GSO algorithm follows the producer–scrounger framework that consists of producer, scrounger and ranger members. There are multiple key parameters in the GSO algorithm that directly affect the performance of the algorithm. Among these parameters, the maximum pursuit distance parameter plays an important role because it determines the step length of the producer and rangers of the GSO algorithm. In this paper, we develop a modified GSO algorithm by using the success rate model to adjust the maximum pursuit distance parameter of the algorithm. We test the proposed algorithm on a rich set of benchmark functions including 30- and 300-dimensional problems and compare the results with popular evolutionary and swarm algorithms. The experimental results demonstrate that the scanning mechanism of the proposed algorithm quickly optimises not only the 30-dimensional problems but also the high-dimensional (300D) problems.

Keywords: evolutionary algorithms; group search optimiser; success rate; swarm intelligence

1. Introduction

Artificial intelligence (AI) (Chalmers, French, & Hofstadter, 1992) tries to overcome human restrictions in solving seemingly unsolvable mysteries. The AI benefits human through its defying logic properties and boosts the quality of human life. Evolutionary computing (EC) is a subset of AI that aims at solving optimisation problems with the inspiration of adventurous characteristics of fauna over their evolutionary time. Over time, natural evolution forces fauna to improve their qualities and utilities in order to lift their productivity, longevity and lifestyle. Evolutionary algorithms (EAs) are population-based and momentary EC techniques that are mostly derived from animal's natural behaviour. These algorithms have been widely studied by many authors for solving engineering optimisation problems and triggering a new way of modelling and simulation of big data. The challenges behind optimisation problems feed the fierce ideas of neoteric emerging EAs.

*Corresponding author. Email: a.rezvanian@aut.ac.ir

EA techniques model the biological behaviour of evolution by utilizing different algorithms. Imperialist competitive algorithm (Yas, Kamarian, & Pourasghar, 2014) consists of random points called countries where their fitness is measured by their power. Countries divide the problem space into two types of empires, including colonies and imperialists. The powerful countries become imperialist and start taking control of weak ones (colonies). Harmony search (HS) algorithm (Estahbanati, 2014) is inspired by extemporaneously music composition. In the HS algorithm, each decision variable (musician) generates (plays) a value (note) for finding a global optima (best harmony). Particle swarm optimisation (PSO) (Hasanzadeh, Meybodi, & Ebadzadeh, 2013; Kordestani, Rezvanian, & Meybodi, 2014; Nabizadeh, Rezvanian, & Meybodi, 2012; Rezaee Jordehi & Jasni, 2013) is a heuristic-based iterative technique that uses a population of particles. Each particle of PSO represents a feasible solution of problem space. PSO keeps track of the best values of each individual and the entire population in order to optimise the problem. Cuckoo search algorithm (Kavousi-Fard & Kavousi-Fard, 2013) is inspired by the special breeding behaviour of cuckoo. The cuckoos (population) are laying their eggs (new solutions) in the nest of other species. Firefly algorithm (Kamarian, Yas, Pourasghar, & Daghagh, 2014) is inspired by the flashing behaviour of fireflies. All the fireflies (population) attract other fireflies through the associated light intensity. Artificial immune algorithms (Rezvanian & Meybodi, 2010) consists of several optimisation techniques inspired by human immune mechanism. In this algorithm, the immune cells (population) try to protect body through immune mechanisms (operators). Genetic algorithm (GA) (Manurung, Ritchie, & Thompson, 2012) is a search meta-heuristic that emulates the process of natural selection. Moreover, non-dominated sorting genetic algorithm II (NSGA II) (Javadi, Saniei, & Rajabi Mashhadi, 2014) is a multiobjective GA that covers pareto-optimal front. Finally, ant colony optimisation (ACO) (Shyu, Yin, Lin, & Haouari, 2003; Soleimani-pouri, Rezvanian, & Meybodi, 2014) is inspired by the colonial behaviour of ants. ACO is a probabilistic approach for finding shortest paths through a graph.

The group search optimiser (GSO) (He, Wu, & Saunders, 2009b) is a population-based optimisation heuristic that adopts the producer–scrounger (PS) model (Barnard & Sibly, 1981). The PS model is a group living methodology with two strategies: (1) producing, e.g. searching for food, and (2) joining (scrounging), e.g. joining resources uncovered by others. Besides producer and scrounger members, the population of GSO algorithm also contains ranger members who perform random walks to avoid getting trapped in pseudo-minima. Moreover, GSO inspires from the concept of animal-foraging behaviour to search the most promising areas of problem space.

This paper proposes success rate GSO (SRGSO) algorithm that enhances the potential advantages of GSO. Maximum pursuit distance is a key parameter of the GSO algorithm. This parameter determines the step size of each producer member of the group. Despite its importance, GSO sets the default value of this parameter as a constant value. Thus, in order to improve the performance of GSO, an adaptive approach to adjust the value of maximum pursuit distance parameter is proposed. The proposed adaptive approach employs success rate strategy (Beyer & Schwefel, 2002) in order to provide a proper feedback from the movements of producer members. The performance of the SRGSO algorithm has been validated on 12 benchmark functions, including 7 unimodal and 5 multimodal problems in two different contexts of general and large scales. Furthermore, the performance of SRGSO has been compared with GA, PSO and GSO. Experimental results show that GSO performs best in terms of both average performance and scalability.

The reminder of this paper is organised as follows. Section 2 provides a short introduction to GSO and its variants. Section 3 provides the SRGSO algorithm. Section 4 describes experimental settings and results. Finally, Section 5 provides the conclusion of paper.

2. Group search optimiser

2.1 GSO concepts and formulations

The GSO algorithm follows the PS framework that models the interactions within and between animals. PS suggests two kinds of individuals: producers that explore for resource opportunities and scroungers that exploit limited resources provided by producers. The population and individuals of GSO are called group and members, respectively. A group consists of three types of members:

(1) Producer members that explore the search space for new optima.
(2) Scrounger members that exploit the local optima uncovered by the producer members.
(3) Ranger members that move randomly in order to avoid group trapping in pseudo-minima.

In an n-dimensional search space, the ith member at the kth generation has $X_i^k \in R^n$, $\varphi_i^k = (\varphi_{i1}^k, \ldots, \varphi_{i(n-1)}^k) \in R^{n-1}$ and $D_i^k(\varphi_i^k) = (d_{i1}^k, \ldots, d_{in}^k) \in R^n$ as position, head angle and head direction, respectively. Here, D_i^k can be calculated from φ_i^k via polar to Cartesian coordinate transformation:

$$\begin{aligned} d_{i1}^k &= \prod_{q=1}^{n-1} \cos\left(\varphi_{iq}^k\right) \\ d_{ij}^k &= \sin\left(\varphi_{i(j-1)}^k\right) \times \prod_{q=j}^{n-1} \cos\left(\varphi_{iq}^k\right) (j = 2, \ldots, n-1) \quad d_{in}^k = \sin\left(\varphi_{i(n-1)}^k\right). \end{aligned} \tag{1}$$

At the kth generation of the GSO algorithm, the producer X_{P} behaves as follows:

(1) The producer scans at zero degree and then randomly selects three points in the reachable scanning area: one point at zero degree (2), one point on the left-side hypercube (3) and one point on the right-side hypercube (4):

$$X_z = X_p^k + r_1 l_{\max} D_p^k(\varphi^k), \tag{2}$$

$$X_r = X_p^k + r_1 l_{\max} D_p^k\left(\varphi^k + r_2 \frac{\theta_{\max}}{2}\right), \tag{3}$$

$$X_l = X_p^k + r_1 l_{\max} D_p^k\left(\varphi^k - r_2 \frac{\theta_{\max}}{2}\right), \tag{4}$$

where $r_1 \in R^1$ is a normally distributed random number with mean 0 and standard deviation 1, $r_2 \in R^{n-1}$ is a uniformly random sequence in the range of (0, 1), $l_{\max}$ is the maximum pursuit distance and $\theta_{\max}$ is the maximum pursuit angle.

(2) Then the producer will find the best position with the best fitness. If the best position of current generation has better fitness than its current position, then it will move to this position, otherwise it will stay on its current position and turn its head to a new angle:

$$\varphi^{k+1} = \varphi^k + r_2 \alpha_{\max}, \tag{5}$$

where $\alpha_{\max}$ is the maximum turning angle.

(3) If the producer cannot find a promising area after a couple of generations, it will turn its back into zero degree:

$$\varphi^{k+a} = \varphi^k, \tag{6}$$

where a is a constant. At the kth generation, the area-copying behaviour of the ith scrounger can be modelled as a random walk towards the producer:

$$X_i^{k+1} = X_i^k + r_3 \circ (X_p^k - X_i^k), \tag{7}$$

where $r_3 \in R^n$ is a uniform random sequence in the range of (0, 1). If the ith member of group is selected as a ranger, at the kth generation, it generates head angle φ_i using Equation (5). Here $\alpha_{\max}$ is the maximum turning angle, and it chooses a random distance

$$l_i = a \times r_1 l_{\max}, \tag{8}$$

and moves to the new point:

$$X_i^{k+1} = X_i^k + l_i D_i^k(\varphi^{k+1}). \tag{9}$$

2.2 GSO applications and variants

EAs have applications in different emerging research areas, including grid computing (Hasanzadeh & Meybodi, 2013, 2014), cloud computing, social networks (Soleimani-Pouri, Rezvanian, & Meybodi, 2012a, 2012b; Soleimani-pouri et al., 2014), etc. Since introducing GSO (He et al., 2009b) for solving optimisation problems, the authors noticed its application in different fields of science, including Truss Structure Design (Li, Xu, Liu, & Wu, 2010), power distribution networks (Kang, Lan, Yan, Wang, & Wu, 2012), economic dispatch problem (Guo, Zhan, & Wu, 2012), premalignant pancreatic cancer discovery (He, Cooper, Ward, Yao, & Heath, 2012), breast cancer diagnosis (He, Wu, & Saunders, 2009a), combined heat and power economic dispatch problem (Hagh, Teimourzadeh, Alipour, & Aliasghary, 2014), mechanical design optimisation (Shen, Zhu, Niu, & Wu, 2009), solving nonlinear equations (Yao, Cui, Wei, & Tan, 2011), ethylene cracking furnace (Nian, Wang, & Qian, 2013) and artificial neural network training (Yan, Yang, & Shi, 2012). With regards to these applications, some improved versions of GSO will discuss in the following section.

2.2.1 Improved GSO

The improved GSO (Shen et al., 2009) divides the population into feasible and infeasible subpopulations based on some problem-specific constraints where each subpopulation has a specific evolutionary process.

(1) In the feasible subpopulation, the member with the best position is selected as a producer member and the scrounger members perform scrounging using Equation (7) and the ranger members perform ranging based on Equation (10):

$$X_i^{k+1} = X_g^k + l_i D_i^k(\varphi^{k+1}), \tag{10}$$

where X_g^k is the best position among the history of subpopulation.

(2) In the infeasible subpopulation, the scrounger members perform scrounging using Equation (7) and the others will perform ranging using Equations (5), (8) and (9).

2.2.2 Quadratic interpolation GSO

The quadratic interpolation GSO (Yao et al., 2011) uses the quadratic interpolation function of Equation (11). Suppose the minimum point is $x = (x_1, \ldots, x_n)$, then

$$\begin{aligned} x_i &= \frac{1}{2}\left\{\frac{A}{B}\right\}, \quad i = 1, \ldots, n \\ A &= \left[(x_i^b)^2 - (x_i^c)^2\right] f_a + \left[(x_i^c)^2 - (x_i^a)^2\right] f_b + \left[(x_i^a)^2 - (x_i^b)^2\right] f_c \\ B &= (x_i^b - x_i^c) f_a + (x_i^c - x_i^a) f_b + (x_i^a - x_i^b) f_c, \end{aligned} \tag{11}$$

where $x^a = (x_1^a, \ldots, x_n^a)$, $x^b = (x_1^b, \ldots, x_n^b)$ and $x^c = (x_1^c, \ldots, x_n^c)$ are three given points and f_a, f_b and f_c are their fitness values, respectively, and the minimum of the three fitness values is f_b.

In each generation, the producer member of GSO uses Equations (2)–(4) to sample the three points. After generating zero and left positions of the producer through Equations (2) and (4), Equation (11) interpolates one polynomial with these positions and the producer current position in order to obtain the right position of the producer. The applied interpolation function utilises fitness landscape to improve the local search.

2.2.3 Improved small world topology GSO

The improved small world topology GSO (ISWGSO) (Yan et al., 2012) utilises the small world scheme of complex network to increase the diversity of GSO's scroungers. In small world (SW) topology, each node connects a small number of adjacent neighbours and then a small number of connections are added to some randomly chosen unconnected nodes.

After the scrounger builds its original neighbourhood, it chooses some other neighbours with the probability of $P_i^k = (p_{i1}^k, \ldots, p_{ij}^k, \ldots, p_{im}^k)$ where p_{ij}^k is the probability of ith member choosing the jth member as a neighbour and is calculated as follows:

$$p_{ij}^k = w_1 \times \frac{k}{\text{MaxInt}} + w_2 \times \left(\exp\left(-\frac{\text{dis}_{ij}^k}{l_{\max}}\right) - \exp(-1)\right), \tag{12}$$

where k is the current generation number, MaxInt is the maximum generation number, $l_{\max}$ is the maximum pursuit distance of GSO, dis_{ij}^k is the Euclidean distance between the ith and jth members at the kth generation and w_1 and w_2 ($w_1, w_2 \in (0, 2)$) are two scalar weights applied to p_{ij}^k.

In ISWGSO, the SW topology is rebuilt for scroungers at each generation. After building neighbourhood, each scrounger learns from the best member in its neighbourhood (local best position) and the best member in the group (global best position) using the following equation:

$$X_i^{k+1} = X_i^k + w_3 \times r_3(X_p^k - X_i^k) + w_4 \times r_4(X_{li}^k - X_i^k), \tag{13}$$

where r_3 and r_4 are two uniform random variables in the range of (0, 1). Also, w_3 and w_4 ($w_3, w_4 \in (0, 2)$) are weights that are applied to the producer's position (X_p^k) and the best position of the ith scrounger's local neighbourhood (X_{li}^k).

2.2.4 Group search particle swarm optimiser

The group search particle swarm optimiser (GSPSO) (Yan & Shi, 2011) hybridises PSO (Hasanzadeh et al., 2013; Hasanzadeh, Meybodi, & Ebadzadeh, 2012; Hasanzadeh, Meybodi, & Ghidary, 2011) and GSO (He et al., 2009b) in order to utilise the high computing power of PSO in general-scale problems and the high convergence speed of GSO in large-scale problems. In GSPSO, the PSO population offers a feasible local search space to the scroungers of GSO and then the rangers of GSO population refine this local search area before remitting it to the PSO population. A mutual rescue method is used for switching between PSO and GSO algorithms. In each generation the switch rate of this method is calculated by the best values obtained from these algorithms.

2.2.5 Novel multiobjective GSO

The novel multiobjective GSO (NMGSO) (Wang, Zhong, & Liu, 2012) proposes a limited pattern search procedure in the polar coordinate instead of GSO's angle-based scanning strategy. Also, in NMGSO, the ranger members search based on an adaptive mutation probability and multiply by random coefficients in order to balance the global and local searches during group evolution and enrich the group diversity, respectively.

When applying NMGSO to the multiobjective optimisation problems, the following challenges should be addressed:

(3) Retain the non-dominated solutions found during the search process.
(4) Select members among the non-dominated solutions to be the next producers.
(5) Maintain the diversity of group.

NMGSO used the Pareto-ranking scheme in order to handle the multiobjective optimisation. Moreover, the kernel diversity estimator is used to retain non-dominated solutions. Also randomness is used to determine the coefficients of members to enhance the group's diversity.

3. Success rate GSO

In the GSO algorithm (He et al., 2009b), the $l_{\max}$ parameter is considered as a fixed value. This parameter has effects, not only on producer members (Equations (2)–(4)), but also on ranger members (Equation (9)). The proposed adaptive strategy tries to balance the convergence speed of the group by adapting the step size of producer members.

Here, we used the success rate (Beyer & Schwefel, 2002) of the group, and applied it as a proper feedback of the group's improvement or deterioration to the $l_{\max}$. High success rate implies that the group starts to improve while low rate implies that the group starts to deteriorate. The success status of the ith member of group in the kth generation of algorithm is defined as follows:

$$S_i^k = \begin{cases} 1 & f\left(X_i^k\right) < f\left(X_i^{k-1}\right) \\ 0 & f\left(X_i^k\right) \geq f\left(X_i^{k-1}\right) \end{cases}, \tag{14}$$

where $f()$ is the fitness function. Moreover, the success rate of group is calculated through Equation (15):

$$P = \frac{\sum_{i=1}^{m} S_i^k}{m}, \tag{15}$$

where m represents the group size and $P \in (0, 1)$ is the success rate of group. From the group distribution viewpoint, the low P value implies that group members are sparsely distributed while the

high P value implies that the group members are densely distributed. Thus, if the $l_{\max}$ value is determined through the P value, the maximum pursuit distance is defined as follows:

$$l_{\max} = f(P^k). \tag{16}$$

Moreover, if the positions of members are not improved, P will be zero ($P = 0$). In this condition, the group is stagnated and the success rate strategy will fail. In order to avoid this situation, the $l_{\max}$ value is calculated as a linear weighted function of P using the following equation:

$$l^k_{\max} = (\sigma_{\max} - \sigma_{\min})P^k + \sigma_{\min}, \tag{17}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are weight parameters and $l_{\max}$ has any acceptable value within the range of ($\sigma_{\min}$, $\sigma_{\max}$). The pseudo-code of the SRGSO algorithm is presented in Figure 1.

Figure 2 shows a schematic view of the SRGSO algorithm. After SRGSO initialisation, the algorithm turns into the generation loop in which producer, scrounger and ranger members perform producing, scrounging and ranging around the local optima, respectively. Furthermore, after calculating the fitness of members, the success rate strategy takes place. It adapts the

```
Define
    Generation number k; Maximum generation MG; Current member index i;
    Position X; head angle φ; distance l; maximum pursuit distance l_max;
    Success status S; Success rate P;
Calculate
    Fitness f(X);
For each generation k Do // generation loop
    For each member i Do // group loop
        Choose producer X_P
        Perform producing
            i. The producer scans at zero degree and then randomly selects three
               points in the reachable scanning area using (2) – (4).
            ii. The producer finds the best point among three points. If the best point
                has a better fitness than its current one, then the producer will move
                to this point. Otherwise, it will stay at its current position and turn its
                head to a new angle using (5).
            iii. If the producer cannot find a promising area after a couple of
                 generations, it will turn its back into the zero degree using (6).
        Perform scrounging:
            Randomly select 80% of rest of group as scroungers.
        Perform ranging:
            The remained members selected as ranges:
                i. Generate a random head angle φ_i using (5);
                ii. Choose a random distance l_i using (8);
                iii. Move to the new position using (9);
        Calculate fitness f(X_i)
        Update parameters
            Calculate success status S_i using (14);
            Calculate success rate P using (15);
            Update maximum pursuit distance l_max using (17);
        i++;
    End For
    k++;
End For;
```

Figure 1. Pseudo-code of the SRGSO algorithm.

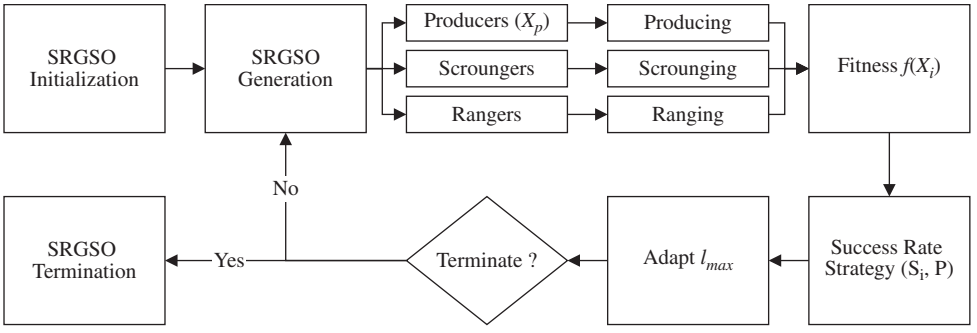

Figure 2. Schematic view of the SRGSO algorithm.

maximum pursuit distance $l_{\max}$ by means of success rate P. Finally, if the termination condition is not met, the algorithm proceeds to another generation loop.

4. Simulation results

4.1 Experimental settings

In order to evaluate the performance of the SRGSO algorithm, a set of 12 benchmark functions (He et al., 2009b) is used, which is given in Table 1. Benchmark functions are tested on two-dimensional scales of 30 and 300. Also, this set includes two types of benchmark functions:

(6) Unimodal functions with simple landscape (f_1–f_7)
(7) Multimodal functions with many local minima (f_8–f_{12})

GSO and SRGSO are tested on 12 benchmark functions with the following parameters. The initial population is drawn from a uniform distribution. The group includes 48 members in which 1 of them is the producer, about 80% of them are scroungers and about 20% of them are rangers. The initial head angle φ^0 and constant a are $\varphi^0 = ((\pi/4)_1, \ldots, (\pi/4)_n)$and $a = \text{round}(\sqrt{n+1})$, respectively, where n is the dimension of the problem. Furthermore, The maximum pursuit angle $\theta_{\max}$ and the maximum turning angle $a_{\max}$ are $\theta_{\max} = \pi/a^2$ and $a_{\max} = \theta_{\max}/2$, respectively. In SRGSO, the maximum pursuit distance $l_{\max}$ is calculated using Equations (14)–(17) where $w_{\min} = 0.4$ and $w_{\max} = 0.9$. Also, in GSO, the $l_{\max}$ is calculated as

$$l_{\max} = \|U - L\| = \sqrt{\sum_{i=1}^{n} (U_i - L_i)^2}, \tag{18}$$

where L_i and U_i are, respectively, the lower and upper bounds for the ith dimension. Also, the SRGSO algorithm is compared with GSO, PSO, evolutionary programming (EP), evolutionary strategy (ES), differential evolution (DE), artificial bee colony (ABC) and GA. The parameter settings for GSO, PSO, EP, ES, and GA are set as recommended by He et al. (2009b) and for DE and ABC based on Lam, Li, and Yu (2012) and Li, Niu, and Xiao (2012), respectively.

4.2 Experiment 1: general-scale benchmark functions (30D)

Table 2 shows the performance comparison of GA, PSO, EP, ES, GSO, DE, ABC and SRGSO algorithms while optimising 12 benchmarks (f_1–f_{12}) in 30-dimensional problems. The SRGSO algorithm could attain the best results in all unimodal benchmarks (f_1–f_7), and especially could

Table 1. Benchmark functions, where n is the dimensionality of function and $f_{\min}$ is the optimum of function.

No.	Name	Function	n	Range	$f_{\min}$
1	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
2	Schwefel 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
3	Schwefel 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	$[-100, 100]$	0
4	Schwefel 2.21	$f_4(x) = \max \{ \lvert x_i \rvert, 1 \le i \le n \}$	30	$[-100, 100]$	0
5	Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	30	$[-30, 30]$	0
6	Step	$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	$[-100, 100]$	0
7	Quartic	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	30	$[-1.28, 1.28]$	0
8	Schwefel 2.26	$f_8(x) = -\sum_{i=1}^{n} \left(x_i \sin \left(\sqrt{\lvert x_i \rvert} \right) \right)$	30 300	$[-500, 500]$	-12569.5
9	Rastrigin	$f_9(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cos(2\pi x_i) + 10)^2$	30 300	$[-5.12, 5.12]$	0
10	Ackley	$f_{10} = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos 2\pi x_i \right) + 20 + e$	30 300	$[-32, 32]$	0
11	Griewank	$f_{11} = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n-1} \cos\left(\frac{x_i - 100}{\sqrt{i}} \right) + 1$	30 300	$[-600, 600]$	0
12	Penalized	$f_{12}(x) = \frac{\pi}{n} \left\{ \begin{array}{l} 10 \sin^2(\pi y_1) \\ + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] \\ + (y_n - 1)^2 \end{array} \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$ $y_i = 1 + \frac{1}{4}(x_i + 1)$	30 300	$[-50, 50]$	0

Table 2. Mean and standard deviation of GA (He et al., 2009b), PSO (He et al., 2009b), EP (He et al., 2009b), ES (He et al., 2009b), DE (Lam et al., 2012), ABC (Li et al., 2012), GSO (He et al., 2009b) and SRGSO algorithms on benchmark functions f_1–f_{12}.

	Function			
Optimiser	f_1	f_2	f_3	f_4
GA	3.17E + 00 ± 1.66E + 00	5.77E − 01 ± 1.31E − 01	9.75E + 03 ± 2.59E + 03	7.96E + 00 ± 1.51E + 00
PSO	3.69E − 37 ± 2.45E − 36	2.92E − 24 ± 1.14E − 23	1.20E − 03 ± 2.11E − 03	4.12E − 01 ± 2.50E − 01
EP	2.20E − 04 ± 5.90E − 04	2.60E − 03 ± 1.70E − 04	5.00E − 02 ± 6.60E − 02	2.00E + 00 ± 1.20E + 00
ES	3.40E − 05 ± 8.60E − 06	2.10E − 02 ± 2.20E − 03	1.30E − 04 ± 8.50E − 05	3.50E − 01 ± 4.20E − 01
DE	6.57E − 06 ± 1.13E − 06	2.89E − 04 ± 2.51E − 05	1.12E + 04 ± 1.55E + 03	5.79E + 00 ± 4.55E − 01
ABC	3.62E − 09 ± 5.85E − 09	5.11E − 06 ± 2.23E − 06	1.12E − 04 ± 3.00E − 03	2.45E + 01 ± 5.65E + 00
GSO	1.94E − 08 ± 1.16E − 08	3.70E − 05 ± 8.62E − 05	5.78E + 00 ± 3.68E + 00	1.08E − 01 ± 4.00E − 02
SRGSO	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**
	Function			
Optimiser	f_5	f_6	f_7	f_8
GA	3.39E + 02 ± 3.61E + 02	3.70E + 00 ± 1.95E + 00	1.05E − 01 ± 3.62E − 02	−12566.1 ± 2.11E + 00
PSO	3.74E + 01 ± 3.21E + 01	1.46E − 01 ± 4.18E − 01	9.90E − 03 ± 3.54E − 02	−96597.0 ± 4.64E + 02
EP	6.17E + 00 ± 1.36E + 01	5.77E + 02 ± 11.25E + 02	1.80E − 02 ± 6.40E − 03	−79171.1 ± 6.34E + 02
ES	6.69E + 00 ± 1.44E + 01	4.11E + 02 ± 6.95E + 02	3.00E − 02 ± 1.50E − 02	−75499.0 ± 6.31E + 02
DE	9.33E + 01 ± 1.73E + 01	7.00E − 02 ± 2.93E − 01	3.96E − 02 ± 7.83E − 03	−12570.0 ± 2.33E − 05
ABC	4.55E + 00 ± 4.87E + 00	**2.49E − 09 ± 3.68E − 09**	1.56E − 01 ± 4.65E − 02	−12130.3 ± 1.59E + 02
GSO	4.98E + 01 ± 3.02E + 01	1.60E − 02 ± 1.33E − 01	7.38E − 02 ± 9.26E − 02	**−12569.5 ± 2.21E − 02**
SRGSO	**2.89E + 00 ± 3.83E − 02**	2.21E + 00 ± 1.78E + 00	**2.24E − 04 ± 2.17E − 04**	−25821.8 ± 1.23E + 02
	Function			
Optimiser	f_9	f_{10}	f_{11}	f_{12}
GA	6.51E − 01 ± 3.59E − 01	8.68E − 01 ± 2.81E − 01	1.00E + 00 ± 6.75E − 02	4.36E − 02 ± 5.06E − 02
PSO	2.08E + 01 ± 5.94E + 00	1.34E − 03 ± 4.23E − 02	2.32E − 01 ± 4.43E − 01	3.95E − 02 ± 9.14E − 02
EP	4.06E − 02 ± 1.20E − 02	1.80E − 02 ± = 2.10E − 02	1.60E − 02 ± 2.20E − 02	9.20E − 06 ± 6.13E − 05
ES	7.08E + 00 ± 2.14E − 01	9.07E + 00 ± 2.84E + 00	0.38E + 00 ± 0.77E + 00	1.18E + 00 ± 1.87E + 00
DE	7.26E − 05 ± 3.37E − 05	7.13E − 04 ± 6.19E − 05	9.05E − 05 ± 3.40E − 05	1.88E − 07 ± 4.26E − 08
ABC	4.53E − 01 ± 5.14E − 01	2.71E − 05 ± 2.13E − 05	3.81E − 03 ± 8.45E − 03	1.18E − 10 ± 2.55E − 10
GSO	1.02E + 00 ± 9.51E − 01	2.65E − 05 ± 3.08E − 05	3.08E − 02 ± 3.09E − 02	**2.76E − 11 ± 9.17E − 11**
SRGSO	**0.00E + 00 ± 0.00E + 00**	**8.88E − 16 ± 0.00E + 00**	**0.00E + 00 ± 0.00E + 00**	7.20E − 02 ± 1.23E − 02

Note: The best result is highlighted in boldface.

reach the global optimum in f_1–f_4 benchmarks. Moreover, it could reach the best solution in three out of five multimodal benchmarks (f_8–f_{12}).

From Table 2, the SRGSO algorithm achieves the best results in unimodal benchmarks that have a little number of local optima and simple structure. Also, the PSO algorithm could achieve good results in this type of benchmarks. Because each particle of the PSO population absorbs to its personal best position (*pbest*) and population's global best position (*gbest*), PSO easily balances between global and local searches and achieves good results in unimodal benchmarks. However, in complex multimodal benchmarks, PSO's absorbing strategy fails and it traps in the local optima. PSO is followed by EP and ES where these two algorithms comply with a deterministic evolutionary approach based on individual rankings. Moreover, GA gets the worst results in this experiment, and it suffers from premature convergence. It is worth noting that GA is specifically designed for solving discrete optimisation problems and not for continues ones.

In the GSO algorithm, the maximum pursuit distance $l_{\max}$ determines the step length of the producer while moving towards the local optimum. From Table 2, the success rate strategy of SRGSO boosts its performance and it performs better than GSO in 10 out of 12 benchmark functions. This adaptive strategy draws a comprehensive perception from the success status of all members and enforces it to the next-generation members through adjusting the $l_{\max}$ parameter.

4.3 Experiment 2: statistical significance on benchmark function (30D)

In order to investigate the statistical significance between the SRGSO and other counterpart algorithms, several multi-comparison statistical tests (Friedman and Iman–Davenport) with 95% confidence level ($\alpha = 05$) are conducted (García, Molina, Lozano, & Herrera, 2009). As a statistical analysis, Friedman's test is first applied to obtain rankings. To obtain the adjusted p-values for each comparison between the control algorithm (the best-performing one) and the other algorithms, Bonferroni–Dunn (Bon-Dunn), Holm and Hochberg tests are conducted as post-hoc methods (if significant differences are detected). The rankings obtained by Friedman's test are presented in Table 3. According to the results of statistical significance in Table 3, one can conclude that SRGSO is ranked first. The p-value computed by Friedman's test is 2.1989E − 04, which is below the significance interval of 95% ($\alpha = 0.05$). Thus, a significant difference exists among the observed results. Post-hoc methods (Bonferroni–Dunn, Holm and Hochberg tests) are also performed to obtain the adjusted p-values. Table 4 shows the adjusted p-values of the Bonferroni–Dunn, Holm and Hochberg tests. The adjusted p-value in Table 4 indicates that SRGSO outperforms the other algorithms (PSO, GSO, DE, EP, ES and GA) with $\alpha = 0.05$.

4.4 Experiment 3: large-scale benchmark function (300D)

There are numerous parameters and constraints involved in real-world problems. The large-scale optimisation problems try to model the complex real-world problems. This experiment aims to

Table 3. Average ranking of Friedman's test of the comparing algorithms (GA, PSO, EP, ES, DE, ABC, GSO and SRGSO) on benchmark functions f_1–f_{12}.

	Optimiser							
Test results	SRGSO	ABC	PSO	GSO	DE	EP	ES	GA
Average ranking	2.083	3.916	3.999	4.166	4.583	4.666	5.583	7.000
Ranking	1	2	3	4	5	6	7	8

Table 4. *p-values* of the comparing algorithms (GA, PSO, EP, ES, DE, ABC, GSO) on benchmark functions f_1–f_{12}.

	Test results			
SRGSO vs.	Unadjusted *p*-value	Bon-Dunn *p*-value	Holm *p*-value	Hochberg *p*-value
GA	8.8030E − 07	6.1621E + 00	6.1621E + 00	6.1621E + 00
ES	4.6525E − 04	3.2568E − 03	2.7915E − 03	2.7915E − 03
EP	9.7785E − 03	6.8495E − 02	4.8925E − 02	4.8925E − 02
DE	1.2419E − 02	8.6935E − 02	4.9677E − 02	4.9677E − 02
GSO	3.7220E − 02	2.6054E − 01	1.1166E − 01	6.6753E − 02
PSO	5.5280E − 02	3.8696E − 01	1.1166E − 01	6.6753E − 02
ABC	6.6753E − 02	4.6727E − 01	1.1166E − 01	6.6753E − 02

compare the performance of SRGSO in contrast to its counterparts in large-scale (300 dimensions) benchmark functions (f_8–f_{12}). Table 5 shows the simulation results of GA, PSO, EP, ES GSO and SRGSO on 300-dimensional problems. The GSO algorithm could achieve the best results in f_8, f_9 and f_{12}, while the SRGSO algorithm could achieve the best results in f_{10} and f_{11}.

Curse of dimensionality describes the exponential rapid growth in the difficulty of problems as the number of dimensions increases. A scalable and efficient optimiser could maintain its performance even in large-scale problems. Due to this phenomenon affecting the performance of optimisation algorithms in large-scale optimisation problems, the performance of GA, PSO, EP and ES deteriorates heavily.

Table 5 shows that, in f_{10} and f_{11}, SRGSO achieves exactly the same performance as shown in Table 2. This result rejects the hypothesis of the curse of dimensionality on SRGSO.

4.5 Discussion of results

Basically, the experiments consist of two types of benchmark functions including unimodal (f_1–f_8) and multimodal (f_9–f_{10}). The following are notable remarks derived from Tables 1 and 5:

(1) Sphere (f_1) has no local minima except for the global one. PSO and SRGSO could easily achieve the best results in this benchmark, while SRGSO digs out its global optimum.
(2) Schwefel 2.22 (f_2) has a product term of absolute value with the global optimum placed at zero. PSO and SRGSO efficiently optimise this function. By using the success rate of the group, SRGSO can drill down into the benchmark as much as possible.

Table 5. Mean of GA (He et al., 2009b), PSO (He et al., 2009b), EP (He et al., 2009b), ES (He et al., 2009b), GSO (He et al., 2009b) and SRGSO algorithms on benchmark functions f_8–f_{12}.

	Function				
Optimiser	f_8	f_9	f_{10}	f_{11}	f_{12}
GA	− 117275.3	121.3	6.24	0.37	52.82
PSO	− 87449.2	427.1	3.95E − 06	1.81	14.56
EP	− 78311.9	383.3	0.2946	2.8244E − 02	39.3
ES	− 66531.3	583.2	9.6243	0.1583	3093.2
GSO	**− 125351.2**	**98.9**	1.35E − 03	1.82E − 07	**8.26E − 08**
SRGSO	− 147124.2	654.2	**8.88E − 16**	**0.00E + 00**	7.09E − 02

Note: The best result is highlighted in boldface.

(3) Schwefel 1.2 (f_3) has a convex structure. A gradient descent or a local optimiser could easily solve this problem, but slow convergence and weak local search of GA failed GA to find the local optimum. ES uses some kind of elitist selection that showed promising results for this benchmark. SRGSO is capable of finding the optimal solution of this function efficiently.
(4) Schwefel 2.21 (f_4) has an inverted pyramid landscape. It returns the maximum absolute value with global optimum at 0. Like previous listed functions, SRGSO could easily locate the local minimum of this function.
(5) Rosenbrock (f_5) has a non-convex, parabolic-shaped landscape in which its global minimum is inside a long, narrow and flat valley. The numerical results of Table 2 shows that SRGSO performs best in optimising this benchmark due to maintaining group diversity while crossing the Rosenbrock's valley. EP emphasises on the behavioural connection between parents and their offspring besides applying genetic operators in GA. This algorithm outperformed other optimisers except for SRGSO.
(6) Step (f_6) is a piecewise constant function. In 30-dimensional step function, ABC obtained the best result among all optimisers because of its intuitive explanation capabilities. ABC tries to balance the exploitation capabilities by using the combination of employed and onlooker bees, and the combination of onlooker and scout bees, respectively.
(7) Quartic (f_7) has a polynomial term to the power of four which is summed by a noisy random number. Results show that SRGSO performed significantly better on the Quartic benchmark with a success rate strategy than GSO with three points sampling strategy.
(8) Schwefel 2.26 (f_8) has a sine term that covers its surface with several local minima. The algorithm that considers different characteristics like separability, multimodality and regularity is effective for optimising this function. The use of social foraging model implemented in GSO provides a simple and robust framework to optimise the general- and large-scale Schwefel 2.26 problem. SRGSO may fail while optimising shifted problems like this.
(9) Rastrigin (f_9) has a cosine term that covers its non-convex, nonlinear landscape with a large number of local minima whose values increases with the distance to the global minimum. The SRGSO algorithm outperformed the other algorithms in the 30-dimensional Rastrigin and lagged in the 300-dimensional Rastrigin. After SRGSO, DE obtains the best result in this benchmark. DE tries to produce new solutions by combining existing candidates and its iterative simple formula.
(10) Ackley (f_{10}) has an exponential term that covers its surface with numerous local minima. The gradient descent algorithms will be easily trapped in a local optimum. SRGSO gains a strong local search by adapting $l_{\max}$ parameter of producer members, and an agile global search by employing $l_{\max}$ parameter of ranger members.
(11) Griewank (f_{11}) has a product term that makes its dimensions non-separable. The optimiser that observes the problem dimensions independently will be failed while optimizing this function. The $l_{\max}$ parameter is limited to the performance of GSO while the adaptive step-controlling mechanism boosts the SRGSO performance in both 30- and 300-dimensional Griewank problems.
(12) Penalised (f_{12}) has a sine term summed by a sine series. SRGSO is surpassed by GSO on penalised function. The GSO's group movement could efficiently solve this problem.

From Table 2, SRGSO shows that it has a significantly tolerable margin of error for the estimated standard deviations, because its ranger members perform dispersion through problems. Note that this is not strictly said to withdraw the results reported in Section 4.3, inasmuch as the statistical significance aims at determining the meaning and repetition of results.

The term 'independently/separability' points to the fact that two dimensions of benchmark function are independent/separated from each other in terms of dimension independence/interaction, while the term 'dependability' points to the fact that two dimensions of benchmark function are dependable to each other in terms of dimension dependency. From Table 5, a major factor that contributes to the difficulty of large-scale benchmark functions is the interaction between dimensions. Two dimensions are separable if they cannot be optimised independently. This phenomenon is commonly referred to as 'non-separability'. The combination of novel search strategy of GSO with success rate strategy could produce appropriate group members that search among promising subsets of benchmark dimensions.

A brief summary of positive and negative effects of success rate strategy are reported in the following:

(1) According to Table 2, compared with GSO, success rate strategy improves the performance of SRGSO in both simple unimodal problems (f_1,f_2,f_3,f_4,f_5,f_7) and complex multimodal problems (f_8,f_9,f_{10},f_{11}). SRGSO could maintain group diversity inasmuch as the group success status simply reinforces the probability of producing elite members.
(2) According to Tables 2 and 3, compared with GSO, SRGSO cannot perfectly optimise the Schwefel 2.26 (f_8) problem. This observation shows that when the global optimum of f_8 is shifted to -12569.5, SRGSO fails to detect this rearrangement of the problem's structure. This may happen due to the lack of SRGSO's perception for properties of shifted problems inasmuch as this optimiser has a single-parameter tuning strategy.

From Tables 2–5, one can conclude that the influence of selecting a fixed value for $l_{\max}$ parameter is limited to the performance of GSO. Finally, SRGSO is a robust and stable algorithm that optimises 10 out of 12 general-scale problems (30 dimensional) and 2 out of 5 large-scale problems (300 dimensional).

5. Conclusion

In this paper, we develop a modified GSO algorithm called the SRGSO algorithm. Similar to the GSO algorithm, the SRGSO contains three types of members: (i) producer members for finding the local optima, (ii) scrounger members that join the local optima found by the producer and, finally, (iii) ranger members that are responsible for discovering new promising regions. In the GSO algorithm, the scanning ability to synthesise the surrounding environment is the key characteristic of the producer member. The GSO algorithm simply uses a fixed constant step length in scanning mechanism, while, in the SRGSO, we improve the scanning mechanism through computing the step length ($l_{\max}$) adaptively.

The SRGSO algorithm uses the success rate strategy in order to obtain a proper feedback from the group movements to adjust the $l_{\max}$ parameter. The SRGSO algorithm makes the following contributions: (i) develops a simple adaptive parameter-adjusting strategy with low overload, (ii) tunes the evolution-aware parameter, (iii) balances the exploration and exploitation abilities and, finally, (iv) maintains the group diversity.

We test the SRGSO algorithm for 12 benchmark functions. Simulation results show that the SRGSO algorithm easily optimises the unimodal and multimodal benchmark functions. Furthermore, it could quickly and efficiently solve the large-scale benchmark functions. Along with these advantages, we should mention that the SRGSO algorithm performs weak in solving general shifted problems (e.g. Schwefel 2.26).

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which improved the paper.

Notes

1. Email: mdhassanzd@aut.ac.ir, hasanzadeh@ictrc.ir
2. Email: sadeghi@pnuhp.ac.ir
3. Email: mmeybodi@aut.ac.ir

References

Barnard, C. J., & Sibly, R. M. (1981). Producers and scroungers: A general model and its application to captive flocks of house sparrows. *Animal Behaviour*, *29*, 543–550.

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies – A comprehensive introduction. *Natural Computing*, *1*, 3–52.

Chalmers, D. J., French, R. M., & Hofstadter, D. R. (1992). High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental & Theoretical Artificial Intelligence*, *4*, 185–211.

Estahbanati, M. J. (2014). Hybrid probabilistic-harmony search algorithm methodology in generation scheduling problem. *Journal of Experimental & Theoretical Artificial Intelligence*, *26*, 283–296.

García, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, *15*, 617–644.

Guo, C. X., Zhan, J. P., & Wu, Q. H. (2012). Dynamic economic emission dispatch based on group search optimizer with multiple producers. *Electric Power Systems Research*, *86*, 8–16.

Hagh, M. T., Teimourzadeh, S., Alipour, M., & Aliasghary, P. (2014). Improved group search optimization method for solving CHPED in large scale power systems. *Energy Conversion and Management*, *80*, 446–456.

Hasanzadeh, M., & Meybodi, M. R. (2013). Deployment of gLite middleware: An E-Science grid infrastructure. In *2013 21st Iranian conference on electrical engineering (ICEE)* (pp. 1–6). Tehran: IEEE.

Hasanzadeh, M., & Meybodi, M. R. (2014). Grid resource discovery based on distributed learning automata. *Computing*, *96*, 909–922.

Hasanzadeh, M., Meybodi, M. R., & Ebadzadeh, M. M. (2012). A robust heuristic algorithm for cooperative particle swarm optimizer: A learning automata approach. In *2012 20th Iranian conference on electrical engineering (ICEE)*. Tehran: IEEE.

Hasanzadeh, M., Meybodi, M. R., & Ebadzadeh, M. M. (2013). Adaptive cooperative particle swarm optimizer. *Applied Intelligence*, *39*, 397–420.

Hasanzadeh, M., Meybodi, M. R., & Ghidary, S. S. (2011). Improving learning automata based particle swarm: An optimization algorithm. In *2011 IEEE 12th International symposium on computational intelligence and informatics (CINTI)* (pp. 291–296). Budapest: IEEE.

He, S., Cooper, H. J., Ward, D. G., Yao, X., & Heath, J. K. (2012). Analysis of premalignant pancreatic cancer mass spectrometry data for biomarker selection using a group search optimizer. *Transactions of the Institute of Measurement and Control*, *34*, 668–676.

He, S., Wu, Q. H., & Saunders, J. R. (2009a). Breast cancer diagnosis using an artificial neural network trained by group search optimizer. *Transactions of the Institute of Measurement and Control*, *31*, 517–531.

He, S., Wu, Q. H., & Saunders, J. R. (2009b). Group search optimizer: An optimization algorithm inspired by animal searching behavior. *IEEE Transactions on Evolutionary Computation*, *13*, 973–990.

Javadi, M. S., Saniei, M., & Rajabi Mashhadi, H. (2014). An augmented NSGA-II technique with virtual database to solve the composite generation and transmission expansion planning problem. *Journal of Experimental & Theoretical Artificial Intelligence*, *26*, 211–234.

Kamarian, S., Yas, M. H., Pourasghar, A., & Daghagh, M. (2014). Application of firefly algorithm and ANFIS for optimisation of functionally graded beams. *Journal of Experimental & Theoretical Artificial Intelligence*, *26*, 197–209.

Kang, Q., Lan, T., Yan, Y., Wang, L., & Wu, Q. (2012). Group search optimizer based optimal location and capacity of distributed generations. *Neurocomputing*, *78*, 55–63.

Kavousi-Fard, A., & Kavousi-Fard, F. (2013). A new hybrid correction method for short-term load forecasting based on ARIMA, SVR and CSA. *Journal of Experimental & Theoretical Artificial Intelligence*, *25*, 559–574.

Kordestani, J. K., Rezvanian, A., & Meybodi, M. R. (2014). CDEPSO: A bi-population hybrid approach for dynamic optimization problems. *Applied Intelligence*, *40*, 682–694.

Lam, A. Y., Li, V. O., & Yu, J. J. (2012). Real-coded chemical reaction optimization. *IEEE Transactions on Evolutionary Computation*, *16*, 339–353.

Li, G., Niu, P., & Xiao, X. (2012). Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. *Applied Soft Computing*, *12*, 320–332.

Li, L., Xu, X.-T., Liu, F., & Wu, Q. H. (2010). The group search optimizer and its application to truss structure design. *Advances in Structural Engineering*, *13*, 43–52.

Manurung, R., Ritchie, G., & Thompson, H. (2012). Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, *24*, 43–64.

Nabizadeh, S., Rezvanian, A., & Meybodi, M. R. (2012). Tracking extrema in dynamic environment using multi-swarm cellular PSO with local search. *International Journal of Electronics & Informatics*, *1*, 29–37.

Nian, X., Wang, Z., & Qian, F. (2013). A hybrid algorithm based on differential evolution and group search optimization and its application on ethylene cracking furnace. *Chinese Journal of Chemical Engineering*, *21*, 537–543.

Rezaee Jordehi, A., & Jasni, J. (2013). Parameter selection in particle swarm optimisation: A survey. *Journal of Experimental & Theoretical Artificial Intelligence*, *25*, 527–542.

Rezvanian, A., & Meybodi, M. R. (2010). An adaptive mutation operator for artificial immune network using learning automata in dynamic environments. In *2010 2nd world congress on nature and biologically inspired computing (NaBIC)* (pp. 479–483). Kitakyushu: IEEE.

Shen, H., Zhu, Y., Niu, B., & Wu, Q. H. (2009). An improved group search optimizer for mechanical design optimization problems. *Progress in Natural Science*, *19*, 91–97.

Shyu, S. J., Yin, P.-Y., Lin, B. M., & Haouari, M. (2003). Ant-tree: An ant colony optimization approach to the generalized minimum spanning tree problem. *Journal of Experimental & Theoretical Artificial Intelligence*, *15*, 103–112.

Soleimani-Pouri, M., Rezvanian, A., & Meybodi, M. R. (2012a). Finding a maximum clique using ant colony optimization and particle swarm optimization in social networks. In *Proceedings of the 2012 international conference on advances in social networks analysis and mining (ASONAM 2012)* (pp. 58–61). Istanbul: IEEE Computer Society. Retrieved from http://dx.doi.org/10.1109/ASONAM.2012.20.

Soleimani-Pouri, M., Rezvanian, A., & Meybodi, M. R. (2012b). Solving maximum clique problem in stochastic graphs using learning automata. In *2012 4th international conference on computational aspects of social networks (CASoN)* (pp. 115–119). Sao Carlos: IEEE.

Soleimani-pouri, M., Rezvanian, A., & Meybodi, M. R. (2014). An ant based particle swarm optimization algorithm for maximum clique problem in social networks. In F. Can, T. Özyer, & F. Polat (Eds.), *State of the art applications of social network analysis* (pp. 295–304). Switzerland: Springer International Publishing.

Wang, L., Zhong, X., & Liu, M. (2012). A novel group search optimizer for multi-objective optimization. *Expert Systems with Applications*, *39*, 2939–2946.

Yan, X., & Shi, H. (2011). A hybrid algorithm based on particle swarm optimization and group search optimization. In *2011 7th international conference on natural computation (ICNC)* (Vol. *1*, pp. 13–17). Shanghai: IEEE.

Yan, X., Yang, W., & Shi, H. (2012). A group search optimization based on improved small world and its application on neural network training in ammonia synthesis. *Neurocomputing*, *97*, 94–107.

Yao, J., Cui, Z., Wei, Z., & Tan, Y. (2011). Hybrid group search optimiser with quadratic interpolation method and its application. *International Journal of Wireless and Mobile Computing*, *5*, 98–106.

Yas, M. H., Kamarian, S., & Pourasghar, A. (2014). Application of imperialist competitive algorithm and neural networks to optimise the volume fraction of three-parameter functionally graded beams. *Journal of Experimental & Theoretical Artificial Intelligence*, *26*(1), 1–12.