Neural Comput & Applic (2014) 25:1423–1435
DOI 10.1007/s00521-014-1629-6

ORIGINAL ARTICLE

Binary optimization using hybrid particle swarm optimization and gravitational search algorithm

Seyedali Mirjalili · Gai-Ge Wang · Leandro dos S. Coelho

Received: 9 January 2014 / Accepted: 15 May 2014 / Published online: 11 June 2014

Abstract The PSOGSA is a novel hybrid optimization algorithm, combining strengths of both particle swarm optimization (PSO) and gravitational search algorithm (GSA). It has been proven that this algorithm outperforms both PSO and GSA in terms of improved exploration and exploitation. The original version of this algorithm is well suited for problems with continuous search space. Some problems, however, have binary parameters. This paper proposes a binary version of hybrid PSOGSA called BPSOGSA to solve these kinds of optimization problems. The paper also considers integration of adaptive values to further balance exploration and exploitation of BPSOGSA. In order to evaluate the efficiencies of the proposed binary algorithm, 22 benchmark functions are employed and divided into three groups: unimodal, multimodal, and composite. The experimental results confirm better performance of BPSOGSA compared with binary gravitational search algorithm (BGSA), binary particle swarm optimization (BPSO), and genetic algorithm in terms of avoiding local minima and convergence rate.

S. Mirjalili (✉)
School of Information and Communication Technology, Griffith University, Nathan, Brisbane, QLD 4111, Australia
e-mail: seyedali.mirjalili@griffithuni.edu.au

G.-G. Wang
School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, Jiangsu, China

L. S. Coelho
Industrial and Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR), Curitiba, Parana, Brazil

L. S. Coelho
Electrical Engineering Graduate Program (PPGEE), Department of Electrical Engineering, Polytechnic Center, Federal University of Parana (UFPR), Curitiba, Parana, Brazil

Keywords Binary optimization · Binary algorithms · PSOGSA · Particle swarm optimization · Gravitational search algorithm

1 Introduction

Recently, nature-inspired stochastic optimization techniques have received much attention. Such optimization mostly mimics social/individual behavior of a group of animal or natural phenomena. Such techniques start the optimization process by creating a set of random solutions and improve them as candidate solutions for a particular problem. Due to the superior performance of such techniques compared to mathematical optimization approaches, the application of stochastic optimization methods can be found in different fields. It has been logically proven by the well-known No Free Lunch (NFL) theorem that there is no optimization technique which for solving all optimization problems [1]. This theorem therefore allows researchers to propose new optimization techniques or improve the current algorithms for solving a wider range of problems. Some of the most popular and well-known algorithms are particle swarm optimization (PSO) [2], genetic algorithm (GA) [3], differential evolution algorithm (DE) [4], simulated annealing (SA) [5], harmony search (HS) [6], ant colony optimization (ACO) [7], gravitational search algorithm (GSA) [8], biogeography-based optimization algorithm (BBO) [9–12], grey wolf optimizer (GWO) [13], and Krill Herd (KH) algorithm [14–20].

The literature shows that hybridizing stochastic optimization techniques is one of the ways for designing superior algorithms and using the advantages of multiple algorithms when solving optimization problems [21–28]. The PSOGSA algorithm is a novel hybrid of PSO and GSA, which was proposed in 2010 [29]. It has been proven that this algorithm has a good performance in solving optimization problems. In this algorithm, the search process is carried out by agents, which mimics the behavior of GSA in the exploration phase and PSO in exploitation phase. In fact, this algorithm was proposed to alleviate slow exploitation of the GSA algorithm, which was identified to be one of its main drawbacks.

Since the proposal of the GSA algorithm, many studies have been aimed for improving the performance of this algorithm. In 2011, Hatamlou et al. [30] hybridized GSA with another heuristic method for improving the solutions obtained by GSA for solving clustering problems. In 2012, an position-based learning GSA [31] and Immune Gravitation Optimization Algorithm (IGOA) [32] were employed to improve the convergence speed of GSA. The latter study incorporated the characteristics of antibody diversity and vaccination to the GSA algorithm. Similarly to PSOGSA [29], social thinking and individual thinking of PSO were integrated to GSA for solving a continuous problem called parameter identification of hydraulic turbine governing system [33].

The GSA algorithm was modified by Rashedi et al. [34] to solve binary problems as well. Since BGSA utilizes the same concepts of GSA, however, it suffers from the same drawbacks of GSA mentioned above. Needless to say, the binary version of improved GSA algorithms in the literature would also have superior performance compared to BGSA if they design in a way that uses the same concepts of continuous optimization for binary optimization. This motivates the authors to extend PSOGSA algorithm as one of the best improved GSA algorithms to a binary version and investigate its performance since the original version of PSOGSA has been designed for solving problems with continuous real search spaces (domains) [29].

In real world, there are many optimization problems with discrete binary search spaces (for instance, dimensionally reduction or feature selection in different fields). There are different methods in the literatures to require a continuous algorithm for solving binary problems as well. Some of these methods change the structure of the algorithm, whereas others maintain the algorithm's mechanism in binary search spaces. In the literatures, stochastic optimization algorithms such as HS, DE, PSO, and GSA [35] have been adapted to solve binary problems. The binary HS algorithm utilizes a set of harmony search considerations and pitch adjustment rules to perform binary optimization [36]. In addition, binary DE uses a probability estimation operator in order to solve discrete problems [37]. The mechanism of binary PSO [38] and binary GSA [34] is quite similar since they both use transfer functions to solve binary problems. The difference of BPSO and BGSA compared to other above-mentioned binary optimization methods is that these two algorithms retain the same concepts of position and velocity updating procedures.

In this paper, a binary version of PSOGSA is introduced called BPSOGSA in order to solve binary problems. A transfer function and new positing updating procedure are integrated to BPSOGSA. The paper also considers the proposal of adaptive values for BPSOGSA in order to balance exploration and exploitation.

The rest of the paper is organized as follows. Section 2 presents a brief introduction to hybrid PSOGSA. Section 3 discusses the basic principles of binary version of PSOGSA. The experimental results are demonstrated in Sect. 4. Finally, Sect. 5 concludes the work and suggests some researches for future works.

2 The hybrid PSOGSA

The hybrid PSOGSA algorithm was introduced by Mirjalili and Mohd Hashim [29]. The basic idea of PSOGSA is to combine the ability of social thinking (*gbest*) in PSO with exploration capability of GSA. The PSOGSA algorithm was mathematically modeled as follows:

Similarly to PSO and GSA, every search agent has a position vector reflecting the current position in search spaces as follows:

$$\overrightarrow{X_i} = \left(x_i^1, \ldots, x_i^d\right), \quad i = 1, 2, \ldots, N \tag{2.1}$$

where N is the number of search agents, d is the dimension of the problem, and x_i^d is the position of the ith agent in the d-th dimension.

So the whole population is represented as a matrix as follows:

$$\vec{X} = \begin{bmatrix} x_1^1 & x_1^2 & \ldots & x_1^d \\ x_2^1 & x_2^2 & \ldots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \ldots & x_n^d \end{bmatrix} \tag{2.2}$$

There is another matrix for storing the corresponding fitness value for each search agent as follows:

$$\vec{O} = \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_n \end{bmatrix} \tag{2.3}$$

The optimization process begins with filling out the position matrix by random values. During optimization, the gravitational force from agent j on agent i at a specific time t is defined as follows:

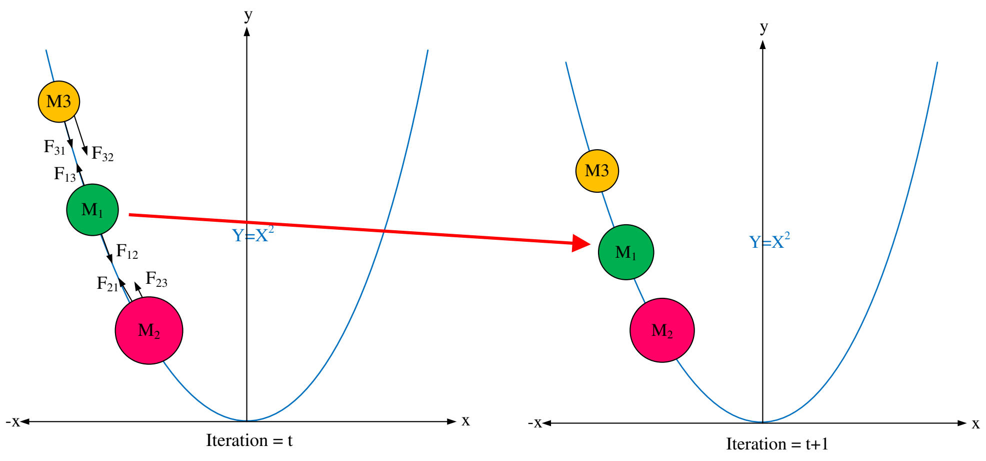

Fig. 1 Operation of GSA's masses in two iterations

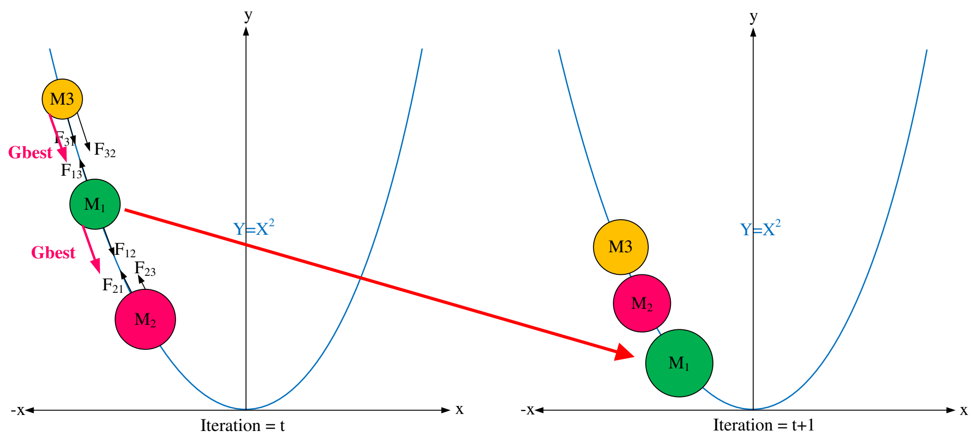

Fig. 2 Operation of PSOGSA's particles in two iterations

```
Generate initial population
Repeat
        Calculate fitness of all agents
        Update gbest and G
        For each search agent:
                -Calculate all gravitational forces
                -Calculate acceleration
                -Update velocity
                -Update position
        End for
Until satisfying the stop criterion
Return gbest
```

Fig. 3 Pseudocode of PSOGSA

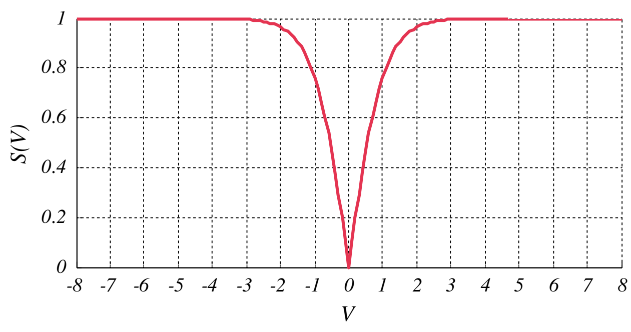

Fig. 4 Tangent hyperbolic transfer function

$$F_{ij}^{d}(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \tag{2.4}$$

where M_{aj} is the active gravitational mass related to agent j, M_{pi} is the passive gravitational mass related to agent i, $G(t)$ is a gravitational constant at time t, ε is a small constant, and $R_{ij}(t)$ is the Euclidian distance between two agents i and j.

The gravitational constant (G) and the Euclidian distance between two agents i and j are calculated as follows:

$$G(t) = G_0 \times \exp(-\alpha \times iter/maxiter) \tag{2.5}$$

$$R_{ij}(t) = X_i(t), X_j(t)_2 \tag{2.6}$$

where α is the descending coefficient, G_0 indicates the initial gravitational constant, *iter* is the current iteration, and *maxiter* shows the maximum number of iterations.

Fig. 5 Pseudocode of BPSOGSA

```
For each search agent
        Create and initialize a D-dimensional vector randomly
End for
Repeat
        Calculate fitness of all agents
        Update gbest and G
        For each particle:
                -Calculate all gravitational forces by equation 2.2
                -Calculate acceleration by equation 2.6
                -Calculate velocity by equation 2.7
                -Calculate probability of changing position vector's element using 3.1
                -Update position vector's elements according to rules in 3.2
        End for
Until the satisfaction of the stop criterion
Return gbest
```

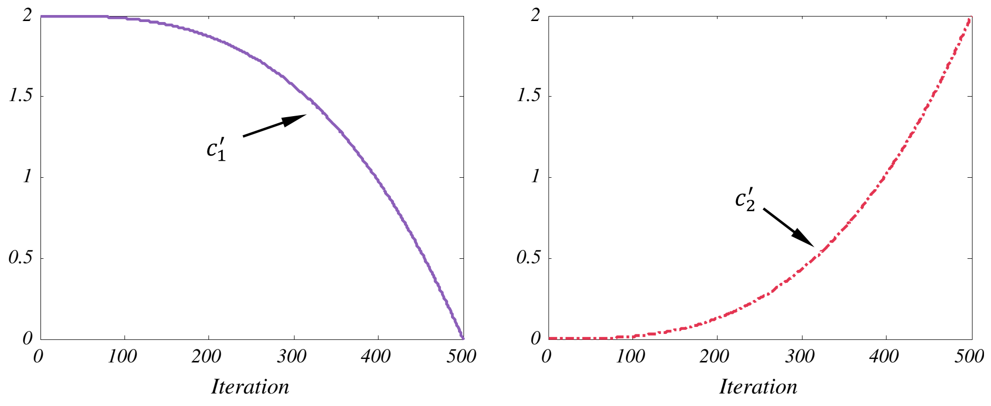

Fig. 6 Adaptive behavior of coefficients

In a problem space with dimension equals to d, the total force that acts on agent i is calculated by the following equation:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d(t) \quad (2.7)$$

where $rand_j$ is a random number generated with uniform distribution in the interval [0, 1].

The law of motion has also been utilized in this algorithm which states that the acceleration of a mass is proportional to the resultant force and inverse of its mass, so the acceleration of all agents are calculated as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \quad (2.8)$$

where d is the dimension of the problem, t is a specific time, and M_{ii} is the inertial mass of agent i.

During optimization, the best obtained solution so far is saved in *gbest* variable inspired by PSO. The reason for using the best solution is that GSA suffers from slow exploitation and deteriorates in final iterations [29, 39, 40]. In GSA, the masses movements are calculated based on their weights, and the weights are directly calculated by the fitness function. Thus, the masses that have better values of fitness function are considered as heavy objects, and consequently, they move slowly. According to the concepts of EA, particles should wander through the search space at initial iterations. Then, after finding a good solution, they have to gather around that solution in order to exploit the best solution. In GSA, during running time, masses become

Table 1 Unimodal benchmark functions

Function	Dim	Range	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	5	[−100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	5	[−10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	5	[−100, 100]	0
$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \leq i \leq n \}$	5	[−100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	5	[−30, 30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	5	[−100, 100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	5	[−1.28, 1.28]	0

Table 2 Multimodal benchmark functions

Function	Dim	Range	$f_{\min}$
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	5	[−500, 500]	−418.9829 × 5
$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	5	[−5.12, 5.12]	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	5	[−32, 32]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	5	[−600, 600]	0
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	5	[−50, 50]	0
$y_i = 1 + \frac{x_i + 1}{4}$			
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$			
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	5	[−50, 50]	0
$f_{14}(x) = -\sum_{i=1}^{n}\sin(x_i).\left(\sin\left(\frac{i.x_i^2}{\pi}\right)\right)^{2m}, \quad m = 10$	5	[0, π]	−4.687
$f_{15}(x) = \left[e^{-\sum_{i=1}^{n}(x_i/\beta)^{2m}} - 2e^{-\sum_{i=1}^{n} x_i^2}\right].\prod_{i=1}^{n}\cos^2 x_i, \quad m = 5$	5	[−20, 20]	−1
$f_{16}(x) = \left\{\left[\sum_{i=1}^{n}\sin^2(x_i)\right] - \exp\left(-\sum_{i=1}^{n} x_i^2\right)\right\}.\exp\left[-\sum_{i=1}^{n}\sin^2\sqrt{\lvert x_i \rvert}\right]$	5	[−10, 10]	−1

Table 3 Composite benchmark functions

Function	Dim	Range	f_min
f₁₇(CF1): $f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, .., 5/100]$	5	[-5,5]	0
f₁₈(CF2): $f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank's Function* $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, .., 5/100]$	5	[-5,5]	0
f₁₉(CF3): $f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank's Function* $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1,1,1,..,1]$	5	[-5,5]	0
f₂₀(CF4): f_1, f_2 = Ackley'sFunction f_3, f_4 = Rastrigin's Function f_5, f_6 = Weierstrass Function f_7, f_8 = Griewank's Function f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	5	[-5,5]	0
f₂₁(CF5): f_1, f_2 = Rastrigin's Function f_3, f_4 = Weierstrass Function f_5, f_6 = Griewank's Function f_7, f_8 = Ackley'sFunction f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1,1,1,..,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	5	[-5,5]	0
f₂₂(CF6): f_1, f_2 = Rastrigin's Function f_3, f_4 = Weierstrass Function f_5, f_6 = Griewank's Function f_7, f_8 = Ackley'sFunction f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	5	[-5,5]	0

Table 4 Initial parameters for BPSOGSA, BGSA, BPSO, and GA

Algorithm	Parameter	Value	Algorithm	Parameter	Value
BPSOGSA	Number of particles	30	BPSO [1]	Number of individuals	30
	c_1'	$(-2t^3/T^3) + 2$		c_1, c_2	2, 2
	c_2'	$(2t^3/T^3)$		W	Decreases linearly from 0.9 to 0.4
	G_0	1			
	α	20		Max generation	500
	Max iterations	500		Stoppig criteria	Max iteration
	Stoppig criteria	Max iteration			
BGSA [2]	Number of masses	30	GA [3]	Number of individuals	30
	G_0	1		Selection	Roulette wheel
	α	20		Crossover(probability)	One-point (0.9)
	Max iterations	500		Mutation(probability)	Uniform (0.005)
	Stoppig criteria	Max iteration		Max generation	500
				Stoppig criteria	Max iteration

T indicates the maximum number of interactions

t is the current iteration

Table 5 Minimization results of unimodal benchmark functions over 10 independent runs

f	BPSOGSA	BGSA	BPSO	GA
f_1				
Ave	**0.753881836**	2,052.005	5.2965	10.0705
Std	**0.744054218**	41.45277	2.7657	24.9445
f_2				
Ave	**0.158447266**	1.325269	0.2292	0.269483
Std	**0.121911192**	0.67277	0.0938	0.23788
f_3				
Ave	45.28667603	509.0988	**22.48915**	555.9039
Std	94.45222722	266.3714	**14.11401**	250.693
f_4				
Ave	2.464062500	7.999219	2.608854	**1.59375**
Std	2.429516395	3.450794	0.838937	**1.21348**
f_5				
Ave	281.4149623	2,620.465	**148.0799**	369.7545
Std	667.8743127	1,735.901	**137.1896**	342.8893
f_6				
Ave	8.093701172	126.6157	**5.492444**	6.984222
Std	17.67056950	88.03597	**3.070168**	7.010388
f_7				
Ave	**0.006396714**	0.023861	0.015542	0.047174
Std	**0.008876364**	0.02688	0.007474	0.043587

heavier and heavier. In the final steps of iterations, masses have almost the same weights due to gathering around a promising solution. They approximately attract each other with the same intensity of gravitational forces. Therefore, they are not able to move toward the best solution quickly. This problem can be seen in Fig. 1. This figure shows a simple one-dimensional problem where the fitness function is $Y = X^2$. This above-mentioned problem is more critical for high dimensional problems.

In order to see how PSOGSA alleviate this drawback, a conceptual model is illustrated in Fig. 2. This figure shows that the hybrid PSOGSA employs the best solution obtained so far for guiding the heavy masses toward the global optimum. Apparently, this method speeds up the overall movement of masses as well, resulting in enhancing exploitation ability of PSOGSA.

The Eq. (2.9) was proposed as follows for combining PSO and GSA:

$$V_i(t+1) = rand \times V_i(t) + c_1' \times ac_i(t) + c_2' \times (gbest - X_i(t)) \quad (2.9)$$

where $V_i(t)$ is the velocity of agent i at iteration t, c_j' is an accelerating factor, *rand* is a random number generated with uniform distribution between 0 and 1, $ac_i(t)$ is the acceleration of agent i at iteration t, and *gbest* is the best obtained solution so far.

In each iteration, the positions of agents are updated as follows:

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2.10)$$

In PSOGSA, at first, all agents are randomly initialized using uniform distribution. Each agent is considered as a candidate solution. After initialization, gravitational force, gravitational constant, and resultant forces among agents are calculated by Eqs. 2.2, 2.3, and 2.5, respectively. After that, the accelerations of particles are defined by Eq. 2.6. In each iteration, the best attained solution should be updated. After calculating the acceleration and updating the best

Table 6 Minimization results of multimodal benchmark functions over 10 independent runs

f	BPSOGSA	BGSA	BPSO	GA
f_8				
Ave	−979.8132	−828.602	**−988.355**	−929.324
Std	25.03774	42.63769	**14.21898**	27.95231
f_9				
Ave	**1.875194**	5.999694	4.977688	2.1896
Std	**1.271683**	2.963102	1.597929	0.8330273
f_{10}				
Ave	**0.541234**	2.947044	2.725568	1.399853
Std	**0.800463**	1.481999	0.472191	1.338105
f_{11}				
Ave	**0.179551**	0.647846	0.3873	0.7067
Std	**0.092974**	0.228547	0.1302	0.3223
f_{12}				
Ave	0.370201	12.66839	0.621354	**0.191197**
Std	0.485135	8.774814	0.388572	**0.244347**
f_{13}				
Ave	0.255321	922.4907	0.444445	**0.193006**
Std	0.305777	2,458.653	0.211701	**0.254864**
f_{14}				
Ave	**−3.902076**	−2.494	−3.6416	−3.88492
Std	**0.446362**	0.126732	0.32452	0.717682
f_{15}				
Ave	−1.66e−107	−1e−112	−0.055483	**−0.474555**
Std	3.48e−107	3.1e−112	0.1351484	**0.4856118**
f_{16}				
Ave	0.0003171	0.002329	**2.95E−04**	0.001575
Std	0.0002562	0.000998	**0.000215**	0.000818

solution, the velocities of all agents can be calculated by Eq. 2.7. Finally, the positions of agents are updated by Eq. 2.8. The process of updating velocities and positions will be stopped when meeting an end criterion. The steps of hybrid PSOGSA are represented in Fig. 3.

It was experimentally proved that PSOGSA is powerful enough to solve optimization problems better than PSO and GSA [29]. However, the original PSOGSA is a continuous algorithm, which is not capable of solving binary problems directly. In the following subsections, the binary version of PSOGSA is proposed and investigated.

3 Binary version of PSOGSA (BPSOGSA)

In the original PSOGSA, agents can continuously move around the search space because of having position vectors with continuous real domain. In order to require the search agents to move in a binary search space, we have to modify position updating (Eq. 2.10). According to [34, 38], a transfer function is also needed to change position of an agent with the probability of its velocity. Transfer functions map the velocities values to the probability values for updating the positions. A set of standard transfer functions can be found in [41]. According to Mirjalili and Lewis [41], a transfer function should be able to provide a high probability of changing the position for a large absolute value of the velocity. In addition, it should present a small probability of changing the position for a small absolute value of the velocity. Moreover, the range of a transfer function should be bounded in the interval of [0, 1] and increased with the increasing of velocity. The utilized function in [34] is presented as Eq. 3.1. This function is also depicted in Fig. 4.

$$S\left(v_{i,j}^k(t)\right) = \left|\tanh\left(v_{i,j}^k(t)\right)\right| \tag{3.1}$$

We use this equation for mapping velocities of agents in BPSOGSA to probabilities of flipping their position vectors' elements. After calculating the probabilities, the agents update their positions based on the presented rules in Eq. 3.2.

$$\text{If } rand < S\left(v_{i,j}^k(t+1)\right) \text{ then } x_{i,j}^k(t+1) = \text{complement}\left(x_{i,j}^k(t)\right) \text{ else } x_{i,j}^k(t+1) = x_{i,j}^k(t) \tag{3.2}$$

Table 7 Minimization results of composite benchmark functions over 10 independent runs

f	BPSOGSA	BGSA	BPSO	GA
f_{17}				
Ave	**89.14436567**	252.8848	194.8523	193.6682
Std	**55.4314206**	43.63499	60.03402	121.9127
f_{18}				
Ave	**95.51982614**	254.0506	146.7613	205.6785
Std	**48.23949857**	36.16261	29.08005	160.9849
f_{19}				
Ave	**152.1242021**	213.2075	445.7764	384.7761
Std	**54.29495933**	63.65201	49.3449	118.0311
f_{20}				
Ave	**186.3464791**	255.0879	479.9867	588.1262
Std	**43.54435273**	50.36717	30.19361	102.3373
f_{21}				
Ave	**150.8872309**	245.235	172.0816	246.3021
Std	**48.09488617**	52.89851	64.2674	183.5344
f_{22}				
Ave	365.0728328	**232.2638**	691.65	914.5375
Std	132.8960595	**27.63692**	149.6255	12.32191

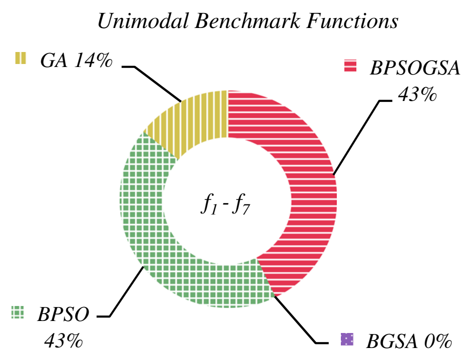

Fig. 7 Statistical results for unimodal benchmark functions

The pseudocode regarding the general steps of BPSOGSA is culminated in Fig. 5.

In should be noted that we utilize adaptive values for c_1' and c_2' in this study. According to Mirjalili and Lewis [42], adding *gbest* to the velocity vector have weakened the exploration phase, since it establishes a permanent element of velocity updating. In order to resolve this issue, we utilize adaptive values for c_1' and c_2' as follows [42]:

$$c_1' = -2\frac{t^3}{T^3} + 2 \tag{3.3}$$

$$c_2' = 2\frac{t^3}{T^3} + 2 \tag{3.4}$$

where t indicates the current iteration and T is the maximum number of iterations.

The behavior of these two variables is illustrated in Fig. 6. We adaptively decrease c_1' and increase c_2' so that the masses tend to accelerate toward the best solution as the algorithm reaches the exploitation phase. Since there is no clear border between the exploration and exploitation phases in evolutionary algorithm, the adaptive method is one of the best options for allowing an algorithm to gradually transit between these two phases.

Theoretically, the BPSOGSA algorithm has the potential to outperform both BPSO and BGSA since it uses the same concepts of PSOGSA. In the following subsection, a comparative study is provided in order to justify the performance of BPSOGSA practically. Note that the source code of this algorithm can be found in http://www.alimirjalili.com/Projects.html.

4 Experimental results and discussion

In this section, 22 benchmark functions dividing into three groups such as unimodal, multimodal, and composite are chosen to evaluate the performance of BPSOGSA [43–47]. Tables 1, 2 and 3 present these benchmark functions, range of search space, and the optimum values.

A comparative study with BGSA, BPSO, and GA is conducted for verifying the performance of BPSOGSA. The dimension of all benchmark functions is set to 5 ($m = 5$). Fifteen bits are used for representing each variable in binary format (one bit is reserved for the sign). Therefore, the dimensions of test functions and search agents are 75. Note that the initial values of basic parameters of the algorithms are provided in Table 4.

The experimental results are presented in Tables 5, 6 and 7. The results are collected over 10 independent runs. The average (*ave*) and standard deviation (*std*) of the best obtained solutions in the last iterations are reported as the results in bold type.

4.1 Unimodal benchmark functions

The unimodal test functions are useful to examine exploitation of the algorithms. As per the results of unimodal functions in Table 5, the BPSOGSA algorithm shows the best results on three of the unimodal benchmark functions in terms of the mean and standard deviation the results. According to the statistical results in Fig. 7, BPSOGSA and BPSO provide better results on 43 % of the unimodal benchmark functions, followed by GA with 14 %. BGSA did not show good performance on this set of functions due to the above-mentioned drawback (slow exploitation). Therefore, this is evidence that the proposed algorithm has high performance in finding the global solution of unimodal benchmark functions.

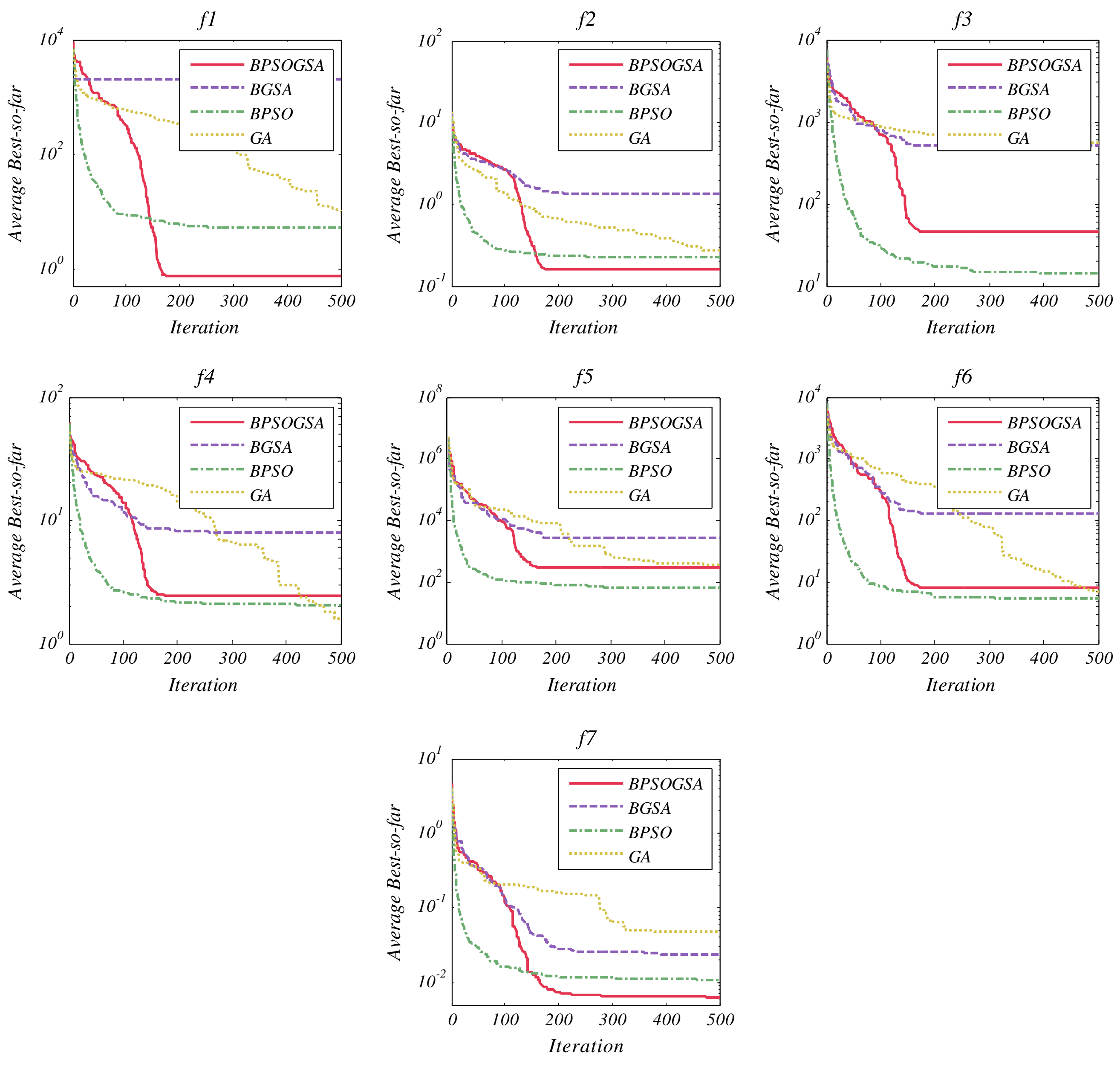

Fig. 8 Convergence curves for unimodal benchmark functions

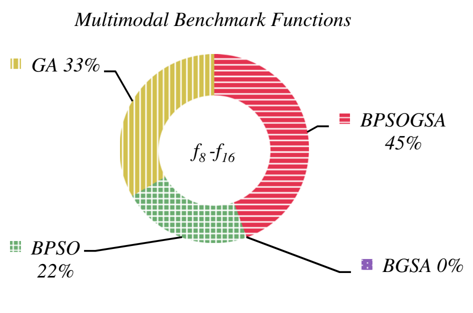

Fig. 9 Statistical results for unimodal multimodal benchmark functions

According to the convergence curves of algorithm when solving unimodal test functions in Fig. 8, the BPSOGSA and BPSO algorithms have the fastest convergence rate. These findings prove that the BPSOGSA algorithm seems to show the best exploitation ability and convergence rate.

4.2 Multimodal benchmark functions

The results of the multimodal benchmark functions are provided in Table 6 and Figs. 9, 10. Multimodal test

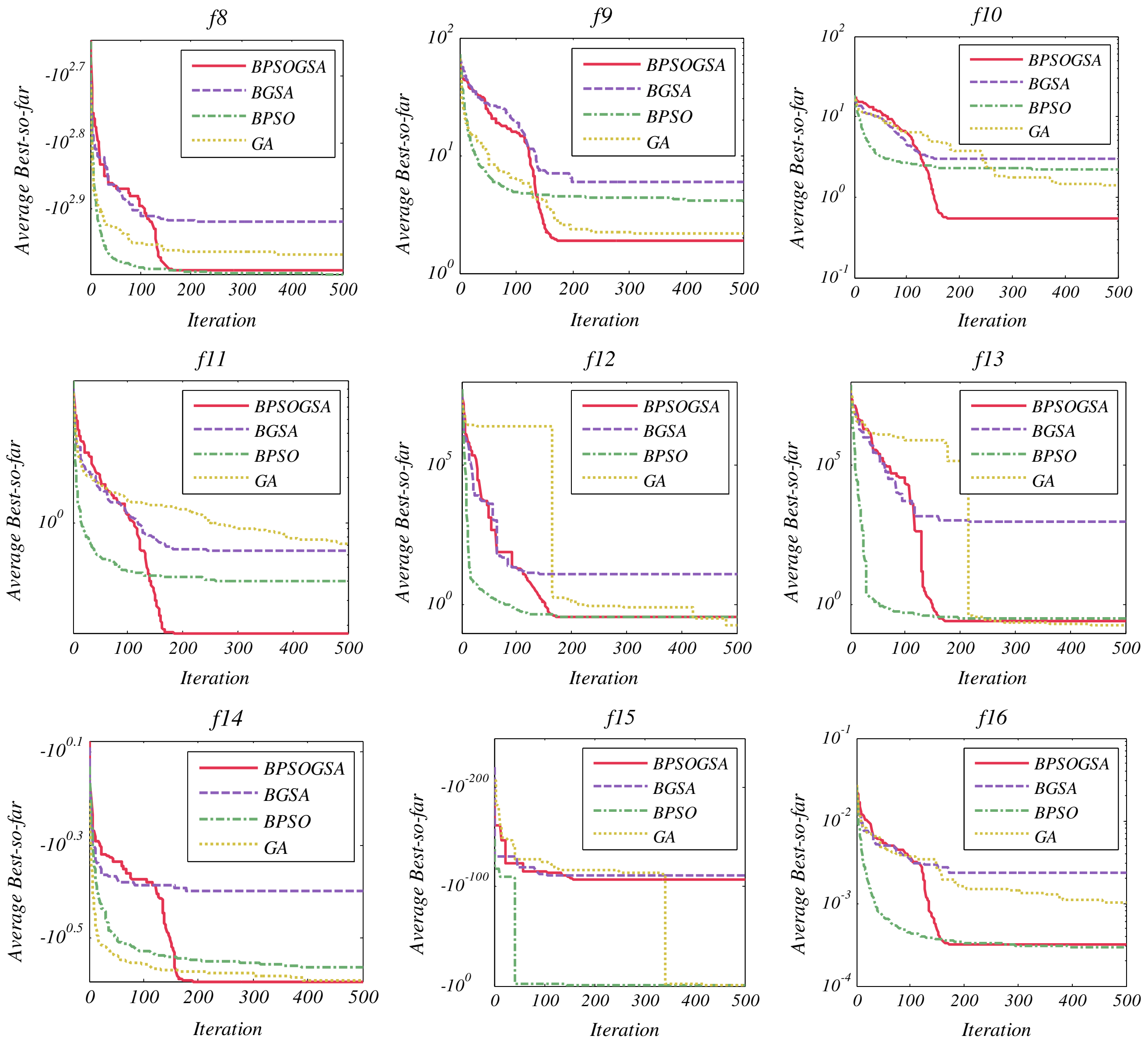

Fig. 10 Convergence curves for multimodal benchmark functions

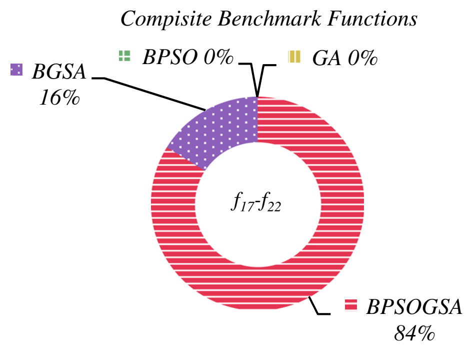

Fig. 11 Statistical results for composite benchmark functions

functions are helpful for benchmarking local optima avoidance of the algorithms. According to the results of Table 6 and Fig. 9, BPSOGSA shows the best results in four of the multimodal benchmark functions (45 % of the functions). However, the BPSO and GA algorithms outperform BPSOGSA in two (22 %) and three (33 %) multimodal test functions, respectively. Once again, the BGSA algorithm could not provide competitive results.

The convergence curves in Fig. 10 prove that BPSOGSA has the fastest convergence behavior in four out of nine functions. These findings prove that the BPSOGSA algorithm is able to avoid local optima, and this avoidance does

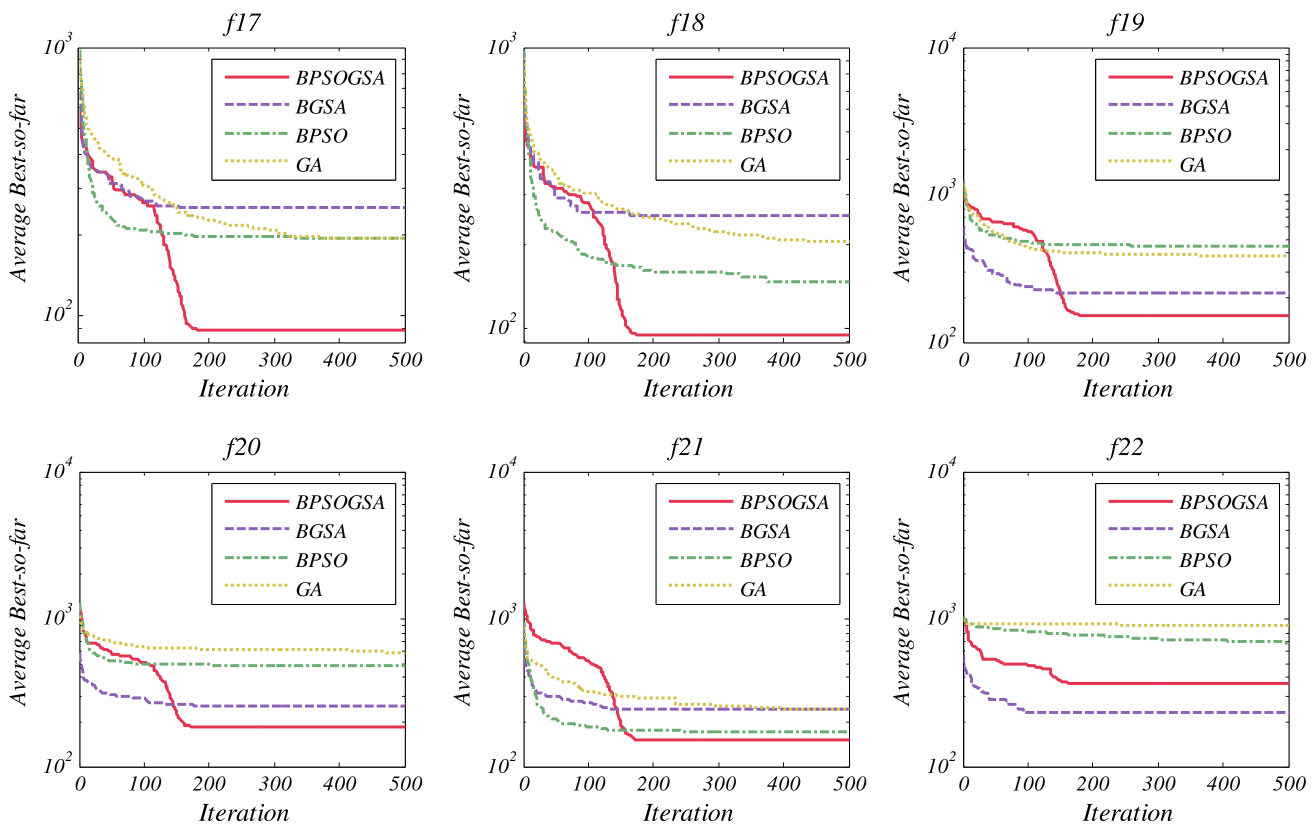

Fig. 12 Convergence curves for composite benchmark functions

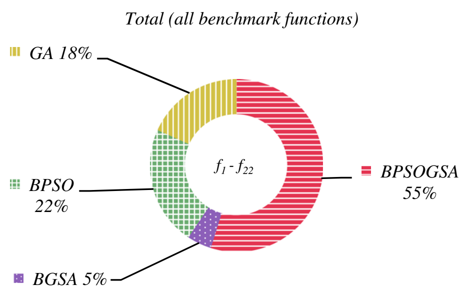

Fig. 13 Overall statistical results

not have negative impact of the convergence speed. In addition, the results can also evidence high exploration of BPSOGSA.

4.3 Composite benchmark functions

Composite functions are the most challenging test functions and suitable for benchmarking exploration and exploitation combined. The results in Table 7 and Figs. 11, 12 show that the BPSOGSA is evidently better than other algorithms on 84 % of the composite functions. For the function f_{22}, BGSA has the best results, followed by BPSOGSA.

The convergence of algorithms when solving composite functions are illustrated in Fig. 12. This figure shows that BPSOGSA has the fastest convergence rate. These findings prove that BPSOGSA properly and efficiently balances exploration and exploitation, which is due to the employed adaptive values for c_1' and c_2'.

Figure 13 shows the overall statistical results on all the benchmark functions whereby the BPSOGSA algorithm has the best result for 55 % of the benchmark functions.

To summarize, results prove BPSOGSA have better performance than BGSA, BPSO, and GA. Therefore, it can be said that the binary version of hybrid PSOGSA also has the strengths of both PSO and GSA in solving binary problems. According to this comprehensive comparative study and discussion, we state that this algorithm has merit among other binary algorithms.

The reason of the high performance of BPSOGSA on unimodal test functions is due to the fact the algorithm has higher exploitation compared to GSA. The social component of PSO allows BPSOGSA to exploit accurately around the best mass obtained so far. High exploration of BPSOGSA algorithm originates from the intrinsic characteristic of the GSA algorithm, wherein all search agents have impact on each other at each iteration (in contrast to the search agents of PSO that only sees *pbest* and *gbest*). This high exploration ability assists BPSOGSA to outperform other algorithms on multimodal test functions. The reason of significantly better results of BPSOGSA on composite test functions is due to adaptive values for $c_1^{'}$ and $c_2^{'}$. These two variables require BPSOGSA to balance between exploration and exploitation, so it emphasizes exploration in initial steps of iteration. However, exploitation is promoted as iteration increases. This cases local minima avoidance and accelerated convergence toward the global optimum over the course of iterations. Finally, the employed v-shaped transfer function obliges search agents of BPSOGSA to switch their positions when they are moving into unpromising areas of the search space, making it more likely to move toward promising areas of search space in contrast to s-shaped transfer functions.

5 Conclusion

In this paper, a binary version of hybrid PSOGSA called BPSOGSA was proposed by utilizing the same concepts of the continuous version in terms of search behavior. In order to justify the performance BPSOGSA, 22 benchmark functions were employed, and the results were compared with BGSA, BPSO, and GA. The results proved that BPSOGSA was able to provide competitive results and has merit among binary heuristic optimization algorithms in binary search spaces. According to the findings, the BPSOGSA algorithm successfully inherits the advantages of the PSOGSA. On one hand, BPSOGSA shows superior exploration since all search agents participate in updating position of a search agent. On the other hand, the exploitation of BPSOGSA is very accurate due to the social component of PSO integrated that causes accelerated convergence. The findings also proved that the adaptive values for $c_1^{'}$ and $c_2^{'}$ balance exploration and exploitation, so BPSOGSA is able to avoid local optima and converge to the promising regions of the search space. The transfer function employed also proved to be advantageous since there is no obligation for search agents to take 0 or 1.

For future studies, it is recommended to apply BPSOGSA in real optimization problems in order to further evaluate the efficiencies of BPSOGSA in solving real-world problems. Investigating the effect of different transfer functions on BPSOGSA would be interesting as well.

References

1. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
2. Kennedy J, Eberhart R (1995) Particle swarm optimization, vol 4, pp 1942–1948
3. Holland JH (1992) Genetic algorithms. Sci Am 267:66–72
4. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
5. Aarts EHL, Laarhoven PJM (1989) Simulated annealing: an introduction. Stat Neerl 43:31–52
6. Geem ZW, Kim JH (2001) A new heuristic optimization algorithm: harmony search. Simulation 76:60–68
7. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1:28–39
8. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
9. Simon D (2008) Biogeography-based optimization. IEEE Trans Evol Comput 12:702–713
10. Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. Inf Sci 269:188–209. doi:10.1016/j.ins.2014.01.038
11. Saremi S, Mirjalili S, Lewis A (2014) Biogeography-based optimisation with chaos. Neural Comput Appl 1–21. doi:10.1007/s00521-014-1597-x
12. Saremi S, Mirjalili S (2013) Integrating chaos to biogeography-based optimization algorithm. Int J Comput Commun Eng 2:655–658
13. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. doi:10.1016/j.advengsoft.2013.12.007
14. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17:4831–4845
15. Guo L, Wang G-G, Gandomi AH, Alavi AH, Duan H (2014) A new improved krill herd algorithm for global numerical optimization. Neurocomputing 138:392–402
16. Wang G-G, Gandomi AH, Alavi AH (2013) An effective krill herd algorithm with migration operator in biogeography-based optimization. Appl Math Model 38(9–10):2454–2462
17. Wang G-G, Gandomi AH, Alavi AH (2014) Stud krill herd algorithm. Neurocomputing 128:363–370
18. Wang G-G, Guo L, Gandomi AH, Hao G-S, Wang H (2014) Chaotic Krill Herd algorithm. Inf Sci 274:17–34
19. Wang G, Guo L, Wang H, Duan H, Liu L, Li J (2012) Incorporating mutation scheme into krill herd algorithm for global numerical optimization. Neural Comput Appl 1–19. doi:10.1007/s00521-012-1304-8
20. Saremi S, Mirjalili SM, Mirjalili S (2014) Chaotic Krill Herd optimization algorithm. Procedia Technol 12:180–185. doi:10.1016/j.protcy.2013.12.473
21. Esmin A, Lambert-Torres G, Alvarenga GB (2006) Hybrid evolutionary algorithm based on PSO and GA mutation. In: Sixth international conference on hybrid intelligent systems, pp 57–57
22. Holden N, Freitas AA (2008) A hybrid PSO/ACO algorithm for discovering classification rules in data mining. J Artif Evol Appl 2008:2
23. Holden NP, Freitas AA (2007) A hybrid PSO/ACO algorithm for classification. In: GECCO '07 proceedings of the 9th annual conference companion on genetic and evolutionary computation, pp 2745–2750
24. Lai X, Zhang M (2009) An efficient ensemble of GA and PSO for real function optimization. In: 2nd IEEE international conference on computer science and information technology, pp 651–655
25. Niu B, Li L (2008) A novel PSO-DE-based hybrid algorithm for global optimization. In: Advanced intelligent computing theories and applications. With aspects of artificial intelligence, pp 156–163
26. Zhang WJ, Xie XF (2003) DEPSO: hybrid particle swarm with differential evolution operator. In: IEEE international conference on systems, man and cybernetics, vol 4, pp 3816–3821
27. Wang G-G, Gandomi AH, Alavi AH, Hao G-S (2013) Hybrid krill herd algorithm with differential evolution for global

numerical optimization. Neural Comput Appl 1–12. doi:10.1007/s00521-013-1485-9
28. Wang G-G, Gandomi AH, Alavi AH (2013) A chaotic particle-swarm krill herd algorithm for global numerical optimization. Kybernetes 42(6):6962–6978
29. Mirjalili S, Hashim SZM (2010) A new hybrid PSOGSA algorithm for function optimization. In: 2010 international conference on computer and information application (ICCIA), pp 374–377. doi:10.1109/ICCIA.2010.6141614
30. Hatamlou A, Abdullah S, Othman Z (2011) Gravitational search algorithm with heuristic search for clustering problems. In: 3rd conference on data mining and optimization (DMO), pp 190–193
31. Shaw B, Mukherjee V, Ghoshal SP (2012) A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. Int J Electr Power Energy Syst 35:21–33
32. Zhang Y, Wu L, Zhang Y, Wang J (2012) Immune gravitation inspired optimization algorithm advanced intelligent computing, vol 6838. In: Huang D-S, Gan Y, Bevilacqua V, Figueroa J (eds) Advanced intelligent computing. Springer, Berlin, pp 178–185
33. Li C, Zhou J (2011) Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. Energy Convers Manag 52:374–381
34. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2010) BGSA: binary gravitational search algorithm. Nat Comput 9:727–745
35. Rashedi E, Nezamabadi S, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
36. Wang L, Xu Y, Mao Y, Fei M (2010) A discrete harmony search algorithm. Life Syst Model Intell Comput 37–43
37. Wang L, Fu X, Menhas M, Fei M (2010) A modified binary differential evolution algorithm. Life Syst Model Intell Comput 6329:49–57
38. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm, vol 5, pp 4104–4108
39. Mirjalili S, Mohd Hashim SZ, Moradian Sardroudi H (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137. doi:10.1016/j.amc.2012.04.069
40. Mirjalili S (2011) Hybrid particle swarm optimization and gravitational search algorithm for multilayer perceptron learning. Universiti Teknologi Malaysia, Faculty of Computer Science and Information System, Master thesis
41. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14. doi:10.1016/j.swevo.2012.09.002
42. Mirjalili S, Lewis A (2014) Adaptive gbest-guided gravitational search algorithm. Neural Comput Appl. doi:10.1007/s00521-014-1640-y
43. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3:82–102
44. Yang XS (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, London
45. Molga M, Smutnicki C (2005) Test functions for optimization needs. http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf
46. Digalakis J, Margaritis K (2001) On benchmarking functions for genetic algorithms. Int J Comput Math 77:481–506
47. Liang J, Suganthan P, Deb K (2005) Novel composition test functions for numerical global optimization, pp 68–75