Nat Comput (2010) 9:727–745
DOI 10.1007/s11047-009-9175-3

BGSA: binary gravitational search algorithm

Esmat Rashedi · Hossein Nezamabadi-pour · Saeid Saryazdi

Published online: 23 December 2009

Abstract Gravitational search algorithm is one of the new optimization algorithms that is based on the law of gravity and mass interactions. In this algorithm, the searcher agents are a collection of masses, and their interactions are based on the Newtonian laws of gravity and motion. In this article, a binary version of the algorithm is introduced. To evaluate the performances of the proposed algorithm, several experiments are performed. The experimental results confirm the efficiency of the BGSA in solving various nonlinear benchmark functions.

Keywords Gravitational search algorithm · Heuristic search algorithms · Law of gravity · Optimization

1 Introduction

Over the last decades, there has been a growing interest in algorithms inspired by the observation of natural phenomena. It has been shown by many researches that these algorithms are good replacements as tools to solve complex computational problems. Various heuristic approaches have been adopted by researches including genetic algorithm (Holland 1975), simulated annealing (Kirkpatrick et al. 1983), immune system (Farmer et al. 1986), ant system (Dorigo et al. 1996) and particle swarm optimization (Kennedy and Eberhart 1995; Kennedy and Eberhart 1997). Unfortunately, there is no algorithm to achieve the best solution for all optimization problems, and some algorithms give a better

E. Rashedi · H. Nezamabadi-pour (✉) · S. Saryazdi
Department of Electrical Engineering, Shahid Bahonar University of Kerman,
P.O. Box 76169-133, Kerman, Iran
e-mail: nezam@mail.uk.ac.ir

E. Rashedi
e-mail: rashedi_es@yahoo.com

S. Saryazdi
e-mail: saryazdi@mail.uk.ac.ir

solution for some problems than the others (Wolpert and Macready 1997; Engelbrecht et al. 2005; Cheng et al. 2007; Elbetagi et al. 2005; Youssef et al. 2001).

Gravitational search algorithm (GSA) is one of the latest heuristic optimization algorithms, which was first introduced by Rashedi et al. (2009) based on the metaphor of gravitational interaction between masses. GSA is inspired by the Newton theory that says: "Every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them". Gravity is a force, pulling together all matter.

The original version of GSA was designed for search spaces of real valued vectors. However, many optimization problems are set in binary discrete space. In this article, a version of the algorithm for binary encoding is introduced. In the previous version of GSA, the algorithmic "gravitational forces" lead directly to changes in the position of search points in a many-dimensional continuous space (the search space). In the binary version of GSA (BGSA), the outcome of these forces is converted into a probability value for each element of the binary vector, which guides whether that elements will take on the value 0 or 1.

This paper is organized as follows. Section 2 provides a brief review of GSA. In Sect. 3, we introduce basic aspects of binary version of GSA. An experimental study is given in Sect. 4, where the performance of the algorithm will be evaluated on nonlinear benchmark functions. Finally, a conclusion is given in Sect. 5.

2 The gravitational search algorithm

In the Newton gravitational law, each particle attracts every other particle with a "gravitational force" (Holliday et al. 1993; Schutz et al. 2003). The gravitational force between two bodies is directly proportional to the product of their masses and inversely proportional to the square of their distance (Schutz et al. 2003).

In this section, we introduce a brief review of GSA (Rashedi et al. 2009). In GSA, agents are considered as objects and their performance is measured by their masses. All these objects attract each other by a gravity force, and this force causes a movement of all objects globally towards the objects with heavier masses. The heavy masses correspond to good solutions of the problem.

In GSA, each mass (agent) has four specifications: its position, its inertial mass, its active gravitational mass, and its passive gravitational mass. The position of the mass corresponds to a solution of the problem, and its gravitational and inertial masses are determined using a fitness function.

In other words, each mass presents a solution, and the algorithm is navigated by properly adjusting the gravitational and inertia masses. By lapse of time, we expect that masses be attracted by the heaviest mass. This mass will present an optimum solution in the search space.

The GSA could be considered as an isolated system of masses. It is like a small artificial world of masses obeying the Newtonian laws of gravitation and motion. More precisely, masses obey the following laws (Rashedi et al. 2009):

Law of Gravity: each particle attracts every other particle and the gravitational force between two particles is directly proportional to the product of their masses and inversely proportional to the distance between them, R. We use here R instead of R^2,

because according to our experiment results, R provided better results than R^2 in all experimental cases.

Law of Motion: the current velocity of any mass is equal to the sum of the fraction of its previous velocity of mass and the variation in the velocity. Variation in the velocity or acceleration of any mass is equal to the force acted on the system divided by mass of inertia.

Now, consider a system with N agents (masses), the position of the ith agent is defined by:

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^n) \quad \text{for } i = 1, 2, \ldots, N \tag{1}$$

where x_i^d presents the position of ith agent in the dth dimension and n is the space dimension.

At a specific time "t", the force acting on mass "i" from mass "j" is defined as following:

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \tag{2}$$

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 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 total force that acts on agent i in a dimension d is a randomly weighted sum of dth component of the forces exerted from *Kbest* agents:

$$F_i^d(t) = \sum_{j \in kbest, j \neq i} rand_j F_{ij}^d(t) \tag{3}$$

where $rand_j$ is a random number in the interval [0,1] and *Kbest* is the set of first K agents with the best fitness value and biggest mass. *Kbest* is a function of time, initialized to K_0 at the beginning and decreasing with time. In such a way, at the beginning all agents apply force, and as the time passes, *Kbest* is decreased linearly and at the end, there will be just one agent apply force to others.

By the law of motion, the acceleration of the agent i at time t, and in direction d, $a_i^d(t)$, is given as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{4}$$

where M_{ii} is the inertial mass of ith agent. The next velocity of an agent is considered as a fraction of its current velocity added to its acceleration. Therefore, its velocity and its position could be calculated as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{5}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{6}$$

where $rand_i$ is an uniform random variable in the interval [0, 1]. This random number gives a randomized characteristic to the search. The gravitational constant, G, is initialized at the beginning and will be reduced with time to control the search accuracy. Hence, G is a function of initial value, G_0, and time, t:

$$G(t) = G(G_0, t) \tag{7}$$

Gravitational and inertia masses are simply calculated by the fitness evaluation. A heavier mass means a more efficient agent. This means that better agents have higher attractions and walk more slowly. Assuming the equality of the gravitational and inertia mass, the values of masses are calculated using the map of fitness. The gravitational and inertial masses are updated by the following equations (Rashedi et al. 2009):

$$M_{ai} = M_{pi} = M_{ii} = M_i, \ i = 1, 2, \ldots, N \tag{8}$$

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{9}$$

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{N} q_j(t)} \tag{10}$$

where $fit_i(t)$ represent the fitness value of the agent i at t, and, $worst(t)$ and $best(t)$ are defined as follows (for a minimization problem):

$$best(t) = \min_{j \in \{1,..,N\}} fit_j(t) \tag{11}$$

$$worst(t) = \max_{j \in \{1,..,N\}} fit_j(t) \tag{12}$$

The principle of GSA is shown in Fig. 1.

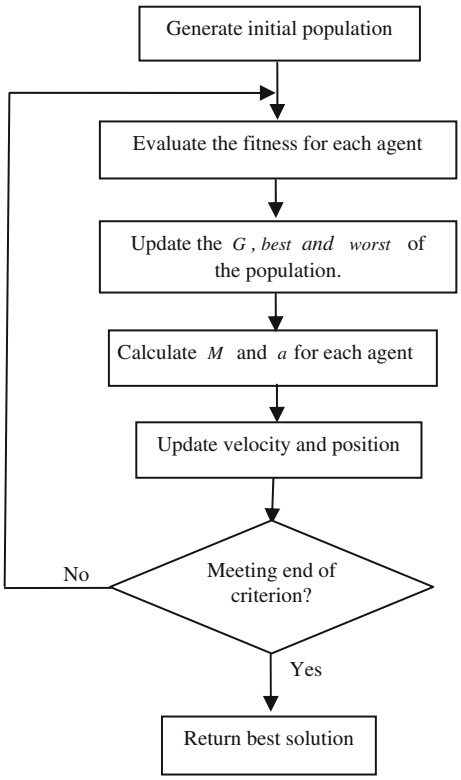

Fig. 1 General principle of GSA (Rashedi et al. 2009)

To see how the GSA is efficient, some remarks are noted (Rashedi et al. 2009):

- Since each agent could observe the performance of the others, the gravitational force is an information-transferring tool.
- Due to the force that acts on an agent from its neighborhood agents, it can see space around itself.
- A Heavy mass has a large effective attraction radius and hence a great intensity of attraction. Therefore, agents with a higher performance have a greater gravitational mass. As a result, agents tend to move toward the best agent.
- The inertia mass is against the motion and make the mass movement slow. Hence, agents with heavy inertia mass move slowly and hence search the space more locally. So, it can be considered as an adaptive learning.
- Gravitational constant adjusts the accuracy of the search, so it decreases with time similar to the temperature in a simulated annealing algorithm.
- GSA is memory-less. However, it works efficiently like the algorithms with memory.

Theoretically, GSA belongs to the class of swarm based heuristic algorithms. Rashedi et al. (2009) gave a comparative study between GSA and a small number of well-known swarm algorithms like PSO. The results suggest that this approach which is inspired by the law of gravity has merit in the field of optimization.

3 Binary GSA

There are many optimization problems such as feature selection and dimensionality reduction (Avishek and Maiti 2010; Beretaa and Burczynski 2007; Wang et al. 2007; Chuang et al. 2008; Zeng et al. 2009), data mining (Srinivasa et al. 2007), unit commitment (Yuan et al. 2009), and cell formation (Wu et al. 2008), in which it is natural to encode solutions as binary vectors. In addition, problems defined in the real space, may be considered in the binary space, too. The solution is to display real digits with some bits in the binary mode. The binary search space is considered as a hypercube in which an agent may move to nearer and farther corners of the hypercube by flipping various numbers of the bits.

In this section, we present a binary version of GSA. To do this, some basic concepts of GSA will be modified. In discrete binary environment, every dimension can take only 0 or 1. Moving through a dimension means that the corresponding variable value changes from 0 to 1 or vice versa. In order to introduce a binary mode for the gravitational algorithm, the updating procedure of the force, acceleration and velocity may be considered similar to the continuous algorithm (Eqs. 4–6). The main difference between continuous and binary GSA is that in the binary algorithm, the position updating means a switching between "0" and "1" values. This switching should be done according to the mass velocity. Our idea is to update the position in a manner that the current bit value is changed with a probability that is calculated according to the mass velocity. In other words, BGSA updates the velocity based on Eq. 5 and considers the new position to be either 1 or 0 with the given probability.

Before defining a transfer function to map the velocity to the probability of position updating, it will be useful to remind some basic concepts of GSA.

- A large absolute value of the velocity shows that the current position of the mass is not proper and a great movement is required to reach the optimum position.
- A small absolute value of the velocity indicates that the current position of the mass is close to the optimum position and there is a small distance reaching to the optimum position. Then, after finding the optimal solution, the velocity becomes zero.

Based on the above concepts of the GSA, in the implementation of the BGSA the following concepts should be taken into account:

- A large absolute value of velocity must provide a high probability of changing the position of the mass respect to its pervious position (from 1 to 0 or vice versa).
- A small absolute value of the velocity must provide a small probability of changing the position. In other words, a zero value of the velocity represents that the mass position is good and must not be changed.

Based on the above-mentioned concepts, a proper probability function should be defined such that for a small $|v_i^d|$, the probability of changing x_i^d must be near zero and for a large $|v_i^d|$, the probability of x_i^d movement must be high. We define function $S(v_i^d)$ to transfer v_i^d into a probability function. $S(v_i^d)$ should be bounded within interval [0,1] and increases with increasing $|v_i^d|$. We define $S(v_i^d)$ according to Eq. 13, which is illustrated in Fig. 2. As this figure shows, the proposed function satisfies all the above-mentioned requirements.

$$S(v_i^d(t)) = |\tanh(v_i^d(t))| \tag{13}$$

Once $S(v_i^d)$ is calculated, the agents will move according to the rule explained in Eq. 14.

$$\begin{aligned} &\text{if } rand < S(v_i^d(t+1)) \text{ then } x_i^d(t+1) = \text{complement}(x_i^d(t)) \\ &\text{else } x_i^d(t+1) = x_i^d(t) \end{aligned} \tag{14}$$

To achieve a good converge rate, we limit the velocity, $|v_i^d| < v_{\max}$. Based on our experiments, $v_{\max}$ is set to be 6. It is to be noted here that the distance, R, is computed based on the Hamming distance.

4 Experimental results

To evaluate the proposed algorithm, a set of 23 minimization and 2 maximization benchmark functions are tested in this section and the results are reported along with Genetic Algorithm (GA) and Binary Particle Swarm Optimization (BPSO).

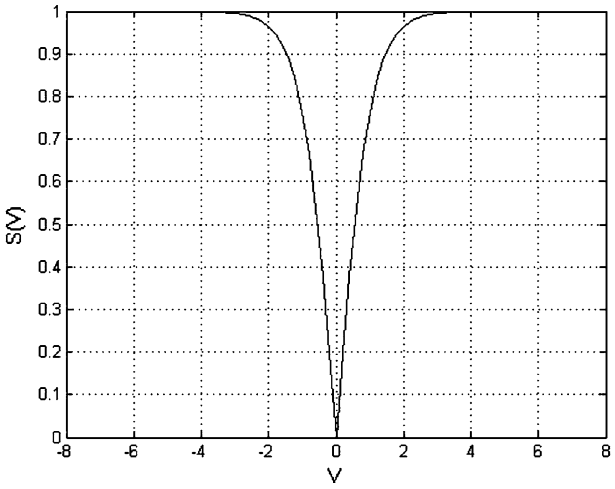

Fig. 2 The proposed function

4.1 Benchmark functions

The benchmark functions are taken from (Yao et al. 1999; Digalakis and Margaritis 2002). These functions are given in Tables 1, 2, 3, and 4, where m is the dimension of the function, f_{opt} is the optimum value of the function and $S \subseteqq R^m$. The first seven functions (F_1 to F_7) are unimodal. For unimodal functions, the convergence rates of the algorithm are more interesting than the final results of optimization. F_8 to F_{13} are multimodal having many local minima and the algorithm must be capable in finding the optimum solution (or a good near-global optimum) and it should not be trapped in local optima. F_{14} to F_{23} are multimodal functions not having many local minima. Functions in Table 4 have discrete nature.

Functions in Tables 1, 2, and 3 are minimization problems. The minimum value, f_{opt}, of the functions of Tables 1 and 2 are zero, except for F_8 which has a minimum value of $-418.9829 \times m$. The optimum location, X_{opt}, for functions of Tables 1 and 2, are in $[0]^m$, except for F_5, F_{12} and F_{13} with X_{opt} in $[1]^m$ and F_8 in$[420.96]^m$. A detailed description of functions of Table 3 is given in Appendix.

Functions in Table 4 are Maximization binary problems. These functions are described only in binary space 0–1 values. The maximum values of these functions are depending with m.

4.2 Comparative study

To see how well the proposed algorithm perform, we applied BGSA to the above benchmark functions and compared the results with those of GA as well as BPSO. In GA, one-point crossover, uniform mutation and roulette wheel selection were used. The crossover probability and mutation probability were set to 0.9 and 0.005, respectively (Goldberg 1989).

Table 1 Unimodal test functions

Test function	S
$F_1(X) = \sum_{i=1}^{m} x_i^2$	$[-100, 100]^m$
$F_2(X) = \sum_{i=1}^{m} \lvert x_i \rvert + \prod_{i=1}^{m} \lvert x_i \rvert$	$[-10, 10]^m$
$F_3(X) = \sum_{i=1}^{m} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^m$
$F_4(X) = \max_i \{\lvert x_i \rvert, 1 \le i \le m\}$	$[-100, 100]^m$
$F_5(X) = \sum_{i=1}^{m-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	$[-30, 30]^m$
$F_6(X) = \sum_{i=1}^{m} ([x_i + 0.5])^2$	$[-100, 100]^m$
$F_7(X) = \sum_{i=1}^{m} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]^m$

Table 2 Multimodal test functions

Test function	S
$F_8(X) = \sum_{i=1}^{m} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^m$
$F_9(X) = \sum_{i=1}^{m} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^m$
$F_{10}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{m} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{m}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^m$
$F_{11}(X) = \frac{1}{4000}\sum_{i=1}^{m} x_i^2 - \prod_{i=1}^{m}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^m$
$F_{12}(x) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{m-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{m} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$	$[-50, 50]^m$
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	
$F_{13}(X) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{m}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_m - 1)^2[1 + \sin^2(2\pi x_m)]\} + \sum_{i=1}^{m} u(x_i, 5, 100, 4)$	$[-50, 50]^m$

Table 3 Multimodal test functions with fix dimension

Test Function	S
$F_{14}(X) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	$[-65.53, 65.53]^2$
$F_{15}(X) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]^4$
$F_{16}(X) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^2$
$F_{17}(X) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$	$[-5, 10] \times [0, 15]$
$F_{18}(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-2, 2]^2$
$F_{19}(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^3$
$F_{20}(X) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^6$
$F_{21}(X) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$
$F_{22}(X) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$
$F_{23}(X) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$

Table 4 Maximization functions in binary search space

Test Function	S
$\text{Max-Ones} = \sum_{i=1}^{m} x_i$	$[\{0,1\}]^m$
$\text{Royal-Road} = \sum_{i=1}^{\frac{m}{8}} \left(\prod_{j=8(i-1)+1}^{8i} x_j \right)$	$[\{0,1\}]^m$

In BPSO algorithm, v_i^d and x_i^d are calculated as follows (Kennedy and Eberhart 1997):

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_1 \left(pbest_i^d - x_i^d(t)\right) + c_2 r_2 \left(gbest^d - x_i^d(t)\right) \tag{15}$$

$$\begin{aligned} &if\ rand < \left(\frac{1}{1+e^{-v_{id}(t+1)}}\right) \quad then\ x_i^d(t+1) = 1 \\ &\qquad\qquad\qquad\qquad\qquad\quad else\ x_i^d(t+1) = 0 \end{aligned} \tag{16}$$

where r_1 and r_2 and *rand* are three uniform random variables in the range [0, 1], c_1 are c_2 are positive constants, w is the inertia weight. $X_i = (x_i^1, x_i^2, \ldots, x_i^n)$ and $V_i = (v_i^1, v_i^2, \ldots, v_i^n)$ represent position and velocity of the ith particle, respectively. $pbest_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^n)$ and $gbest = (gbest^1, gbest^2, \ldots, gbest^n)$ represent the best pervious position of the ith particle and the best pervious position among all the particles in the population, respectively. In addition, $c_1 = c_2 = 2$ is taken and the inertia factor, w, is decreased linearly from 0.9 to 0.2.

In binary version of GSA, G is considered as a linear decreasing function as follows.

$$G(t) = G_0(1 - \frac{t}{T}) \tag{17}$$

where G_0 is a constant and set to 100 and T is the total number of iterations (the total age of system). Furthermore, K_0 (in Eq. 4) is set to N(total number of agents) and is decreased linearly to 1.

4.3 Results and discussion

We applied the three mentioned algorithms to the benchmark functions. In all cases, population size is set to 50 ($N = 50$). Dimension is 5 ($m = 5$) for functions of Tables 1 and 2 and maximum iteration is 500 for functions of Tables 1, 2 and 3. To represent each continuous variable, 15 bits have been used. Therefore, for each continuous function the dimension of agents is $n = m \times 15$. The results are averaged over 30 independent runs under 30 different random seeds and the average best-so-far solution, standard deviation of best solution and median of the best solution in the last iteration are reported.

4.3.1 Unimodal functions

The results for unimodal functions are reported in Table 5. As this table illustrates BGSA provides better results than BPSO and slightly better than GA for all functions. In addition, the good convergence rate of BGSA could be concluded from Fig. 3. According to Table 5 and Fig. 3, BGSA tends to find the global optimum in unimodal functions and has a good convergence rate.

Table 5 Comparison between BGSA, GA and BPSO on unimodal functions of Table 1 with m = 5, where "ABSF", "MBSF" and "STDV" indicate Average best so far solution, Median best so far solution, and Standard Deviation of the best so far solution in the last iteration, respectively

Function		GA	BPSO	BGSA
F_1	ABSF	4.65×10^{-5}	64.62	4.65×10^{-5}
	STDV	0	73.16	0
	MBSF	4.65×10^{-5}	68.83	4.65×10^{-5}
F_2	ABSF	0.0015	1.0823	0.0016
	STDV	5.22×10^{-8}	0.1087	1.49×10^{-7}
	MBSF	0.0015	1.0404	0.0015
F_3	ABSF	649.73	56.16	26.29
	STDV	$3.65 \times 10^{+5}$	$1.02 \times 10^{+3}$	$4.64 \times 10^{+3}$
	MBSF	408.53	51.37	0.6855
F_4	ABSF	0.9186	4.39	1.28
	STDV	0.7583	2.07	15.56
	MBSF	0.705	4.2787	0.0214
F_5	ABSF	406.26	647.24	3.8456
	STDV	$4.68 \times 10^{+5}$	$2.87 \times 10^{+4}$	2.0474
	MBSF	13.39	465.41	3.9854
F_6	ABSF	0.6505	45.11	0.4584
	STDV	0.0375	166.39	0.0386
	MBSF	0.6712	47.73	0.5062
F_7	ABSF	0.0032	0.0416	0.0025
	STDV	7.54×10^{-6}	0.0011	2.04×10^{-6}
	MBSF	0.0025	0.0412	0.0024

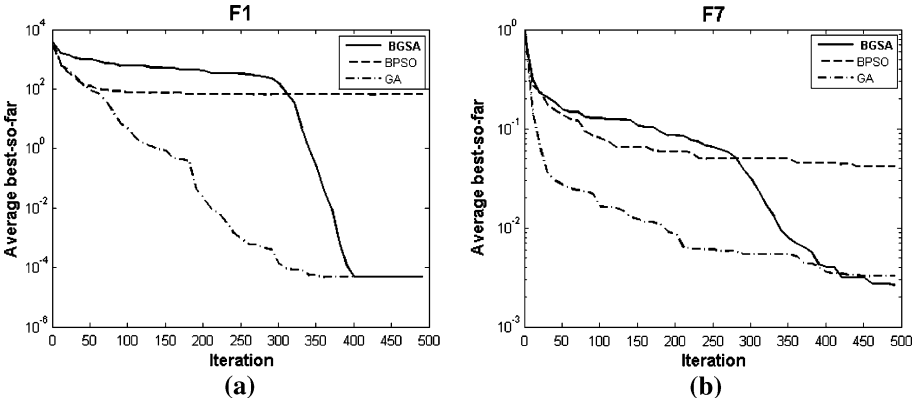

Fig. 3 Comparison between BGSA, BPSO and GA on unimodal functions **a** F_1, **b** F_7 with $m = 5$

4.3.2 Multimodal functions

We have two sets of multimodal functions. The functions of Table 2 (F_8–F_{13}) contain many local optima and their local minima increase exponentially as the dimension of the function increases. The dimension of functions of Table 3 (F_{14}–F_{23}) is fixed and they have a few number of local minima.

Table 6 Comparison between BGSA, GA and BPSO on multimodal functions of Table 2 with m = 5, where "ABSF", "MBSF" and "STDV" indicate Average best so far solution, Median best so far solution, and Standard Deviation of the best so far solution in the last iteration, respectively

Function		GA	BPSO	BGSA
F_8	ABSF	−1034.7	−2009.1	−2083.1
	STDV	$3.16 \times 10^{+4}$	930.3	596.97
	MBSF	−1012	−2021	−2091
F_9	ABSF	5.96	9.3	4.96
	STDV	11.78	4.75	4.58
	MBSF	6.07	9.98	5.73
F_{10}	ABSF	8.82	5	0.004
	STDV	30	1.25	0
	MBSF	10.11	4.75	0.004
F_{11}	ABSF	0.1716	1.28	0.0409
	STDV	0.0078	0.0922	0.0004
	MBSF	0.1511	1.37	0.034
F_{12}	ABSF	2.42	3.53	0.9001
	STDV	4.67	5.15	0.9834
	MBSF	2.1	3.49	0.4725
F_{13}	ABSF	0.1835	0.9359	0.7734
	STDV	0.0454	1.62	3.39
	MBSF	0.0689	0.5694	0.2037

We have carried out experiments on F_8–F_{13}. The dimension of these functions is set to 5. The results are averaged over 30 runs and the average best so far solution, standard deviation of best so far solution and median of the best so far solution in the last iteration are reported for these functions in Table 6. For functions F_8–F_{12} BGSA reaches a much better solution than the other algorithms while for function F_{13}, GA presents a better solution than BGSA. Table 7 shows the comparison between BGSA, GA and BPSO on multimodal benchmark functions of Table 3 (F_{14}–F_{23}). The results show that BPSO and BGSA in most functions have similar solutions and their performances are almost the same; slightly better for BPSO in some cases. The progress of the average best-so-far solution over 30 independent runs for functions F_{10}, F_{12}, F_{15} and F_{19} are shown in Fig. 4.

4.3.3 Discrete functions

The functions of Table 4 (Max-Ones and Royal-Road) are binary in nature. These functions should be maximized. For the maximization problem, the concepts of the *best* and the *worst* are changed (Eqs. 11 and 12). This means that the *best* is computed according to Eq. 12 and vice versa. To evaluate the ability of the algorithms, the functions of Table 4 are considered with several value of m, $m = 32$, 64, 80 and 160 where m indicates the dimension of the function. The maximum iteration for $m = 32$, 64 and 80 is set to be 1000 while for $m = 160$, it is considered to be 2000. The dimension of agents, n, for binary functions is equal to m.

The results are given in Table 8. As this table suggests BGSA provides the optimum solution for all cases. The largest difference in performance between BGSA, GA and BPSO occurs in binary functions especially when the dimension of the functions increases. In addition, the good convergence rate of BGSA could be concluded from Fig. 5.

Table 7 Comparison between BGSA, GA and BPSO on multimodal functions of Table 3, where "ABSF", "MBSF" and "STDV" indicate Average best so far solution, Median best so far solution, and Standard Deviation of the best so far solution in the last iteration, respectively

Function		GA	BPSO	BGSA
F_{14} $m = 2$	ABSF	4.3	0.9983	1.002
	STDV	24.83	6.37×10^{-7}	8.88×10^{-5}
	MBSF	2.39	0.998	0.9987
F_{15} $m = 4$	ABSF	0.0129	0.0009	0.0021
	STDV	8.29×10^{-5}	2.65×10^{-8}	2.99×10^{-6}
	MBSF	0.0135	0.0009	0.0011
F_{16} $m = 2$	ABSF	−0.8933	−1.0281	−1.0285
	STDV	0.017	9.95×10^{-5}	4.90×10^{-5}
	MBSF	−0.9441	−1.0316	−1.0313
F_{17} $m = 2$	ABSF	0.4392	0.3982	0.3979
	STDV	0.0112	5.23×10^{-7}	2.99×10^{-10}
	MBSF	0.399	0.3979	0.3979
F_{18} $m = 2$	ABSF	8.3	3	3.03
	STDV	125.01	2.95×10^{-5}	0.002
	MBSF	3.0008	3.002	3
F_{19} $m = 3$	ABSF	−3.4059	−3.8597	−3.8626
	STDV	0.0576	2.98×10^{-6}	2.94×10^{-7}
	MBSF	−3.4654	−3.8598	−3.8628
F_{20} $m = 6$	ABSF	−1.7508	−3.1106	−3.3095
	STDV	0.2684	0.0023	0.0014
	MBSF	−1.7393	−3.0941	−3.3219
F_{21} $m = 4$	ABSF	−1.0837	−6.3529	−3.5636
	STDV	0.8140	7.70	1.06
	MBSF	−0.7757	−4.9855	−3.0951
F_{22} $m = 4$	ABSF	−0.9690	−8.0993	−5.1643
	STDV	0.12	5.63	8.81
	MBSF	−0.9081	−9.0716	−3.9588
F_{23} $m = 4$	ABSF	−1.4397	−5.7469	−3.5731
	STDV	0.79	2.60	1.29
	MBSF	−1.1190	−5.0309	−3.3737

According to this figure, BGSA tends to find the global optimum faster than the other tested algorithms.

4.4 BGSA with different parameters

The BGSA has some parameter to be tuned such as G_0 and *kbest*. To examine the effect of different values of these parameters on the performance of BGSA in detail, a set of experiments have been carried out on BGSA using different values of G_0 and *kbest*. A set of seven benchmark functions from Tables 1, 2, 3, and 4 were selected in these experiments. The setup of these experiments such as population size, maximum iteration, etc. is the same as before. In the first group of experiments, we examine the impact of G_0. To do this, the values $G_0 = 50$ and $G_0 = 150$ are examined for seven benchmark functions. It is

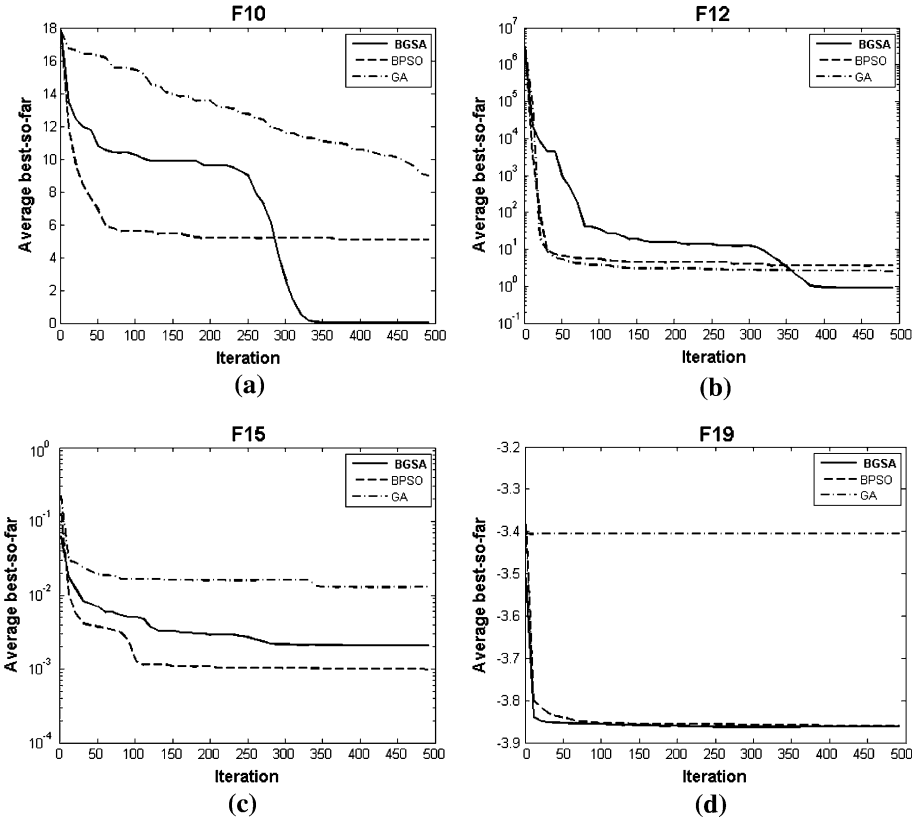

Fig. 4 Comparison between BGSA, BPSO and GA on unimodal functions **a** F_{10} with $m = 5$, **b** F_{12} with $m = 5$, **c** F_{15} with $m = 4$ and **d** F_{19} with $m = 3$

noted that in previous experiments we used the value of 100 for G_0. For simplicity, the results related to $G_0 = 100$ are also reported along with those of new values in Table 9. The results are averaged over 30 independent runs and the average best so far solution, and standard deviation of best so far solutions are reported for these functions. The results show that $G_0 = 100$ is not the optimal value for all functions. This means that G_0 is a problem dependent parameter.

In addition, to see why *kbest* must be varied with time, we performed some experiments with different values of *kbest* in which *kbest* is considered as a constant. The results are given in Table 10. As the table shows, using the linear time varying function for *kbest* produces better results than using constant values.

In fact, different values of parameters lead the algorithm to different exploration and exploitation capabilities. In other words, by setting these parameters, one can control the convergence rate, exploration, and exploitation capabilities and escape trapping local optima. On the other hand, each problem needs to have the specific capabilities of exploration and exploitation. Hence, it is acceptable to have some parameters to be problem dependent.

Table 8 Comparison between BGSA, GA and BPSO on Max-Ones and Royal-Road function with $m = 32$, 64, 80 and 160, where "ABSF", "MBSF" and "STDV" indicate Average best so far solution, Median best so far solution, and Standard Deviation of the best so far solution in the last iteration, respectively

Function		GA	BPSO	BGSA
Max-Ones $m = 32$	ABSF	28	32	32
	STDV	2.4	0	0
	MBSF	27.5	32	32
Max-Ones $m = 64$	ABSF	48.3	59.7	64
	STDV	1.78	0.9	0
	MBSF	48	60	64
Max-Ones $m = 80$	ABSF	58.1	72.1	80
	STDV	11.2	0.98	0
	MBSF	57	72	80
Max-Ones $m = 160$	ABSF	108	131.7	160
	STDV	8.44	10.01	0
	MBSF	107	131	160
Royal-Road $m = 32$	ABSF	2.2	2.7	4
	STDV	1.73	0.23	0
	MBSF	2.5	3	4
Royal-Road $m = 64$	ABSF	4.6	3.5	8
	STDV	0.48	0.5	0
	MBSF	5	3	8
Royal-Road $m = 80$	ABSF	5.5	3.8	10
	STDV	0.72	0.17	0
	MBSF	5.5	4	10
Royal-Road $m = 160$	ABSF	7.5	4.5	20
	STDV	0.94	2.77	0
	MBSF	7	4.5	20

5 Conclusion

In recent years, various heuristic optimization algorithms have been developed. GSA is a new heuristic search algorithm where it is constructed based on the law of Gravity and the notion of mass interactions. The GSA algorithm uses the theory of Newtonian physics and its searcher agents are the collection of masses. In this article, a binary version of GSA has been introduced.

In order to evaluate our algorithm, we have examined it on a set of various standard benchmark functions. In this paper, BGSA has been tested and compared with a small number of well-known alternative approaches. The results suggest that this interesting approach inspired by the law of gravity has merit in the field of optimization in binary search spaces, but much work has to be done to establish how well BGSA performs against state of the art techniques.

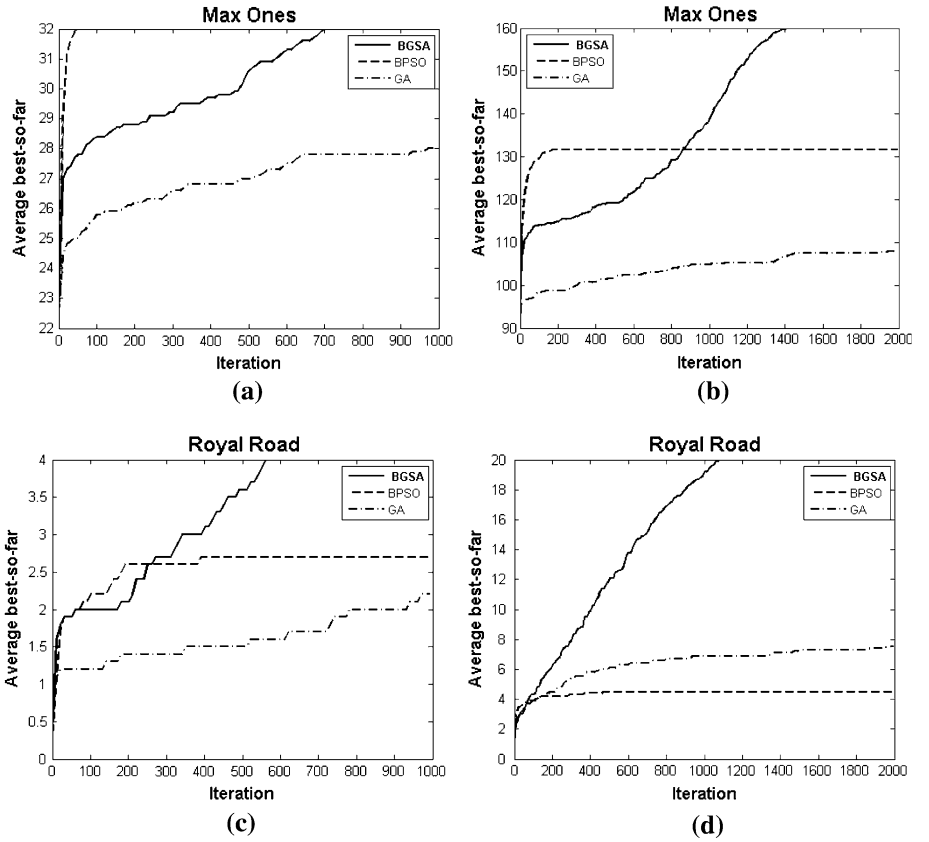

Fig. 5 Comparison between BGSA, BPSO and GA on **a** Max-Ones with $m = 32$, **b** Max-Ones with $m = 160$, **c** Royal-Road with $m = 32$, and **d** Royal-Road with $m = 160$

Table 9 The average best so far solution and standard deviation found by BGSA using different G_0 for seven benchmark functions

Function	$G_0 = 50$	$G_0 = 150$	$G_0 = 100$
F_1	0.245 ± 0.601	$4.65 \times 10^{-5} \pm 0$	$4.65 \times 10^{-5} \pm 0$
F_7	$0.0035 \pm 2.98 \times 10^{-6}$	$0.0035 \pm 2.18 \times 10^{-6}$	$0.0025 \pm 2.04 \times 10^{-6}$
F_9	3.39 ± 2.27	3.68 ± 2.61	4.96 ± 4.58
F_{13}	$722.21 \pm 3.69 \times 10^{+6}$	0.122 ± 0.041	0.7734 ± 3.39
F_{16}	$-1.0280 \pm 1.20 \times 10^{-5}$	$-1.025 \pm 5.16 \times 10^{-5}$	$-1.0285 \pm 4.90 \times 10^{-5}$
F_{20}	$-3.29 \pm 2.50 \times 10^{-3}$	$-3.32 \pm 7.2 \times 10^{-10}$	-3.3095 ± 0.0014
Royal-Road $m = 160$	19.70 ± 0.4557	19.60 ± 0.2667	20 ± 0

Table 10 The average best so far solution and standard deviation found by BGSA using different *kbest* for seven benchmark functions

Function	*kbest* = 5	*kbest* = 25	*kbest* = 50	Time varying *kbest*
F_1	24.5 ± 2633.6	$4.65 \times 10^{-5} \pm 0$	$250.43 \pm 9.55 \times 10^{-3}$	$4.65 \times 10^{-5} \pm 0$
F_7	$0.0033 \pm 2.8 \times 10^{-5}$	$0.0045 \pm 2.65 \times 10^{-6}$	0.05161 ± 0.0004	$0.0025 \pm 2.04 \times 10^{-6}$
F_9	4.15 ± 3.22	4.05 ± 0.7937	12.49 ± 11.17	4.96 ± 4.58
F_{13}	24.19 ± 6130.7	0.3356 ± 0.0511	1710.2 ± 1.39	0.7734 ± 3.39
F_{16}	-1.0116 ± 0.0008	$-1.0292 \pm 1.56 \times 10^{-5}$	$-1.0292 \pm 4.63 \times 10^{-6}$	$-1.0285 \pm 4.90 \times 10^{-5}$
F_{20}	-3.29 ± 0.0014	$-3.32 \pm 8.70 \times 10^{-10}$	-3.26 ± 0.000166	-3.3095 ± 0.0014
Royal-Road $m = 160$	9.2 ± 5.066	19.8 ± 0.1778	20 ± 0	20 ± 0

Acknowledgements The authors would like to thank the guest Editors and the anonymous reviewers for their very helpful suggestions. In addition, the authors would like to extend their appreciation to Dr. Saeid Seydnejad for proof reading the manuscript and providing valuable comments.

Appendix

See Tables 11, 12, 13, 14, 15, and 16.

Table 11 a_{ij} in F_{14}

$$(a_{ij}) = \begin{pmatrix} -32, -16, 0, 16, 32, -32, \ldots, 0, 16, 32 \\ -32, -32, -32, -32, -32, -16, \ldots, 32, 32, 32 \end{pmatrix}$$

Table 12 a_i and b_i in F_{15}

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627	0.0456	0.0342	0.0323	0.0235	0.0246
b_i^{-1}	0.25	0.5	1	2	4	6	8	10	12	14	16

Table 13 a_{ij}, c_i and P_{ij} in F_{19}

i	a_{ij}, j = 1, 2, 3			c_i	P_{ij}, j = 1, 2, 3		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	30	3.2	0.03815	0.5743	0.8828

Table 14 a_{ij}, c_i and P_{ij} in F_{20}

i	a_{ij}, j = 1, 2, 3, 4, 5, 6						c_i	P_{ij}, j = 1, 2, 3, 4, 5, 6					
1	10	3	17	3.5	1.7	8	1	0.131	0.169	0.556	0.012	0.828	0.588
2	0.05	10	17	0.1	8	14	1.2	0.232	0.413	0.830	0.373	0.100	0.999
3	3	3.5	1.7	10	17	8	3	0.234	0.141	0.352	0.288	0.304	0.665
4	17	8	0.05	10	0.1	14	3.2	0.404	0.882	0.873	0.574	0.109	0.038

Table 15 a_{ij} and c_i in F_{21}, F_{22} and F_{23}

i	a_{ij}, j = 1, 2, 3, 4				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Table 16 Optima in functions of Table 3

F	X_{opt}	f_{opt}
F_{14}	(−32, 32)	1
F_{15}	(0.1928, 0.1908, 0.1231, 0.1358)	0.00030
F_{16}	(0.089, −0.712), (−0.089, 0.712)	−1.0316
F_{17}	(−3.14, 12.27), (3.14, 2.275), (9.42, 2.42)	0.398
F_{18}	(0, −1)	3
F_{19}	(0.114, 0.556, 0.852)	−3.86
F_{20}	(0.201, 0.15, 0.477, 0.275, 0.311, 0.657)	−3.32
F_{21}	5 local minima in a_i $i = 1, \ldots, 5$	−10.1532
F_{22}	7 local minima in a_i $i = 1, \ldots, 7$	−10.4028
F_{23}	10 local minima in a_i $i = 1, \ldots, 10$	−10.5363

References

Avishek P, Maiti J (2010) Development of a hybrid methodology for dimensionality reduction in Mahalanobis–Taguchi system using Mahalanobis distance and binary particle swarm optimization. Expert Syst Appl 37(2):1286–1293

Beretaa M, Burczynski T (2007) Comparing binary and real-valued coding in hybrid immune algorithm for feature selection and classification of ECG signals. Eng Appl Artif Intell 20:571–585

Cheng YM, Li L et al (2007) Performance studies on six heuristic global optimization methods in the location of critical slip surface. Comput Geotech 34(6):462–484

Chuang LH, Chang HW et al (2008) Improved binary PSO for feature selection using gene expression data. Comput Biol Chem 32(1):29–38

Digalakis JG, Margaritis KG (2002) An experimental study of benchmarking functions for genetic algorithms. Int J Comput Math 79(4):403–416

Dorigo M, Maniezzo V et al (1996) The Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 26(1):29–41

Elbetagi E, Hegazy T et al (2005) Comparison among five evolutionary-based optimization algorithms. Adv Eng Inf 19:43–53

Engelbrecht AP (2005) Fundamentals of computational swarm intelligence. Wiley, Chichester, UK

Farmer JD, Packard NH et al (1986) The immune system, adaptation, and machine learning. Phys D 22: 187–204

Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA

Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, Michigan

Holliday D, Resnick R et al (1993) Fundamentals of physics. Wiley, New York
Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948
Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: IEEE international conference on computational cybernetics and simulation, vol 5, pp 4104–4108
Kirkpatrick S, Gelatto CD et al (1983) Optimization by simulated annealing. Science 220:671–680
Rashedi E, Nezamabadi-pour H et al (2009) GSA: a gravitational search algorithm. Inf Sci 179(13): 2232–2248
Schutz B (2003) Gravity from the ground up. Cambridge University Press, Cambridge
Srinivasa KG, Venugopal KR et al (2007) A self-adaptive migration model genetic algorithm for data mining applications. Inf Sci 177(20):4295–4313
Wang X, Yang J et al (2007) Feature selection based on rough sets and particle swarm optimization. Pattern Recogn Lett 28:459–471
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
Wu TH, Chang CC et al (2008) A simulated annealing algorithm for manufacturing cell formation problems. Expert Syst Appl 34(3):1609–1617
Yao X, Liu Y et al (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3:82–102
Youssef H, Sait SM et al (2001) Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Eng Appl Artif Intell 14:167–181
Yuan X, Nie et al (2009) An improved binary particle swarm optimization for unit commitment problem. Expert Syst Appl 36(4):8049–8055
Zeng XP, Li YM et al (2009) A dynamic chain-like agent genetic algorithm for global numerical optimization and feature selection. Neurocomputing 72:214–1228