Applied Soft Computing 13 (2013) 2932–2946

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Gases Brownian Motion Optimization: an Algorithm for Optimization (GBMO)

Marjan Abdechiri [a,*], Mohammad Reza Meybodi [b], Helena Bahrami [a]

[a] Electronic, Computer & IT Department, Qazvin Azad University, Qazvin, Iran
[b] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 12 September 2011
Received in revised form 10 March 2012
Accepted 13 March 2012
Available online 11 May 2012

Keywords:
Brownian Motion
Turbulent Rotational Motion
Evolutionary Algorithm
Optimization Algorithm
Heuristic Search Algorithm
Kinetic Energy
Satisfiability Problem
Multimodal Function
Unimodal Function
Ideal Gas Law

ABSTRACT

In recent years, different optimization methods have been developed for optimization problem. Many of these methods are inspired by swarm behaviors in nature. In this paper, a new algorithm for optimization inspired by the gases brownian motion and turbulent rotational motion is introduced, which is called Gases Brownian Motion Optimization (GBMO). The proposed algorithm is created using the features of gas molecules. The proposed algorithm is an efficient approach to search and find an optimum solution in search space. The efficiency of the proposed method has been compared with some well-known heuristic search methods. The obtained results confirm the high performance of the proposed method in solving various functions.

1. Introduction

Development of appropriate algorithms to solve complex problems such as optimization and search has been one of the most important issues in Computer Science. The global optimization methods have found applications in many fields of science such as business and engineering. The main concern of the optimization techniques is if there would be several local optimums in the system. During the last decade Evolutionary Algorithms (EAs) have been applied to optimization problems in a variety of areas [1]. Inartificial intelligence, an EA is a generic population-based meta-heuristic algorithm which has a subset of evolutionary computations. In fact, a typical EA model using the evolution process of biological populations which can adapt to the changing environment to find the optimum value through the candidate solutions. In other words, EAs are optimization techniques that work on a set of population or individuals by applying stochastic operators in order to search an optimal solution.

Different EAs have been developed for optimization which among them we can point to the Genetic Algorithm (GA) proposed by Holland [2] and inspired by the biological evolutionary process. Several other examples are the particle swarm optimization algorithm proposed by Kennedy and Eberhart [3] in 1995, the simulated annealing algorithm [4] based on the thermodynamic effects, the cultural evolutionary algorithm (CE) developed by Reynolds and Jin [5], in the early 1990s, the Ant Colony Optimization algorithm (ACO) which mimics the behavior of ants foraging for food [6], Harmony Search (HS) that is a phenomenon-mimicking algorithm inspired by the improvisation process of musicians [7] and Artificial Immune System (AIS), simulate biological immune systems [8]. Differential evolution (DE) is another optimization algorithm which has been originally introduced by Storn and Price [9] and works on multidimensional real-valued functions which are not necessarily continuous or differentiable. As a more recent developed meta-heuristic algorithm, the Charged System Search (CSS) has been introduced for design of structural problems [10]. The Gravitational Search Algorithm (GSA) presented by Rashedi et al. [11] is another EA using physics phenomena.

Atashpaz-Gargari and Lucas [12] proposed an algorithm which has been inspired by a socio-human instead of natural phenomenon. This algorithm has looked at imperialism process as a stage of human's socio-political evolution.

* Corresponding author.
E-mail address: Marjan.abdechiri@qiau.ac.ir (M. Abdechiri).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.03.068

Table 1
New evolutionary algorithms.

Name	Year	Method
Chuanwen and Bompard	2005, [14]	A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization.
Sha and Hsu	2006, [15]	Tabu search algorithm for the shop scheduling.
Shelokar et al.	2007, [16]	Particle swarm and ant colony algorithms hybridized for improved continuous optimization.
Liu et al.	2007, [17]	A fuzzy adaptive turbulent particle swarm optimization.
He and Wang	2007, [18]	Simulated annealing for the constrained optimization.
Ge et al.	2008, [19]	Artificial Immune System for the shop scheduling.
Kavehand and Talatahari	2009, [20]	Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures.
Huang	2009, [21]	The genetic watermarking for the optimized copyright protection system.
Chen et al.	2010, [22]	Unconstrained global optimization.
Abdel-Kader	2010, [23]	Genetically improved PSO algorithm for efficient data clustering.
Al-Qaheri	2010, [24]	A digital watermarking embedding and retrieval technique using ant colony optimization.
Hsu et al.	2010, [25]	A new particle swarm optimization algorithm for the multiple interference cancellation design of a linear phase array.
Chang et al.	2010, [26]	A particle swarm optimization and the genetic algorithm for cache-efficient swarm intelligence algorithms.
Hashemi and Meybodi	2011, [27]	A note on the learning automata based algorithms for adaptive parameter selection in PSO.
Thangaraj et al.	2011, [28]	Particle swarm optimization: Hybridization perspectives and experimental illustrations.

Another novel algorithm has been presented by Abdechiri et al. [13] is called Gases Brownian Motion Optimization (GBMO). GBMO algorithm takes advantage of gas molecules characteristics and their movements to propose an efficient way to optimization problems. Table 1 shows some of the new evolutionary algorithms.

In population-based optimization methods, proper control of exploration (global search) and exploitation (local search) is important in finding the optimum solution efficiently in search space [11].

In evolutionary algorithms, the exploration is the ability of finding the optima around a good solution. In early iterations, a search algorithm explores the search space to find new solutions. The exploitation guides an algorithm to find a global optimum. To avoid trapping in a local optimal, the search algorithm uses the exploration. To have a high performance search, an essential key is a suitable tradeoff between exploration and exploitation.

The objective of this paper is to present a new optimization algorithm based on principles from physics and Gases Brownian Motion, which will be called Gases Brownian Motion Optimization (GBMO). We utilize the Ideal gas law and Newton's laws. The proposed algorithm is presented by modeling of Gases Brownian Motion. In the proposed algorithm using characteristic gas molecules and to model the movement, efficient way to search for space optimization problems is proposed. The local search and global search using Brownian motion and turbulent rotational motion of each molecule during the process are implemented.

We examined the proposed algorithm in several standard benchmark functions that usually tested in Evolutionary Algorithm. The empirical results obtained from implementing the proposed algorithm with the benchmark functions indicated that the speed of convergence and the quality of solutions are better than ICA, PSO using a Sugeno function as inertia weight decline curve and GA algorithms.

The rest of this paper organized as follows. Section 2 presents the basic aspects and the characteristics of the gas molecules. In Section 3, we introduce the proposed algorithm. In Section 4, we review PSO algorithm and ICA algorithm. We introduce some functions and a problem to solve them with GBMO algorithm in Section 5. Section 4, is devoted to the empirical results of proposed algorithm implementation and its compression with the results obtained by ICA, PSO, GSA and GA algorithms to verify the efficiency of the new algorithm. In this section, the efficiency of the proposed method has been compared with some well-known heuristic search methods. The obtained results confirm the high performance of the proposed method in solving various functions and solving SAT problems. The last section includes conclusions and future work.

2. The characteristics of gas molecules

The ideal gas is a useful approximation to the behavior of many gases [29–31]. An ideal gas obeys the Ideal gas law (general gas equation): $PV = nRT$, where n is the moles of gas and R is universal gas constant [32,33]. An ideal gas is modeled on the kinetic theory of gases which has four basic assumptions for molecules motion [34] (Fig. 1):

1. Gases consist of small particles which are in continuous random motion.
2. The volume of the molecules present is negligible compared to the total volume occupied by the gas.
3. Intermolecular forces are negligible.
4. Pressure is due to the gas molecules colliding with the walls of the container.

The ideal gas can be characterized by absolute pressure (P), volume (V), and absolute temperature (T) [35]. The relationship between molecules can be described as follows:

$$PV = nRT = NKT, \tag{1}$$

where N is the number of particles in the gas; k is Boltzmann's constant which relates energy and temperature in environment. All of these units are derived from the SI units. P is measured in Pascals; V in cubic meters; N is a dimensionless number; and T in Kelvin. k has the value 1.38×10^{-23} J K^{-1} in SI units [36,37]. In Fig. 2, the parameters of ideal gas are shown.

For gas, the *pressure* is defined the force per unit area, $P = F/A$. L is length in the x-direction. If we sum N contributions, one from each particle in the box, each contribution proportional to v_x^2 for

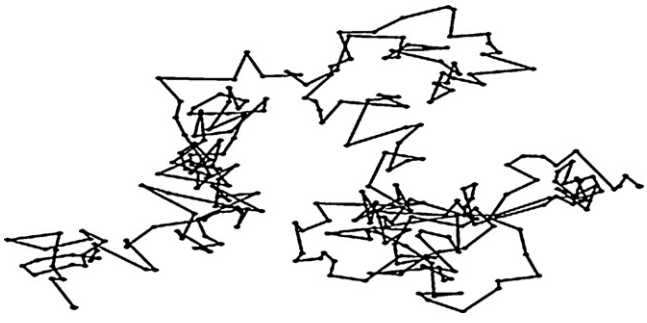

Fig. 1. Gases Brownian Motion.

- n = number of moles
- R = universal gas constant = 8.3145 J/mol K
- N = number of molecules
- k = Boltzmann constant = 1.38066×10^{-23} J/K = 8.617385×10^{-5} e V/K
- $k = R/N_A$
- N_A = Avogadro's number = 6.0221×10^{23}/mol

Fig. 2. The parameters of ideal gas.

that particle, the sum just gives us N times the *average* value of v_x^2 [38]. That is to say,

$$P = \frac{F}{A} = \frac{Nmv_x^2}{LA} = \frac{Nmv^2}{V} \tag{2}$$

Where there are N particles in a box of volume V. Next we note that the particles move in all directions, so the average value of v_x^2 must be the same as that of v_y^2 or v_z^2 hence $v^2 = v_x^2 + v_y^2 + v_z^2$ [38], it follows that

$$P = \frac{Nmv^2}{3V} \tag{3}$$

The macroscopic pressure of a gas relates directly to the average kinetic energy per molecule [38]. Boltzmann's constant $k = R/N_A = 1.38 \times 10^{-23}\,\mathrm{J\,K^{-1}}$. The average molecular kinetic energy is

$$\overline{E_k} = \overline{\frac{1}{2}mv^2} = \frac{3}{2}kT \tag{4}$$

Based on Maxwell–Boltzmann distribution of molecular speeds [39], the velocity of a molecule at temperature T can be represented by a point having coordinates (v_x, v_y, v_z) on a velocity diagram. In this way the velocity states of the whole system of N molecules is represented by N points in velocity space. Thus the number of molecules with x-component of velocity lying between v_x and $v_x + \Delta x$ and with y-component between v_y and $v_y + \Delta y$ and with z-component between v_z and $v_z + \Delta z$ is:

$$N\left(\frac{m}{2\pi KT}\right)^{1/2} e^{-((1/2mv^2)/KT)} \Delta v_x \Delta v_y \Delta v_z \tag{5}$$

The number of molecules with speeds between v and $v + \Delta v$, therefore, is the number of points in the spherical shell between radius v and radius $v + \Delta v$ that is:

$$\left[N\left(\frac{m}{2\pi KT}\right)^{3/2} e^{-((1/2mv^2)/KT)}\right](4\pi v^2 \Delta v)$$
$$= 4\pi v^2 N\left(\frac{m}{2\pi KT}\right)^{3/2} e^{-((1/2mv^2)/KT)} \Delta v \tag{6}$$

Dividing by N we see that the fraction of molecules in a gas at temperature T with speed between v and $v + \Delta v$ is given by:

$$P(v)\Delta v = 4\pi\left(\frac{m}{2\pi KT}\right)^{3/2} v^2 e^{-((1/2mv^2)/KT)} \Delta v \tag{7}$$

The function $P(v)$ gives the Maxwell–Boltzmann distribution of molecular speeds [39].

(i) Average speed: the average speed of the molecules is defined by equation $v = \int_0^\infty vP(v)dv$ and hence using the Maxwell – Boltzmann distribution results in:

$$v = 4\pi\left(\frac{m}{2\pi KT}\right)^{3/2} \int_0^\infty v^3 e^{-(mv^2/2KT)} dv = \sqrt{\frac{8KT}{\pi m}} \approx 1.6\sqrt{\frac{KT}{m}} \tag{8}$$

(ii) RMS speed: the average value of the square of the speed can be determined as follow:

$$v^2 = \int_0^\infty v^2 P(v)dv = 4\pi\left(\frac{m}{2\pi KT}\right)^{3/2} \int_0^\infty v^4 e^{-(mv^2/2KT)} dv = \frac{3KT}{m} \tag{9}$$

And hence the *rms* speed is given by:

$$v_{rms} = \sqrt{v^2} = \sqrt{\frac{3KT}{m}} \approx 1.7\sqrt{\frac{KT}{m}} \tag{10}$$

These molecules are in constant irregular motion with a velocity proportional to the square root of the temperature. The effective velocity for each molecule in space can be obtained using Eq. (10). This is expressed as

$$v = \sqrt{\frac{3kT}{m}} \tag{11}$$

Note that this result shows that $T \to 0$ as $V \to 0$. So this system is stable in $T = 0$. The velocity for each molecule is

$$v(t+1) = v(t) + \sqrt{\frac{3kT}{m}} \tag{12}$$

Velocity is defined as

$$v = \frac{\Delta x}{\Delta t} \tag{13}$$

where Δx is the change in position and Δt is the interval of time [40].

$$v\Delta t = x(t+1) - x(t) \tag{14}$$

So for the proposed algorithm $\Delta t = 1$

$$v(t) + x(t) = x(t+1) \tag{15}$$

3. Gases Brownian Motion Optimization

The GBMO algorithm is based on the law of motion and Gases Brownian Motion and turbulent rotational motion for finding the optimal solution. In the proposed algorithm, the agents are molecules and their performance is measured by their positions. Each position presents a part of solution, and the algorithm is directed by properly adjusting the Gases Brownian Motion and turbulent rotational motion.

All these molecules move in the space of problem, and this motion causes a movement of all molecules towards the solution. $T = 0$ guarantees convergence of GBMO algorithm. In GBMO, molecule has four specifications: position, mass, velocity and radius of turbulent. The different steps of the proposed algorithm are the followings:

Step 1: Randomized initialization. The GBMO algorithm starts with an initial population (Molecules in the space). In fact, it initializes an array of molecules with random positions.

Step 2: For each molecule is considered a random a radius of turbulence will be considered (the radius is a random variable in the range [0,1]).

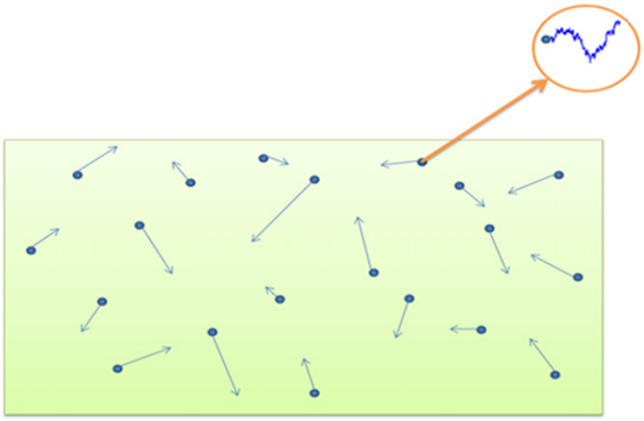

Fig. 3. The turbulent rotational motion and Gases Brownian Motion.

Step 3: A temperature will be assigned to the system.

In GBMO algorithm, temperature guarantees convergence of the GBMO algorithm. Temperature can affect velocity of molecules. The temperature plays a very important role in controlling the exploitation and exploration ability of the algorithm.

At first, the Gases Brownian Motion is a global movement of all molecules towards the objects. The lighter masses which correspond to good solutions – move faster than heavier ones, this guarantees the exploitation step of the algorithm. By lapse of time and decrease in temperature of system, a decrease in temperature causes the random Gases Brownian Motion to decrease. During the early part of the search, high temperature in the gas cause to increase kinetic energy and velocity of molecules. So the Gases Brownian Motion causes global exploration of the full range of the search space and turbulent rotational motion has a good performance for local search.

In the last stage of the search, local exploitation will be achieved for more accuracy in the optimal solution. In low temperatures, turbulent rotational motion mainly leads to global search and Gases Brownian Motion play the major role in local search of the problem space.

In Fig. 3, the turbulent rotational motion and Gases Brownian Motion have been shown in detail.

One way to perform a good tradeoff between exploration and exploitation is to reduce temperature with lapse of time and consequently, changing the roles of Gases Brownian Motion with turbulent rotational motion (global search to local search) during the search. This policy (changing roles) is essential for a good convergence and provides a suitable solution for the problem.

During the search, molecules have a high velocity, therefore, Gases Brownian Motion has exploration ability in the search space and turbulent rotational motion has exploitation ability. By temperature reduction, the effects will be reversed.

Step 4: updating velocity and position

In GBMO algorithm, x_i^d and v_i^d would be calculated as follows:

$$v_i^d(t+1) = v_i^d(t) + \sqrt{\frac{3kT}{m}} \tag{16}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t) \tag{17}$$

Where $x_i = (x_i^1, x_i^2, \ldots, x_i^n)$ and $V_i = (v_i^1, v_i^2, \ldots, v_i^n)$ represent position and velocity of the ith molecules.

Step 5: Fitness evaluation of agents. In this stage the values of the fitness function for the molecules will be evaluated.

This optimization approach update the population of molecules by fitness information obtained from the individuals.

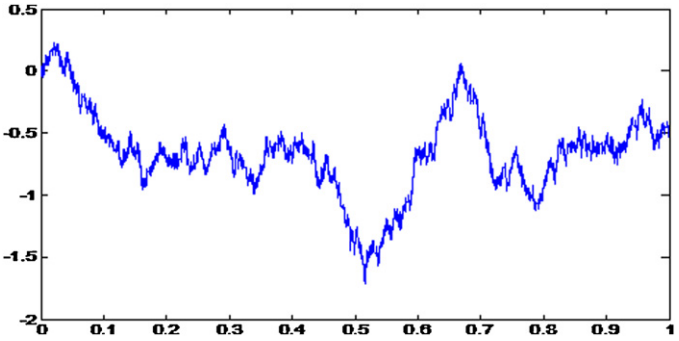

Fig. 4. Turbulent rotational motion.

Step 6: Every molecule, besides to the Brownian Motion has a turbulent rotational motion (vibration in specific radius that the radius is a random number in the interval [0,1]).

The turbulent rotational motion is modeled by chaotic sequence generator called Circle map [41] and it is represented by

$$x_i^d(t+1) = x_i^d(t) + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_i^d(t)) \bmod (1) \tag{18}$$

With $a = 0.5$ and $b = 0.2$, it generates a chaotic sequence in (0, 1) and x_i^d is the current position in turbulent rotational motion .The GBMO is a stochastic search algorithm. Turbulent rotational motion has been shown in Fig. 4.

In early iterations the local search comes from turbulent rotational motion in the search space. This local search embeds in early iterations of the GBMO algorithm and manages the performance of the algorithm. At the final iterations, this motion performs global search in the solution space.

Step 7: Evaluate and compare the values of the objective function with the new positions of molecules.

Step 8: Updating mass and temperature values.

A lighter mass means a more efficient molecule with a higher velocity. The values of masses are calculated using the fitness evaluations. We update the molecules' masses by the following equations:

$$m_i(t) = \frac{\mathrm{fit}_i(t) - \mathrm{worst}(t)}{\mathrm{best}(t) - \mathrm{worst}(t)} \tag{19}$$

Where $\mathrm{fit}_i(t)$ represent the fitness value of the molecule i at the time t, and worst(t) and best(t) are defined as follows (for a minimization problem):

$$\mathrm{best}(t) = \min_{i \in \{1,\ldots,N\}} \mathrm{fit}_i(t) \tag{20}$$

$$\mathrm{best}(t) = \max_{i \in \{1,\ldots,N\}} \mathrm{fit}_i(t) \tag{21}$$

the temperature will be updated by the following equation:

$$T = T - \left(\frac{1}{\mathrm{mean}(\mathrm{fit}_i(t))}\right) \tag{22}$$

where $\mathrm{fit}_i(t)$ represent the fitness value of the molecule i at the time t.

Step 9: Repeating the search steps 3–7 until the stop criteria is reached.

The flowchart of the GBMO algorithm is illustrated in Fig. 5. In the GBMO algorithm, the local search space and global search using turbulent rotational motion and Brownian motion of each molecule during the process are implemented.

4. Comparative study

To start, review PSO algorithm and ICA algorithm. In this section, we introduce these algorithms and explain some details about them.

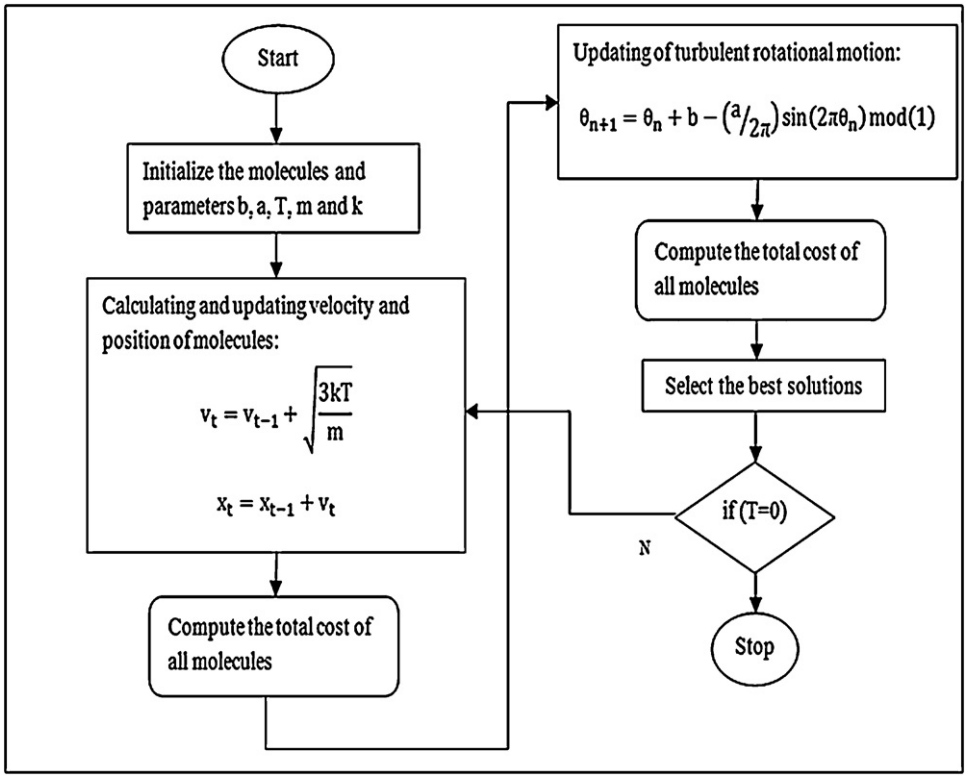

Fig. 5. Flowchart of GBMO algorithm.

4.1. Particle Swarm Optimization

PSO is motivated by social behavior (such as flock of birds). PSO provides population-based search. In the PSO algorithm, individuals called particles. PSO update the position of particles by the fitness information obtained from the environment so that the individuals of the population can be expected to move towards the better solution. In PSO, position x_i^d and velocity v_i^d are calculated as follows [3]:

$$X_i^d(t+1) = X_i^d(t) + v_i^d(t+1) \quad (23)$$

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_{i1}(pbest_i^d - X_i^d(t)) + c_2 r_{i2}(gbest_i^d - X_i^d(t)) \quad (24)$$

Where c_1 and c_2 are positive constants, w is the inertia weight and r_{i1} and r_{i2} are two random variables in the range [0,1].

pbest and *gbest* represent the best previous position of the *i*th particle and the best previous position among all the particles in the population, respectively.

In GBMO and PSO algorithms the optimization is obtained by particles (or molecules) movement in the search space. But in PSO the new position of a particle is calculated using *pbest* and *gbest*. But in GBMO, the new position of molecule is calculated based on the best positions of Gases Brownian Motion and turbulent rotational motion obtained by all other molecules. In PSO algorithm, updating is performed without considering the fitness values while in GBMO the mass of molecules and temperature are proportional to the fitness value.

PSO uses a kind of memory for velocity but GBMO uses memoryless. The search ideas of these algorithms are different. PSO inspires by the social behavior of birds and GBMO inspires by a gases motion.

Finally, In the GBMO algorithm search space is faster than the PSO algorithm because the velocity of molecules is very fast. In GBMO algorithm local search with global search capability and this policy is able to exploit and explore the solution space more effectively.

4.2. Imperialist Competitive Algorithm (ICA)

Imperialist Competitive Algorithm (ICA) is a new evolutionary algorithm in the evolutionary computation field based on the human's socio-political evolution [12]. The algorithm starts with an initial random population called countries. Some of the best countries in the population selected to be the imperialists and the rest form the colonies of these imperialists. In an N dimensional optimization problem, a country is a $1 \times N$ array. This array defined as below

$$\text{country} = [p_1, p_2, \ldots, p_N] \quad (25)$$

The cost of a country is found by evaluating the cost function f at the variables $(p_1, p_2, p_3, \ldots, p_N)$. Then

$$c_i = f(\text{country}_i) = f(p_{i1}, p_{i2}, \ldots, p_{iN}) \quad (26)$$

The algorithm starts with N initial countries and the N_{imp} best of them (countries with minimum cost) chosen as the imperialists. The remaining countries are colonies that each belongs to an empire. The initial colonies belong to imperialists in convenience with their powers. To distribute the colonies among imperialists proportionally, the normalized cost of an imperialist is defined as follows:

$$C_n = \max_i c_i - c_n \quad (27)$$

Where c_n is the cost of nth imperialist and C_n is its normalized cost. Each imperialist that has more cost value, will have less normalized cost value. Having the normalized cost, the power of each imperialist is calculated as below and based on that the colonies distributed among the imperialist countries.

$$p_n = \left| \frac{C_n}{\sum_{N_{imp}}^{i=1} C_i} \right| \quad (28)$$

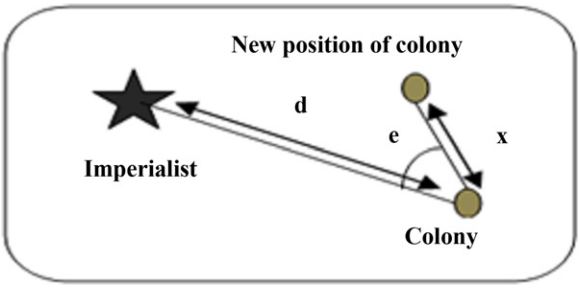

Fig. 6. Moving colonies toward their imperialist.

On the other hand, the normalized power of an imperialist is assessed by its colonies. Then, the initial number of colonies of an empire will be

$$NC_n = \text{rand}\left\{p_n.(N_{col})\right\} \tag{29}$$

where NC_n is initial number of colonies of nth empire and N_{col} is the number of all colonies. To distribute the colonies among imperialist, NC_n of the colonies is selected randomly and assigned to their imperialists. The imperialist countries absorb the colonies towards themselves using the absorption policy. The absorption policy shown in Fig. 6, makes the main core of this algorithm and causes the countries move towards to their minimum optima. The imperialists absorb these colonies towards themselves with respect to their power that described in Eq. (30). The total power of each imperialist is determined by the power of its both parts, the empire power plus percent of its average colonies power.

$$TC_n = \text{cost}(\text{imperialist}_n) + \xi\text{mean}\{\text{cost}(\text{colonies of empire}_n\} \tag{30}$$

Where TC_n is the total cost of the nth empire and ξ is a positive number which is considered to be less than one.

$$x \sim U(0, \beta \times d) \tag{31}$$

In the absorption policy, the colony moves towards the imperialist by x unit. The direction of movement is the vector from colony to imperialist, as shown in Fig. 6, in this figure, the distance between the imperialist and colony shown by d and x is a random variable with uniform distribution. Where β is greater than 1 and is near to 2. So, a proper choice can be $\beta = 2$. In our implementation γ is $\pi/4$ (*Rad*) respectively.

$$\theta \sim U(-\gamma, \gamma) \tag{32}$$

In ICA algorithm, to search different points around the imperialist, a random amount of deviation is added to the direction of colony movement towards the imperialist. In Fig. 6, this deflection angle is shown as θ, which is chosen randomly and with the uniform distribution. While moving toward the imperialist countries, a colony may reach to a better position, so the colony position changes according to position of the imperialist.

In this algorithm, the imperialistic competition has an important role. During the imperialistic competition the weak empires will lose their power and their colonies. To model this competition, firstly we calculate the probability of possessing all the colonies by each empire considering the total cost of empire.

$$NTC_n = \max_i\left\{TC_i\right\} - TC_n \tag{33}$$

Where, TC_n is a total cost of nth empire and NTC_n is the normalized total cost of nth empire. Having the normalized total cost, the possession probability of each empire is calculated as below.

$$p_{p_n} = \left| \frac{NTC_n}{\sum_{N_{imp}}^{i=1} NTC_i} \right| \tag{34}$$

After a while all the empires except the most powerful one will collapse and all the colonies will be under the control of this unique empire.

In ICA algorithm, some of best countries in the population are selected to be the imperialists and the rest form the colonies of these imperialists. One of major differences is that in the ICA algorithm all the colonies of initial population are divided among the mentioned imperialists based on their power.

The movement in GBMO is faster than the ICA, GA and PSO algorithms. The calculation of the masses and temperature in GBMO are different from PSO. All the calculations in GBMO are not different from ICA and GA.

5. The problems

In this section, we introduce some functions and a problem to solve them with GBMO algorithm.

5.1. Satisfiability problem

The satisfiability (SAT) problems belong to an important class of discrete constraint satisfaction problems (CSP) and NP-complete problems [42]. The SAT problem is a set of n clauses $\{C_1, C_2, \ldots, C_n\}$ on m variables $x = (x_1, x_2, \ldots, x_m)$, $x_i \in \{0, 1\}$ and a Boolean formula in conjunctive normal form (CNF)

$$C_1 \Lambda C_2 \Lambda \ldots \Lambda C_n \tag{35}$$

A *clause* is a combination of k *literals* where a literal is a Boolean variable x_i or its negation ($\bar{x}_i$). If the number of literals is k for all clauses, then SAT is referred to as k-SAT. The goal of the SAT problem is to determine whether there exists an assignment of truth values to variables that makes the CNF formula satisfiable. Some of the algorithms are used to solve SAT problems in continue space and some of them are presented in discrete space.

The propositional satisfiability (or SAT) problem is one of researched problems in computer science and artificial intelligence, mathematics, machine vision, robotics and computer-aided manufacturing.

In this paper, we use the fitness function, based on a floating point representation. An objective in continuous space may "smooth out" some infeasible solutions, leading to a smaller number of local minima explored. In the following, we show two such formulations. A simple formulation, UniSAT (Universal SAT Problem Models) [43], suggests:

$$\min_{y \in E^n} f(y) = \sum_{i=1}^{m} c_i(y) \tag{36}$$

where

$$c_i(y) = \prod_{j=1}^{n} q_{i,j}(y_j) \tag{37}$$

$$q_{i,j}(y_j) = \begin{cases} \left|y_j - 1\right| & \text{if } x_j \text{in } c_i \\ \left|y_j + 1\right| & \text{if } \overline{x_j} \text{ in } c_i \\ 1 & \text{otherwise} \end{cases} \tag{38}$$

Values of y that make $f(y) = 0$ are solutions to the original formula in Eq. (38). Many different approaches based on neural networks [44–46], genetic algorithms [47], simulated annealing [48], tabu search [49] and hybrid techniques [50–53] have been proposed. In [54] the cellular memetic algorithms (cMAs) introduced. In this paper, we examine the performance of GBMO on a test suite of satisfiability problems.

Table 2
Benchmarks for simulation.

Functions	Functions	Bounds	Dimension	Type
Sphere	$f1(x)=\sum_{i=1}^{D} x_i^2$	(−100,100)	N	UM[a]
Rastrigrin	$f2(x)=\sum_{i=1}^{D}(x_i^2-10\times\cos(2\pi x_i)+10)$	(−10,10)	N	MM[b]
Rosenbrock	$f3(x)=\sum_{i=1}^{D-1}(100\times(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	(−100,100)	N	UM
Griewank	$f4(x)=\frac{1}{4000}\times\sum_{i=1}^{D}x_1^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}+1)$	(−600,600)	N	MM
Ackley	$f5(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_j^2})-\exp(\frac{1}{n}\sum_{k=1}^{n}\cos 2\pi x_k)+20+e$	(−32,32)	N	MM
Booth	$f6=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	(−10,10)	2	UM
Zakharov	$f7=z_n(x)=\sum_{i=1}^{n}x_i^2+(\sum_{i=1}^{n}0.5ix_i)^2+(\sum_{i=1}^{n}0.5ix_i)^4$	(−5,10)	2	UM

[a] Unimodal.
[b] Multimodal.

5.2. Benchmark functions

Many real-world problems can be formulated as optimization problems most of which can be converted to a minimization problem. The optimization algorithms use some well-known benchmark functions to show their abilities. In order to compare and evaluate different algorithms, various benchmark functions with various properties have been proposed. In the following we show some well-known benchmark functions and their ranges. Table 2 shows them. In this paper, we use the problem instances for satisfiability problem. Table 3 shows these instances and the number of variables and clauses in each instance.

We made simulations for considering the rate of convergence and the quality of the proposed algorithm optima solution, in comparison to ICA, PSO using a Sugeno function as inertia weight and GA algorithms that all the benchmarks tested by 30 dimensions separately. The average of optimum value for 20 trails obtained. In these experiments, all the simulations done during 1000 generations for Sphere and Rosenbrock uni-modal functions and Rastrigin, Griwank, Ackley and Michalewicz multi-modal functions.

5.3. Real world optimization problems

The real world optimization problems can be used to evaluate the performance of Gases Brownian Motion Optimization algorithm. In this section two real world problems to evaluate the performance of the proposed algorithm are defined.

5.3.1. Lennard–Jones potential problem

Lennard–Jones Potential Problem is a potential energy minimization problem [55]. Lennard–Jones potential problem involves the minimization of molecular potential energy associated with pure Lennard–Jones (LJ) cluster [56,57]. The proposed algorithm can be tasted over this function for its capability to conform molecular structure, where the atoms are organized in such a way that the molecule has minimum energy. Lennard–Jones pair potential for N atoms, given by the Cartesian coordinates

$$\vec{p}_i=\left\{\vec{x}_i,\vec{y}_i,\vec{z}_i\right\},\ i=1,\ldots,N \tag{39}$$

is given as follows:

$$V_N(p)=\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(r_{ij}^{-12}-2r_{ij}^{-6}) \tag{40}$$

where $r_{ij=}\overrightarrow{p}_j-\overrightarrow{p_{i2}}$ with gradient:

$$\nabla_j V_N(p)=-12\sum_{i=1,i\neq j}^{N}(r_{ij}^{-14}-r_{ij}^{-8})(\vec{p}_j-\vec{p}_i),j=1,2,\ldots,N \tag{41}$$

Lennard–Jones potential has minimum value at a particular distance between the two points [58].

5.3.2. Tersoff Potential Function Minimization Problem

Tersoff potential problem is about evaluation of inter atomic potentials for covalent systems, particularly for Silicon. Tersoff has given two parameterization of silicon and these are called Sc(B) and Sc(C). If we define the positions of the molecular clusters of N atoms by:

$$\vec{X}=\left\{\vec{X}_1,\vec{X}_2,\ldots,\vec{X}_N\right\} \tag{42}$$

Where $\vec{X}_i$, $i\in\left\{1,2,\ldots,N\right\}$ is a three-dimensional vector denoting the coordinates of the ith atom in Cartesian coordinate system then the total Tersoff potential energy function of the system is a function of atomic co-ordinates and defined as

$$f(\vec{X}_1,\vec{X}_2,\vec{X}_3)=E_1(\vec{X}_1,\vec{X}_2,\vec{X}_3)+\ldots+E_N(\vec{X}_1,\vec{X}_2,\vec{X}_3) \tag{43}$$

Total potential is sum of individual potentials of atoms given by Eq. (43), which further needs calculation of Eq. (44) through (50). The potential of any atom depends on the physical position of their neighbor atoms. Different atoms have different distances and angles subtended with respect to the other atoms so every atom has deferent energies. Now the Tersoff potential [58] of individual atoms can be formally defined as

$$E_i=\frac{1}{2}\sum_{j\neq i}f_c(r_{ij})(V_R(r_{ij})-B_{ij}V_A(r_{ij})),\forall i \tag{44}$$

where r_{ij} is the distance between atoms i and j, V_R is a repulsive term, V_A is an attractive term, $f_c(r_{ij})$ is a switching function and B_{ij} is a many-body term that depends on the positions of atoms i and j and the neighbors of atom i. These quantities are detailed in [58].

$$B_{ij}=(1+\gamma^{n_i}\xi_{ij}^{n_i})^{1/2n_i} \tag{45}$$

Where n_i and γ are known fitted parameters [58]. The term $\xi_{ij}^{n_i}$ for atoms i and j (i.e., for bond ij) is given by:

$$\xi_{ij}=\sum_{k\neq i}f_c(r_{ij})g(\theta_{ijk})\exp(\lambda_3^3(r_j-r_k)^3) \tag{46}$$

Here ξ_{ij} describes the contribution of the neighbors of the atom i, ξ_{ij} increases as the number of k atoms increases but the term B_{ij} decreases as ξ_{ij} increases. The exponential term in Eq. (46) is designed to reduce the contribution of bonds with length greater than r_{ij}, so that the distant neighbors of have a reduced contribution to the bond order term. The term θ_{ijk} is the bond angle between bonds ij and ik, and the function g is given by:

$$g(\theta_{ijk})=1+\frac{c^2}{d^2}-\frac{c^2}{d^2+(h-\cos(\theta_{ijk}))^2} \tag{47}$$

The parameter h is the cosine of the optimum bond angle and c and d control the influence of bond angles on the many-body term. The

quantities λ_3, c, d and h which appear in Eqs. (46) and (47) are also known fitted parameters. The terms V_R and V_A are given by:

$$V_R(r_{ij}) = Ae^{-\lambda_1 r_{ij}} \tag{48}$$

$$V_A(r_{ij}) = Be^{-\lambda_2 r_{ij}} \tag{49}$$

where A, B, λ_1 and λ_2 are given fitted parameters. The switching function $f_c(r_{ij})$ restricts the potential calculations to nearest neighbors only and ensures that atomic interactions decay smoothly to zero as the separation distance increases. It is given by

$$f_c(r_{ij}) = \begin{cases} 1 & \text{if } r_{ij} \le R - D \\ \frac{1}{2} - \frac{1}{2}\sin\left[\frac{\pi(r_{ij} - R)}{D}\right] & \text{if } R - D < r_{ij} < R + D \\ 0 & \text{if } r_{ij} \le R - D \end{cases} \tag{50}$$

Though Tersoff potential problem is $n \times 3$ dimensional in 3-dimentional space, the number of dimensions to be evaluated can be decreased in light of the fact that it depends on relative position of atom instead of actual Cartesian coordinates. One atom can be permanently put on the origin and second atom on the positive x-axis. Thus for a 3-atom system the actual number of variables is four instead of nine. Now, for each additional atom added to the system, the no. of variables increases by three (three Cartesian coordinates of an additional atom). Thus, in general, for the system of N atoms, the no. of unknown variables is $n = 3 \times N - 6$. Now the cluster X of N atoms can be redefined as

$$x = \{x_1, x_2, \ldots, x_n\}, x \in R^{(3N-6)} \tag{51}$$

The search region for both Si(B) and Si(C) model of Tersoff potential is

$$\Omega = \Big\{\{x_1, x_2, \ldots, x_n\} \mid -4.25 \le x_1, x_i \le 4.25, i = 4, \ldots, n, \\ 0 \le x_2 \le 4, 0 \le x_3 \le \pi\Big\} \tag{52}$$

The potential Eq. (52) is a complex and differentiable function whose partial derivatives can be found in [59]. The cost function now can be redefined as:

$$f(x) = E_1(x) + E_2(x) + \ldots + E_N(x), x \in \Omega \tag{53}$$

6. Experimental results

To study the performance of the proposed algorithm, we conduct three groups of simulation. The first group, it evaluate the proposed algorithm on solving the SAT problems. The second group of the simulation is concerned with investigating the efficiency of the proposed algorithm on some well-known benchmark functions. The third group, some real world optimization problems are used to evaluate the performance of Gases Brownian Motion Optimization algorithm.

6.1. Solving the SAT problem

We present a detailed comparative analysis of the proposed algorithm's performance, using some of the benchmark set containing instances from SAT problems from various domains. Experimental results show GBMO algorithm can improve the performance of the Solving SAT problems significantly.

In Table 3, we present the results for the 18 instances of SAT problems. All the algorithms have been tested in terms of efficiency Success Rate (SR). The results have been obtained after 50 runs of the algorithms for every instance. The results show that GBMO algorithm is suitable for solving SAT problems special for large instances.

For solving a larger number of instances, GBMO provide a suitable solution. The ICA algorithm suffers the lack of ability to global search properly in the problem space (exploitive capacity). During the search process, the algorithm may trap into local optima. For each strategy we give the average time in seconds it took to find a satisfying assignment and the number of satisfy clauses. The GBMO algorithm is suitable for a large number of instances. In Tables 4–6, this result is a consequence of the fact that GBMO has a good performance for function optimizations and solving the SAT problems. GBMO is more accurate than CMA and CGA for this set of instances.

In Fig. 7 influence of the number of molecules on the solutions quality is shown. In Fig. 8 influence of the number of molecules on the execution time is shown. We can see when the number of molecules increases, the quality of solutions is improved and the execution time rises. When the aim is to obtain good solutions, we should increase the number of molecules. On the other hand, if execution time should be short, we should decrease the number of

Table 3
Benchmarks for simulation.

Instance	Variables	Clauses
Uniform random-3-SAT		
Uf50-218	50	218
Uf75-325	75	325
Uf100-430	100	430
Uf125-538	125	538
Uf150-645	150	645
Uf175-753	175	753
SAT-encoded "Flat" graph coloring problems		
flat50-115	150	545
flat75-180	300	1117
flat100-239	300	1117
flat125-301	375	1403
flat150-360	450	1680
flat175-417	525	1951
SAT-encoded logistics planning problems		
Logistics.a	828	6718
Logistics.b	843	7301
Logistics.c	1141	10,719
Logistics.d	6325	131,973
N queens		
10 queens	100	1480
20 queens	400	12,560
50 queens	2500	203,400
100 queens	10,000	1,646,800
Random permutation generation		
pp50	2475	159,138
pp60	3568	279,305
pp70	4869	456,129
pp80	6356	660,659
pp90	8059	938,837
pp100	9953	1,265,776
Increasing permutation generation		
ap10	121	671
ap20	441	4641
ap30	961	14,911
ap40	1681	34,481
Latin square		
magic-10	1000	9100
magic-15	3375	47,475
magic-20	8000	152,400
magic-25	15,0625	375,625
magic-30	27,000	783,900
magic-35	42,875	1,458,975
Hard graph-coloring		
g12n-18c	2250	70,163
g250n-15c	3750	233,965
g125n-17c	2125	66,272
g250n-29c	7250	454,622
Satisfiable variants of the pipe benchmarks		
12pipe_bug1	118,040	8,804,672
12pipe_bug2	118,040	8,804,630
12pipe_bug3	118,039	8,804,669
The pigeon hole problem		
hole6.cnf	42	133
hole7.cnf	56	204
hole8.cnf	72	297

Table 4
Comparison of best-cost for solving the SAT problems.

	ICA	PSO	GA	GBMO
Uf50-218	0	0	0	0
Uf75-325	0	0	0	0
Uf100-430	90.2	110.4	189.7	100.6
Uf125-538	373.4	430.4	461.2	208.9
Uf150-645	474.6	216.2	542.2	210.0
Uf175-753	479.6	421.4	602.6	389.6
flat175-417	964.4	955.3	970.4	954.4
flat200-479	1005.1	1000.2	1000.9	967.8
logistics.a	1045.9	1030.7	1035.6	986.5
logistics.b	1097.2	1077.5	1085.2	1054.8
logistics.c	1134.8	1131.6	1109.1	1127.6
logistics.d	1167.3	1145.6	1159.4	1142.7
10 queens	540.0	556.8	734.4	391.3
pp50	2044.5	2411.1	2150.5	1994.6
ap10	991.5	943.5	875.2	754.0
g12n-18c	2722.1	2463.3	2455.6	2004.5
hole6.cnf	261.2	250.5	243.8	206.1
hole7.cnf	367.0	342.5	312.2	254.1

molecules. In GBMO is a trade-off between the quality of solutions and execution time.

The GASAT [60] algorithm is a genetic algorithm for SAT problem in discrete space. In GASAT algorithm, each individual is a binary string where represents the value of variables. The fitness function of GASAT is equal to the number of clauses which are not satisfied by this member. In this algorithm, each step is run tabu search. The tabu search chooses the best variable to flip.

The SAT-WAGA [61] is a genetic algorithm for the SAT problem. In SAT-WAGA algorithm, each individual is a binary string where represents the value of variables. In this algorithm each clause has a weight. The fitness function of a member returns the sum of all the clauses' weights which are satisfied by this member.

We adjusted all parameters of algorithms and found the best parameters.

Table 6
Comparison of results for solving the SAT problems.

Formula Variables	CGA [30] SR	CMA [33] SR	GBMO SR
100	83%	88%	100%
200	77%	79%	98%
400	63%	73%	88%
600	60%	69%	86%
800	56%	58%	83%
1000	32%	46%	75%

The GASAT parameters are:

- the population size: 30,
- the best individuals size: 15,
- the hamming distance: 1,
- the tabu list length: (the variables number)/20,
- the flips in the tabu search: 1000.

The SAT-WAGA parameters are:

- the population size: 10
- the parent size: 10,
- the crossover probability: 55%,
- the mutation probability: 3%,
- $\delta = 0.1(w_n(C))$.

Table 7, shows the results of SAT-WAGA, GASAT, PSO, ICA and GBMO. We run each instance for 20 runs. The information presented the success ratio of the 20 runs. (*) in this table means the algorithm gives fault solution while running the instance.

The performance of GASAT is best SATWAGA because GASAT uses the local search algorithm (Tabu search). We ran GASAT and SAT-WAGA by using the same parameters and we found that SAT-WAGA has very poor performance. In the N queens problem, 20

Table 5
Comparison of results for solving the SAT problem.

Formula		ICA		PSO		GBMO	
Variables	Clauses	Time	Rate of satisfy	Time	Rate of satisfy	Time	Rate of satisfy
100	430	407.378299	100%	500.756060	100%	504.001287	100%
200	860	664.298466	81%	607.441593	95%	619.341290	98%
400	1700	709.677022	66%	619.223453	89%	634.602132	88%
600	2550	804.234178	62%	773.770900	80%	752.991042	86%
800	3400	859.733425	50%	818.452552	76%	805.400300	83%
1000	4250	903.726760	42%	832.561658	64%	824.910671	75%

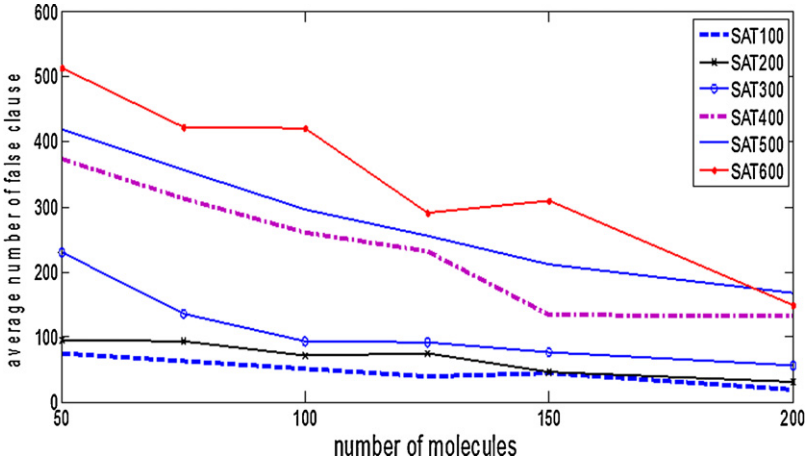

Fig. 7. Influence of the number of molecules on the solutions quality.

Table 7
Comparison of results for solving the SAT problems.

Instance	The rate of success in SAT-WAGA algorithm (DS)	The rate of success in GASAT algorithm (DS)	The rate of success in GASAT algorithm (CS)	The rate of success in PSO-SAT algorithm (DS)	The rate of success in PSO algorithm (CS)	The rate of success in ICA algorithm (CS)	The rate of success in GBMO algorithm (CS)
10 queens	20/20	20/20	20/20	20/20	20/20	20/20	20/20
20 queens	18/20	20/20	20/20	20/20	20/20	20/20	20/20
50 queens	0/20	20/20	20/20	20/20	20/20	20/20	20/20
100 queens	0/20	0/20	1/20	0/20	20/20	19/20	20/20
pp50	0/20	20/20	20/20	20/20	20/20	20/20	20/20
pp60	0/20	0/20	0/20	0/20	1/20	0/20	10/20
pp70	0/20	0/20	0/20	0/20	0/20	0/20	2/20
pp80	0/20	0/20	0/20	0/20	0/20	0/20	0/20
pp90	0/20	0/20	0/20	0/20	0/20	0/20	0/20
pp100	0/20	0/20	0/20	0/20	0/20	0/20	0/20
ap10	0/20	10/20	11/20	3/20	20/20	15/20	20/20
ap20	0/20	0/20	5/20	0/20	10/20	10/20	20/20
ap30	0/20	0/20	0/20	0/20	0/20	0/20	3/20
ap40	0/20	0/20	0/20	0/20	0/20	0/20	0/20
magic-10	0/20	0/20	1/20	0/20	5/20	4/20	10/20
magic-15	0/20	3/20	3/20	3/20	3/20	3/20	3/20
magic-20	0/20	3/20	3/20	3/20	12/20	12/20	20/20
magic-25	0/20	0/20	0/20	0/20	0/20	0/20	4/20
magic-30	0/20	0/20	0/20	0/20	0/20	0/20	0/20
magic-35	0/20	0/20	0/20	0/20	0/20	0/20	0/20
g12n-18c	0/20	20/20	20/20	20/20	20/20	20/20	20/20
g250n-15c	0/20	20/20	20/20	20/20	20/20	20/20	20/20
g125n-17c	0/20	20/20	20/20	20/20	20/20	20/20	20/20
g250n-29c	0/20	0/20	0/20	0/20	0/20	0/20	0/20
12pipe_bug1	0/20	0/20	0/20	0/20	0/20	0/20	5/20
12pipe_bug2	0/20	0/20	0/20	0/20	0/20	0/20	0/20
12pipe_bug3	0/20	0/20	0/20	0/20	0/20	0/20	0/20
hole6.cnf	20/20	20/20	20/20	20/20	20/20	20/20	20/20
hole7.cnf	20/20	20/20	20/20	20/20	20/20	20/20	20/20
hole8.cnf	5/20	8/20	1/20	1/20	10/20	9/20	20/20

queens has bigger search space than 10 queens so local search takes more time to solve 20 queens than 10 queens. The performance of GBMO is the best SATWAGA and GSAT. The performance of GBMO is the best PSO and ICA algorithm for solving SAT problems.

Solving SAT problem in continue space (CS) is not as easy as solving this problem in discrete space (DS). But GBMO has good performance for solving this problem in continue space. The performance of GBMO is as well as discrete methods such as GSAT and SATWAGA.

6.2. The optimization benchmark functions

To evaluate the performance of GBMO algorithm for finding the global optimum, we applied GBMO to 7 standard benchmark functions [62].

These benchmark functions are presented in Table 3. All these problems are minimization problems. We applied GBMO to these minimization functions and compared the results with GA, PSO, GSA and ICA. In all cases, population size is set to 80 ($n = 80$). Dimension is set to ($n = 10$, 20 and 30) and maximum iteration is 1000 for functions of Table 8. In PSO, $c_1 = c_2 = 1.5$ and inertia factor (w) is decreasing linearly from 0.9 to 0.2. In GA, crossover, Gaussian mutation and roulette wheel selection are used as described in [63]. The crossover and mutation probabilities were set to 0.4 and 0.01.

Functions F1, F3, F6 and F7 are unimodal functions. In this case the convergence rate of search algorithm is more important for Unimodal functions than the final results. Functions F2, F4 and F5 are Multimodal functions. F1–F5 are n-dimensional function and F6, F7 are 2-dimentional.

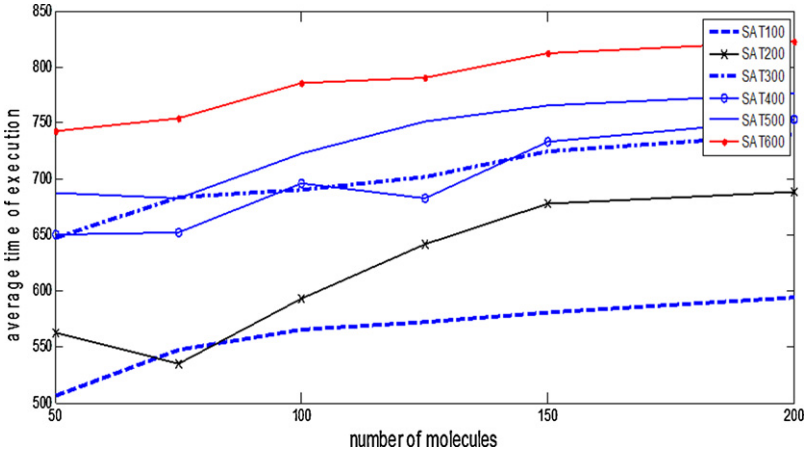

Fig. 8. Influence of the number of molecules on the execution time.

Table 8
The parameters.

Algorithm	Parameters
ICA	Number of countries = 80, number of initial imperialists = 8, number of decades = 1000, revolution rate = 0.3, assimilation coefficient = 2, zeta = 0.02.
GA	Pop size = 80, max generations = 1000, cross percent = 40/100, Mutation rate = 0.01; selection mode: roulette wheel
PSO	$c_1 = 1.5$, $c_2 = 1.5$, $\beta = iter/iter_{max}$, $S = -0.5$, particle size = 80, max generations = 1000
GBMO	Molecule size = 80, temperature = 1000, $a = 0.5$, $b = 0.2$

The results are averaged over 30 runs by 10 dimensions and the average fitness function is reported in Table 9. Table 9, shows the comparison between GSA, GA, ICA and PSO with GBMO on multimodal and unimodal functions low-dimensional benchmark functions. The results show that GA, PSO, GBMO and GSA have some similar solutions.

The results are averaged over 30 runs by 20 dimensions and the average fitness function is reported in Table 10. In Table 10, illustrates GBMO provides better results than GA, ICA, GSA and PSO for all functions. These differences due to balance between the exploration and exploitation. The results are averaged over 30 runs by 30 dimensions and the average fitness function is reported in Table 11.

The performance of each algorithm is shown in Table 12, for two functions. The results show that GBMO has better solution than PSO except for functions F7. GBMO provides better results than GSA for all functions.

For optimization of high-dimensional functions, GBMO provide a suitable solution because molecules search the problem space with fast velocity, a randomized characteristic and a good balance between the exploration and the exploitations. In solving these problems GBMO has a good performance. The good convergence rate of GMO for Sphere function could be concluded from Fig. 9. According to this figure, GBMO tends to find the global optimum faster than other algorithms and hence has a higher convergence rate.

In Fig. 9, illustrates GBMO provides better results than GA, PSO and ICA for Sphere function. GBMO converges at the first few iterations and it can tune itself in the local optimum.

The good convergence rate of GMO for Sphere function could be concluded from Fig. 9. According to this figure, GBMO tends to find the global optimum faster than other algorithms and hence has a higher convergence rate.

Fig. 10, shows the GBMO has a good convergence rate for Ackley function. GBMO converges at the first iterations. For Ackley multimodal function the proposed algorithm has better performance in optima solution quality and in convergence speed rather than the ICA, PSO and GA algorithms.

As is shown in Fig. 11, for Rasrigin multi-modal function the GBMO algorithm has better performance rather than the GA algorithm and the proposed algorithm has been able to escape from the local peaks and reach to global optima at first iterations.

In Rosenbrock, uni-modal function, the performance of PSO and GBMO is better than ICA and GA algorithms but the GBMO proposed algorithm indicates better performance in convergence speed and in obtained optima solution quality in compare with the other algorithms. The performance of GBMO is shown in Fig. 12. PSO has better performance in convergence at the first iterations but PSO trap into local optima during the search. GBMO can avoid trapping in a local optimal by the Gases Brownian Motion and turbulent rotational motion in the last iterations. The figure shows that the exploitation ability of GBMO is better that the PSO algorithm. In Griewank, the performance of GBMO is better than PSO, ICA and GA algorithms that are shown in Fig. 13.

The average time for each algorithm is shown in Table 13. The run time of the GBMO algorithm is less than ICA, GSA and GA but a little more than PSO algorithm.

6.3. Solving real world problems

As it can be seen in Table 14, GBMO in problems with high dimensions, because of powerful exploration technique in the gases Brownian movement is more successful than the other compared algorithms. Although, with 9 and 21 dimensions in Tersoff (siC) and

Table 9
Minimization result of some functions. Maximum number of iterations = 1000. Results are averaged over 30 runs.

	GA	PSO	ICA	GSA	GBMO
F1	0.2499	1.1405×10^{-17}	6.2799×10^{-11}	3.7280×10^{-5}	4×10^{-9}
F2	1.0993×10^{-11}	0	0	1.2076×10^{-7}	7.9357×10^{-9}
F3	48.9673	0.0020	0.4466	0.9531	8.0096
F4	−0.0972	−0.0987	−0.0987	−0.0987	−0.0987
F5	0.1016	0.0298	1.9386×10^{-7}	4.9386×10^{-8}	1.7×10^{-11}

Table 10
Minimization result of some functions. Maximum number of iterations = 1000. Results are averaged over 30 runs.

	GA	PSO	ICA	GSA	GBMO
F1	3.2958	0.0143	0.0732	0.0194	8×10^{-9}
F2	0.6609	5.4021×10^{-4}	8.4028×10^{-4}	4.2267×10^{-5}	1.5871×10^{-9}
F3	1.3182×10^{3}	19.5173	1.9191×10^{3}	23.6711	18.3735
F4	−0.3373	−0.7114	−0.7206	−0.7117	−0.7225
F5	2.1664	0.2105	0.1205	6.8330×10^{-3}	2.5602×10^{-11}

Table 11
Minimization result of some functions. Maximum number of iterations = 1000. Results are averaged over 30 runs.

	GA	PSO	ICA	GSA	GBMO
F1	18.8398	9.0371	28.7678	0.2×10^{-8}	1.2×10^{-9}
F2	6.3023	0.0015	0.4128	0.5041	2.3807×10^{-8}
F3	9.8512×10^{3}	28.7533	1.0179×10^{6}	65.6541	28.0826
F4	−0.3702	−0.3687	−0.8319	−0.4550	−2.3712
F5	3.0695	1.0289	19.9587	1.1×10^{-2}	2.5602×10^{-5}

Table 12
Minimization result of some functions. Maximum number of iterations = 1000. Results are averaged over 30 runs.

	GA	PSO	ICA	GSA	GBMO
F7	2.2516×10^{-9}	0	7.8886×10^{-6}	0.0192	0.0085
F8	1.6432×10^{-10}	3.9162×10^{-51}	1.7×10^{-11}	8×10^{-19}	0

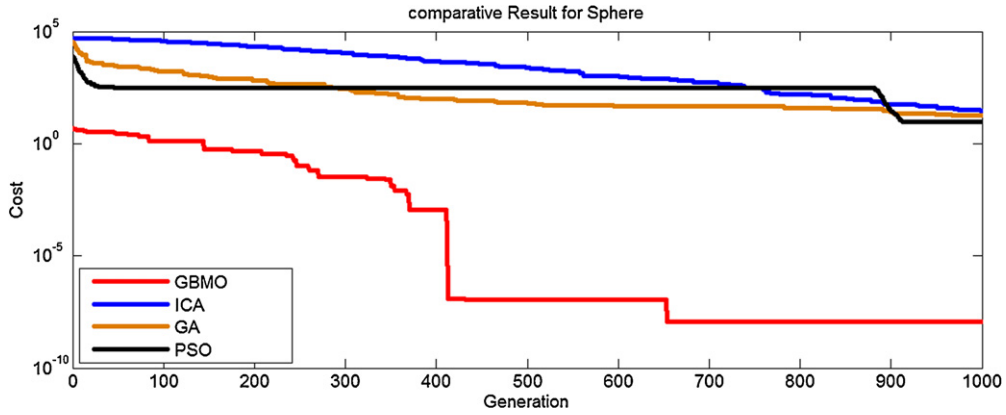

Fig. 9. Comparison of performance of GA, ICA, PSO and GBMO for minimization of Sphere function, with $n = 30$.

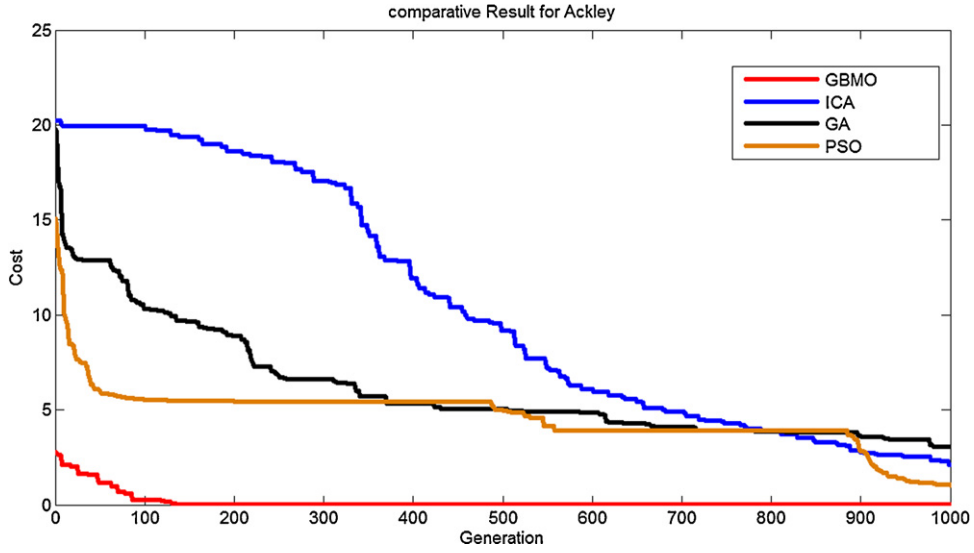

Fig. 10. Comparison of performance of GA, ICA, PSO and GBMO for minimization of Ackley function, with $n = 30$.

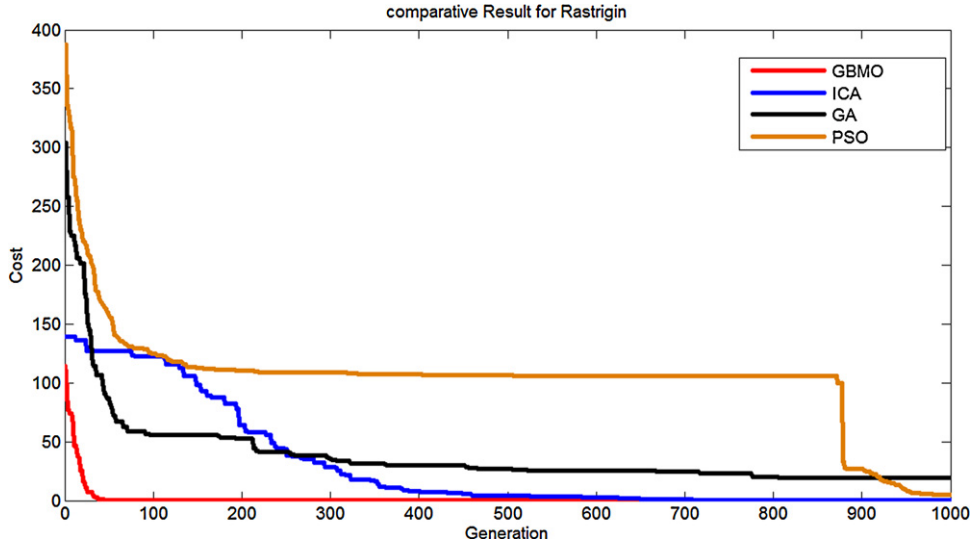

Fig. 11. Comparison of performance of GA, ICA, PSO and GBMO for minimization of Rastrigin function, with $n = 30$.

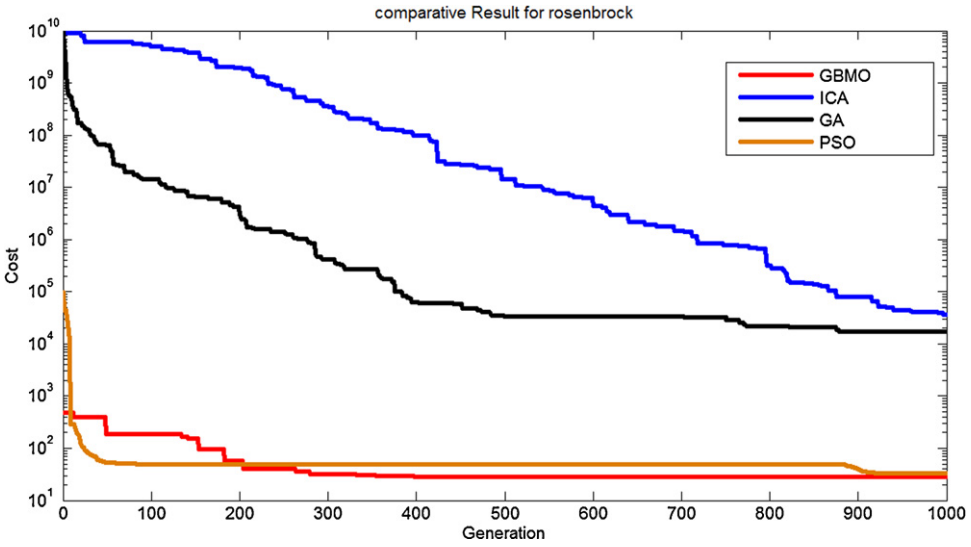

Fig. 12. Comparison of performance of GA, ICA, PSO and GBMO for minimization of Rosenbrock function, with $n = 30$.

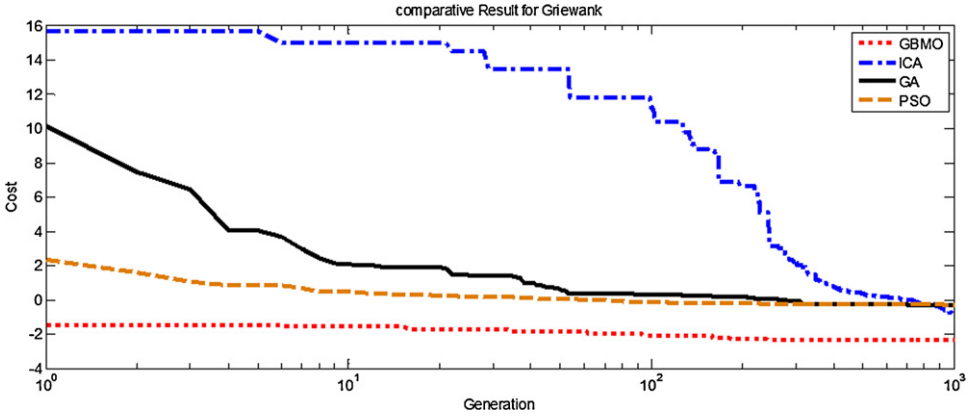

Fig. 13. Comparison of performance of GA, ICA, PSO and GBMO for minimization of Griewank function, with $n = 30$.

Table 13
Comparison of run time of GA, ICA, PSO, GSA and GBMO.

Algorithms	Run-time
ICA	430.5336 s
GSA	585.0367 s
GA	408.0434 s
PSO	304.2455 s
The proposed algorithm (GBMO)	310.0761 s

Table 14
Experimental results for Lennard–Jones and Tersoff potential function.

Algorithm	Dim	Tersoff (siB)	Tersoff (siC)	Lennard–Jonse
GBMO	30	−21.8210	−21.2945	−19.6809
	21	−19.2707	−17.6284	−14.2806
	9	−7.0706	−7.8548	−2.7129
GA	30	−12.8502	−12.4219	−8.5100
	21	−7.3005	−6.9519	−5.2422
	9	−4.9371	−4.1499	−1.0006
PSO	30	−18.5633	−18.4613	−17.0019
	21	−12.2625	−11.7322	−10.0791
	9	−6.5103	−6.8046	−2.6033
DE	30	−8.4939	−8.0400	−2.9371
	21	−4.9904	−4.4186	−1.7015
	9	−2.1482	−2.1107	−0.7294
ICA	30	−21.1570	−21.2741	−19.2087
	21	−18.7940	−18.3333	−15.4406
	9	−7.0659	−7.9240	−2.9614

Lennard–Jones, ICA has reached to better results. This is because of multi-swarm nature of this algorithm that makes it more powerful in tracking global optimums. In general, from the result obtain by GBMO this algorithm is relatively powerful and strong in global optimization tasks.

6.4. The computational complexity

The computational complexity is the study of classes of problems based on the rate of growth of space, time, or other fundamental unit of measure as a function of the size of the input.

An important aspect of understanding algorithms is to analyze time complexity. For a search algorithm this means either the average time until the optimum is founded or the time taken to guarantee that we have a solution that is a good approximation to the optimum.

Run time complexity analysis of the population-based stochastic search techniques like PSO, GA etc. is a critical issue by its own right. The average runtime of a population-based stochastic search algorithm usually depends on its stopping criterion.

The computational complexity of PSO based algorithm is $O(n)$ [64]. The computational complexity of PSO based algorithm is $O(n^2)$ [65]. For comparing the run-time complexity among some algorithms similar operations (initialization, fitness evaluation and termination) should be excluded because similar operation gives

Table 15
The computational complexity.

Algorithm	GA	PSO	ICA	GBMO
Time complexity	$O(n^2)$	$O(n)$	$O(n)$	$O(n)$

same run-time complexity. Also the number of generations, as it depends on the problem complexity and termination criteria (experimentally, GBMO has lower number of generations and it is fixed) should be excluded. Therefore, time-complexity calculations for the main loop of compared algorithms should be done. We consider the most time-consuming processes like velocity and position updates in GBMO algorithm. GBMO requires $O(n)$ time to update the speed and position of a molecule, then it takes $O(n)$ time to update them and compute their fitness values in each generation of the algorithm. Therefore, the time complexity of the algorithm is $O(n)$.

In general, The complexity of the GBMO is $O(n)$. Hence, the algorithm has linear time complexity. The time Complexity of PSO, GA, ICA and GBMO algorithms are shown in Table 15.

7. Conclusions

In this way, a novel evolutionary optimization algorithm inspired by the gases Brownian and turbulent rotational motion was utilized which is called Gases Brownian Motion (GBMO) algorithm. The efficiency of the proposed method has been compared with some well-known heuristic search methods as well. The obtained results confirm the high performance of the proposed method in solving various functions and SAT problems.

The applied PSO, GA, GSA, ICA and GBMO algorithm, and experimental results indicated that they are tractable for medium-size SAT problems. GBMO is more successful than the other compared algorithms for optimization problems with high dimensions because of powerful exploration technique in the gases Brownian movement. The results showed that the number of molecules increases, the quality of solutions is improved and the execution time rises. The run time of the GBMO algorithm is less than ICA, GSA and GA but a little more than PSO algorithm. GBMO algorithm has linear time complexity.

References

[1] H. Sarimveis, A. Nikolakopoulos, A line up evolutionary algorithm for solving nonlinear constrained optimization problems, Computers & Operations Research 32 (2005) 1499–1514.
[2] M. Melanie, An introduction to genetic algorithms, Massachusett's MIT Press 34 (1999) 1–9.
[3] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway: IEEE, 1995, pp. 1942–1948.
[4] L.A. Ingber, Simulated annealing: practice versus theory, Journal of Mathematics and Computer Modelling 18 (1993) 29–57.
[5] B. Franklin, M. Bergerman, Cultural algorithms: concepts and experiments, in: Proceedings of the IEEE Congress on Evolutionary Computation, La Jolla, CA, 2000, pp. 1245–1251.
[6] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transaction on System, Man and Cybernetics 26 (1996) 29–41.
[7] K. Lee, Z. Geem, A new structural optimization method based on the harmony search algorithm, Computers and Structures 82 (2004) 781–798.
[8] F.J. Von Zuben and L.N. De Castro, "Artificial immune systems: Part I – basic theory and applications," *School of Computing and Electrical Engineering*, State University of Campinas, Brazil, Technical Report DCA-RT 01/99, 1999.
[9] A. Kaveh, S. Talatahari, A novel heuristic optimization method: Charged system search, Acta Mechanica 213 (2010) 267–286.
[10] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359.
[11] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, A Gravitational Search Algorithm, Information Science, Special Section on High Order Fuzzy Sets 179 (2009) 2232–2248.
[12] E. Atashpaz-Gargari, C. Lucas, Imperialist Competitive Algorithm: an algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 2007, pp. 4661–4667.
[13] M. Abdechiri, M.R. Meybodi, H. Bahrami, A new algorithm: inspired of gases brownian motion, in: National Conference of Computer Society of Iran CSI2011, Tehran, Iran, 2011, Persian paper, pp. 171–176.
[14] J. Chuanwen, E. Bompard, A hybrid method of chaotic particle swarm optimization and linear interior for reactive power optimization, Mathematics and Computers in Simulation 68 (2005) 57–65.
[15] D.Y. Sha, C.Y. Hsu, A hybrid particle swarm optimization for job shop scheduling problem, Computers and Industrial Engineering 51 (2006) 791–808.
[16] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Applied Mathematics and Computation 188 (2007) 129–142.
[17] H. Liu, A. Abraham, W. Zhang, A fuzzy adaptive turbulent particle swarm optimization, International Journal of Innovative Computing and Applications 1 (2007) 39–47.
[18] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Applied Mathematics and Computation 186 (2007) 1407–1422.
[19] H.W. Ge, L. Sun, Y.C. Liang, F. Qian, An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling, IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans 38 (2008) 358–368.
[20] A. Kavehand, S. Talatahari, Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, Computer Structure 87 (2009) 267–283.
[21] H.C. Huang, C.M. Chu, J.S. Pan, The optimized copyright protection system with genetic watermarking, Journal Soft Computing 13 (2009) 333–343.
[22] M.R. Chen, X. Li, X. Zhang, Y.Z. Lu, A novel particle swarm optimizer hybridized with extremal optimization, Applied Soft Computing 10 (2010) 367–373.
[23] R.F. Abdel-Kader, Genetically improved PSO algorithm for efficient data clustering, in: Proceedings of International Conference on Machine Learning and Computing, Bangalore, 2010, pp. 71–75.
[24] H. Al-Qaheri, Digital watermarking using ant colony optimization in fractional Fourier domain, Journal of Information Hiding and Multimedia Signal Processing 1 (2010) 292–300.
[25] C.H. Hsu, W.J. Shyr, K.H. Kuo, Optimizing multiple interference cancellations of linear phase array based on particle swarm optimization, Journal of Information Hiding and Multimedia Signal Processing 1 (4) (2010).
[26] F.C. Chang, H.C. Huang, A refactoring method for cache-efficient swarm-intelligence algorithms, Information Sciences (2010) 11–12 (in press).
[27] A.B. Hashemi, M.R. Meybodi, A note on the learning automata based algorithms for adaptive parameter selection in PSO, Applied Soft Computing 11 (2011) 689–705.
[28] R. Thangaraj, M. Pant, A. Abraham, P. Bouvry, Particle swarm optimization: hybridization perspectives and experimental illustrations, Applied Mathematics and Computation 217 (2011) 5208–5226.
[29] C.H. Holbrow, J.N. Lloyd, J.C. Amato, E. Galvez, M. Elizabeth Parks, Modern Introductory Physics, Springer-Verlag, New York, 2010, pp. 536–542.
[30] C.A.M. Vassallo, Van der Waals and Ideal Gas Models for Compressibility by Means of Pressure in pneumatic pipes from 1 to 100 Lpm, Electro Electronica 22 (2004) 7–11.
[31] J. Hirschfelder, Molecular Theory of Gases and Liquids, John Wiley and Sons Inc, Wisconsin, 1966, pp. 250–262.
[32] G.E. Uhlenbeck, L.S. Ornstein, On the theory of Brownian motion, Physical Review 36 (1930) 823–841.
[33] L.S. Ornstein, On the Brownian motion, Proceedings of Royal Academy of Amsterdam 21 (1917) 96–108.
[34] J.C. Kotz, P. Treichel, J.R. Townsend, Chemistry and Chemical Reactivity, Cengage Learning 2 (2009).
[35] A. Mittal, Objective Chemistry for Lit Entrance, New Age International, 2002.
[36] L.S. Ornstein, H.C. Burger, On the theory of the Brownian motion, Proceedings of Royal Academy of Amsterdam 21 (1919) 922–931.
[37] S. Chandrasekhar, Stochastic problems in physics and astronomy, Reviews of Modern Physics 15 (1943) 1–89.
[38] S. Sharma, Atomic and Nuclear Physics, Pearson Education India, 2008.
[39] M. Mansfield, C. OSullivan, *Understanding Physics*, 2nd Ed., 2011.
[40] R. Resnick, J. Walker, *Fundamentals of Physics*, 7th Sub-ed., 2004.
[41] M. Zheng, Kneading plane of the circle map, Chaos, Solitons and Fractals 4 (1994).
[42] S.A. Cook, The complexity of theorem proving procedures, in: Proceedings of the 3rd Annual ACM Symposium on the Theory of Computation, New York, USA, 1971, pp. 151–158.
[43] J. Gu, Global optimization for satisfiability (SAT) problem, IEEE Transactions on Knowledge and Data Engineering 6 (1994) 361–381.
[44] J. Balicki, A. Stateczny, B. Zak, Genetic algorithms and Hopfield neural networks to solve combinatorial optimization problems, Applied Mathematics and Computer Science 10 (1997) 568–592.
[45] M. Nagamatu, T. Yanaru, On the Stability of Lagrange Programming Neural Networks for Satisfiability Problems of Propositional Calculus, Neurocomputing 13 (1992) 440–446.
[46] X. Zhao, Y. Wang, L.S. Thakur, Lagrangian relaxation neural networks for job shop scheduling, IEEE Transactions on Robotics and Automation 16 (2000) 78–88.

[47] J.K. Hao, R. Dorne, A new population-based method for satisfiability problems, in: Proceedings of the 1st European Conference on Artificial Intelligence, 1994, pp. 135–139.
[48] W.M. Spears, Simulated annealing for hard satisfiability problems, in: In second DIMACS Implementation Challenge: Cliques, Coloring an satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 26, 1996, pp. 533–558.
[49] M. Bertran, S. Lakhdar, G. Eric, Tabu search for SAT, in: In Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative application of Artificial Intelligence Conference (AAAI-97/IAAI-97), 1997, pp. 281–285.
[50] J. Hao, F. Lardeux, F. Saubion, A hybrid genetic algorithm for the satisfiability problem, in: Proceeding of the 1st International Workshop on Heuristics, Cambridge, MA, USA, 2002.
[51] G. Folino, C. Pizzuti, G. Spezzano, Combining cellular genetic algorithms and local search for solving satisfiability problems, in: Proceedings of ICTAI'98 10th IEEE International Conference Tools with Artificial Intelligence, IEEE Computer Society, 1998, pp. 192–198.
[52] P. Hansen, S. Perron, Brun Yuriy, Merging the local and global approaches to probabilistic satisfiability, International Journal of Approximate Reasoning 47 (2008) 125–140.
[53] M. Bachir, M. Batouche, Solving the maximum satisfiability problem using an evolutionary local search algorithm, The International Arab Journal of Information Technology 2 (2005) 154–161.
[54] E. Alba, B. Dorronsoro, H. Alfonso, Cellular memetic algorithms, Journal of Computer Science & Technology 5 (2005).
[55] S. Das, P.N. Suganthan, 2010. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems, Technical Report, 4–12.
[56] M.R. Hoare, Structure and dynamics of simple microclusters, Advances in Chemical Physics 40 (1979) 49–135.
[57] N.P. Moloi, M.M. Ali, An iterative global optimization algorithm for potential energy minimization, Journal of Computational Optimization and Applications 30 (2005) 119–132.
[58] G.L. Xue, R.S. Maier, J.B. Rosen, Minimizing the Lennard–Jones potential function on amassively parallel computer, in: Proceedings of the 1992 ACM International Conference on Supercomputing, 1991.
[59] J. Tersoff, Empirical inter atomic potential for silicon with improved elastic properties, Physics Review B 38 (1988) 9902–9905.
[60] F. Lardeux, F. Saubion, J. Hao, GASAT: a genetic local search algorithm for the satisfiability problem, Evolutionary Computation 14 (2006) 223–253.
[61] L. Yingbiao, A genetic algorithm based on adapting clause weights, Chinese Journal of Computer 7 (2005) 20–32.
[62] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (1999) 82–102.
[63] R.L. Haupt, E. Haupt, Practical Genetic Algorithms, 2nd ed., John Wiley & Sons, 2004.
[64] M. Giacobini, A. Brabazon, S. Cagnoni, G.A.D. Caro, A. Ekárt, A.I. Esparcia-Alcázar, M. Farooq, A. Fink, P. Machado, Applications of Evolutionary Computing, Springer, 2009, 5484.
[65] S. Ranka, Contemporary computing, in: Third International Conference, IC3 2010, Noida, Springer, India, 2010, pp. 1–13.