Information Sciences 275 (2014) 127–144

Contents lists available at ScienceDirect

Information Sciences

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

KGMO: A swarm optimization algorithm based on the kinetic energy of gas molecules

Sara Moein [a,*], Rajasvaran Logeswaran [a,b]

[a] Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Malaysia
[b] School of Engineering, Science & Technology, Nilai University, 71800 Nilai, Malaysia

ARTICLE INFO

Article history:
Received 18 July 2011
Received in revised form 31 December 2013
Accepted 9 February 2014
Available online 18 February 2014

Keywords:
Kinetic Gas Molecule Optimization (KGMO)
Kinetic energy
Particle Swarm Optimization (PSO)
Gravitational Search Algorithm (GSA)
Optimal convergence

ABSTRACT

Swarm-based algorithms have acquired an important role in solving real-world optimization problems. In this paper, Kinetic Gas Molecule Optimization (KGMO), an optimization algorithm that is based on the kinetic energy of gas molecules, is introduced. The agents are gas molecules that are moving in the search space; they are subject to the kinetic theory of gases, which defines the rules for gas molecule interactions in the model. The performance of the proposed algorithm, in terms of its ability to find the global minima of 23 nonlinear benchmark functions, is evaluated against the corresponding results of two well-known benchmark algorithms, namely, Particle Swarm Optimization (PSO) and the recently developed high-performance Gravitational Search Algorithm (GSA). The simulations that were undertaken indicate that KGMO achieves better results in decreasing the Mean Square Error (MSE). Significant improvements of up to 10^7 and 10^{20} times were achieved by KGMO against PSO and GSA, respectively, in solving unimodal benchmark functions within 150 iterations. Improvements of at least tenfold were achieved in solving the multimodal benchmark functions. The proposed algorithm is more accurate and converges faster than does the benchmark algorithms, which makes this algorithm especially useful in solving complex optimization problems.

1. Introduction

Algorithms that are inspired by natural phenomena have taken center stage in the development of new computer solutions [5,21,25,26,31–33]. These algorithms have been determined to be highly efficient in solving complex computational problems, such as optimizing functions, classification, control objectives, image processing and filter modeling [10–13,16].

There have been numerous studies seeking to find solutions for optimization problems [8–10,12,14,24,27,28,36]. However, classical optimization algorithms, such as Brute-Force Search, Breath-First Search, Uniform Cost Search and Depth First Search [19], do not provide suitable solutions to complex problems in pattern recognition, the optimization of objective functions and image processing, which are associated with high-dimensional search spaces [34]. In addition, techniques, such as exhaustive search, are not practical solutions to such problems [1–4].

Two commonly recognized aspects in the population-based heuristic algorithms are exploration (the ability to expand the search space) and exploitation (the ability to find the optimum around a good solution) [24]. Exploring the search space to find new solutions occurs in the first few iterations of a heuristic search algorithm, where the algorithm must avoid becom-

* Corresponding author.
E-mail address: Sara.moein08@mmu.edu.my (S. Moein).

http://dx.doi.org/10.1016/j.ins.2014.02.026
0020-0255/

ing trapped in a local optimum. Having a suitable tradeoff between exploration and exploitation is necessary, to converge the agents on the most optimum solution. Too much focus on exploration would result in a purely random search, whereas too much focus on exploitation would result in becoming trapped in a local search [6,24].

There has been a large body of work in the area of swarm intelligence for optimization and for solving different problems. Ant Colony Optimization (ACO) [9] is one such algorithm; ACO is modeled based on the indirect communication between ants by means of chemical pheromone trails, which enables them to find short paths between their nest and food sources. Other popular heuristic algorithms include the genetic algorithm (GA) which is inspired by Darwinian evolutionary theory [28], Artificial Immune System (AIS), which simulates the body's immune system [12], and Particle Swarm Optimization (PSO), which is based on simulation of the swarm behavior of a flock of birds [10]. Zne and Chou [36] proposed a hybrid search algorithm that combines the advantages of GA and ACO and that can explore the search space and exploit the best solutions. Dorigo et al. [9] introduced the Ant System (AS), which is an analogy of ACO that capitalizes on positive feedback, cooperation by transferring information and the use of a constructive greedy heuristic in which agents compete to survive. Dong et al. [8] modeled the social foraging behavior of Escherichia coli bacteria to solve optimization problems. They proposed a hybrid approach that involves GA and bacterial foraging (BF) algorithms for solving function optimization problems. Rashedi et al. [24] proposed the Gravitational Search Algorithm (GSA), which is based on the law of gravity. In their algorithm, the agents are a collection of masses that interact with each other based on Newtonian gravity and the laws of motion. In yet another study, Formato [14] introduced the Central Force Optimization (CFO), which is a new deterministic multi-dimensional search metaheuristic that is based on the metaphor of gravitational kinematics.

Overall, the review of the existing literature reveals that there is no single superior method for solving optimization problems. This concept is proven by the "no free lunch theorem for optimization," which states that no one algorithm can solve all of the optimization problems, but each algorithm can solve a special class of problems [35]. Although many of the algorithms that have been developed to solve optimization problems do achieve good performance, there are still shortcomings. For example, the standard PSO algorithm often gets trapped in local optima when solving complex multimodal problems [7,17]. GA provides no absolute assurance of finding a global optimum, and representation of the problem for GA is often difficult. Furthermore, there are various operations, such as mutation and crossover, in GAs; as a result, GAs require a long response time to find the solution for some problems [28]. Evolutionary algorithms (EA) also suffer from slow convergence [7].

The aim of this work is to develop an optimization algorithm that overcomes the above shortcomings. This algorithm must be able to achieve a solution that has the least (or at most, a small) error compared with the globally optimum solution within a minimal number of iterations, thus offering an improvement in terms of accuracy, convergence time and simplicity of operations. The metaheuristic optimization algorithm that is proposed in this work, namely, the Kinetic Gas Molecule Optimization (KGMO), is based on the behavior of gas molecules. The performance of the proposed algorithm is evaluated against two well-known benchmark optimization algorithms, specifically, PSO and GSA. It will be shown that the gas molecule behavior model can achieve more accurate results with a smaller Mean Square Error (MSE) and in a smaller number of iterations, compared with the benchmark algorithms.

Section 2 introduces the fundamentals of the kinetic theory of gas on which the proposed model is based. Next, in Section 3, KGMO and its characteristics are described. A comparative study is presented in Section 4, with a demonstration of the experimental results being given in Section 5 and a discussion being presented in Section 6. Finally, the concluding remarks are presented in Section 7.

2. Kinetic theory of gas molecules

This section provides a brief description of the relevant basic concepts of gas molecule laws that form the underlying principle of the proposed algorithm. Such scientists as Boyle, Charles and Gay-Lussac developed gas laws that are based on empirical observations to describe the macroscopic behavior of gas molecules; they described the properties that can be directly observed and experienced by a person [20]. The atomic theory of gases states that each substance is composed of a large number of very small particles (molecules or atoms). Basically, all of the properties of the gases, including the pressure, volume and temperature, are the consequence of the actions of the molecules that compose the gas [16].

There are 5 postulates that describe the behavior of molecules in a gas. The kinetic molecular theory of ideal gases is given as follows [20]:

1. A gas consists of a collection of small particles (molecules) that travel in straight-line motion. The movement is based on Newton's Laws.
2. The molecules in a gas occupy no volume. They are points.
3. The collisions between molecules are perfectly elastic. No energy is gained or lost during a collision.
4. There are no attractive or repulsive forces between the molecules.
5. The average kinetic energy of a molecule is $3kT/2$, where T is the absolute temperature, and k is the Boltzmann constant, which has the value of 1.38×10^{-23}.

The ideal gas rule is given by [20]:

$$PV = NkT \tag{1}$$

where P is the pressure that is exerted by the gas, V is the volume of the container, and N is the number of particles in the gas.

The kinetic energy equation is derived as follows:

$$\Delta k = w = F\Delta s = ma\Delta s \tag{2}$$

where Δk is the difference of the gas molecule's kinetic energy between the old and new positions, w is the work energy expended, F is the Newton force applied, Δs is the difference in the positions of the gas molecule in a unit time interval, and m is the mass of the gas molecule.

Then, from the kinematics equations via a rearrangement, we have:

$$v^2 = v_0^2 + 2a\Delta s \Rightarrow a\Delta s = \frac{v^2 - v_0^2}{2} \tag{3}$$

where v is the velocity of the gas molecule in the new position, v_0 is the velocity of the gas molecule in the old position, and a is the acceleration of the gas molecule.

Combining the two expressions, we obtain:

$$\Delta k = m\left(\frac{v^2 - v_0^2}{2}\right) \tag{4}$$

and expanding the equation, we obtain:

$$\Delta k = \frac{1}{2}mv^2 - \frac{1}{2}mv_0^2 \tag{5}$$

Naturally, the kinetic energy of an object at rest should be zero. Thus, an object's kinetic energy is expressed by

$$k = \frac{1}{2}mv^2 \tag{6}$$

To illustrate the theory behind the proposed algorithm, consider a closed vertical cylinder within which the pressured air supports the weight of the cover (piston) on top [16], as given in Fig. 1. Based on the rule of ideal gases (Eq. (1)), with constant pressure and decreasing temperature, the gas molecules converge together in the part of the container that has a lower temperature (Eq. (6)) because the lower temperature will cause less motion. This rule is illustrated in Fig. 1, where the lower left part of the container has the lowest temperature, and the gas molecules converge there over several time intervals. This principle and Eq. (6) are used in the proposed optimization algorithm, which is described in the next section.

3. KGMO – The proposed algorithm

In the proposed Kinetic Gas Molecule Optimization (KGMO) algorithm, the gas molecules are the agents in the search space, and kinetic energy is used in measuring the performance. The gas molecules move in the container until they converge

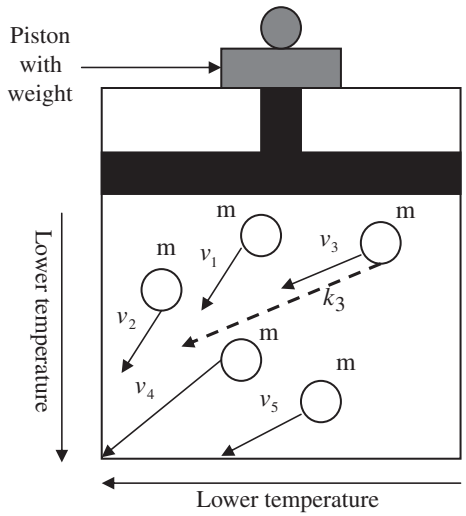

Fig. 1. Under constant pressure, the kinetic energy of gas molecules decreases by decreasing the velocity.

in the part of the container that has the lowest temperature and kinetic energy. It is known that gas molecules attract each other based on weak electrical intermolecular Van Der Waal forces, where the electrical force is the result of positive and negative charges in the molecules [22]. In the KGMO, each gas molecule (agent) has four specifications: position, kinetic energy, velocity and mass. The kinetic energy of each gas molecule determines its velocity and position. In the algorithm, the gas molecules explore the whole search space to reach the point that has the lowest temperature.

Next, consider a system that has N agents (gas molecules). The position of the ith agent is defined by

$$X_i = (x_i^1, \ldots, x_i^d, \ldots x_i^n), \quad \text{for } (i = 1, 2, \ldots, N) \tag{7}$$

where x_i^d represents the position of the ith agent in the dth dimension.

The velocity of the ith agent is presented by

$$V_i = (v_i^1, \ldots v_i^d, \ldots, v_i^n), \quad \text{for } (i = 1, 2, \ldots, N) \tag{8}$$

where v_i^d represents the velocity of the ith agent in the dth dimension.

The movement of the gas molecules in the cylinder is based on the Boltzmann distribution [20], which means that its velocity is proportional to the exponential of the molecules' kinetic energy. This kinetic energy, in turn, is defined as

$$k_i^d(t) = \frac{3}{2} NbT_i^d(t), K_i = (k_i^1, \ldots k_i^d, \ldots, k_i^n), \quad \text{for } (i = 1, 2, \ldots, N) \tag{9}$$

where N is the number of gas molecules, b is the Boltzmann constant, and $T_i^d(t)$ is the temperature of the ith agent in the dth dimension at time t.

The velocity of the molecule is updated by

$$v_i^d(t+1) = T_i^d(t) w v_i^d(t) + C_1 rand_i(t)(gbest^d - x_i^d(t)) + C_2 rand_i(t)(pbest_i^d(t) - x_i^d(t)) \tag{10}$$

where $T_i^d(t)$ for the converging molecules reduces exponentially over time and is calculated as

$$T_i^d(t) = 0.95 \times T_i^d(t-1) \tag{11}$$

The vector $pbest_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^n)$ represents the best previous position of the ith gas molecule, and $gbest = (gbest^1, gbest^2, \ldots, gbest^n)$ is the best previous position among all of the molecules in the container. The velocity and the position of each particle are initialized by random vectors within the corresponding ranges. Here, $[-v_{\min}, v_{\max}]$ is used as the limits of the gas molecules' velocity. If $|v_i| \succ v_{\max}$, then $|v_i| = v_{\max}$. w is the inertia weight that reflects the gas molecule's resistance to slow its movement. Additionally, $rand_i(t)$ is a uniform random variable in the interval [0,1] at time t, which is used to provide a randomized characteristic to the search algorithm. C_1 and C_2 are two acceleration constants.

The mass m of each gas molecule is a random number within the range $0 \prec m \leqslant 1$; once identified, it remains constant throughout the execution of the algorithm because the container is assumed to contain only one type of gas at any one time. The random number is used to simulate different types of gases in different executions of the algorithm.

From the equations of motion in physics, the position of the molecule is defined by [15]

$$x_{t+1}^i = \frac{1}{2} a_i^d(t+1)t^2 + v_i^d(t+1)t + x_i^d(t) \tag{12}$$

where a_i^d represents the acceleration of the ith agent in the dth dimension.

From the acceleration equation, we obtain

$$a_d^i = \frac{(dv_i^d)}{dt} \tag{13}$$

On the other hand, from Eq. (6) of the gas molecule laws, we have

$$dk_d^i = \frac{1}{2} m (dv_i^d)^2 \Rightarrow dv_i^d = \sqrt{\frac{2(dk_i^d)}{m}} \tag{14}$$

Therefore, from Eqs. (13) and (14), the acceleration is defined as

$$a_d^i = \frac{\sqrt{\frac{2(dk_i^d)}{m}}}{dt} \tag{15}$$

In the time interval Δt, Eq. (15) can be re-written as

$$a_d^i = \frac{\sqrt{\frac{2(\Delta k_i^d)}{m}}}{\Delta t} \tag{16}$$

Thus, in a unit time interval, the acceleration would be

$$a_d^i = \sqrt{\frac{2(dk_i^d)}{m}} \tag{17}$$

Then, from Eqs. (12) and (17), the position of the molecule is calculated by

$$\begin{aligned} &x_{t+1}^i = \tfrac{1}{2} a_i^d(t+1)\Delta t^2 + v_i^d(t+1)\Delta t + x_i^d(t) \Rightarrow \\ &x_{t+1}^i = \tfrac{1}{2}\sqrt{\tfrac{2(\Delta k_i^d)}{m}}(t+1)\Delta t^2 + v_i^d(t+1)\Delta t + x_i^d(t) \end{aligned} \tag{18}$$

Finally, considering that the molecule mass (m) is a random number in each execution of the algorithm but the same for all of the molecules in an execution, for simplicity, the position is updated for the unit time interval by

$$x_{t+1}^i = \sqrt{\frac{2(\Delta k_i^d)}{m}}(t+1) + v_i^d(t+1) + x_i^d(t) \tag{19}$$

The minimum fitness function is found by using

$$\left.\begin{aligned} pbest_i &= f(x_i), \quad \text{if} \quad f(x_i) \prec f(pbest_i) \\ gbest &= f(x_i), \quad \text{if} \quad f(x_i) \prec f(gbest) \end{aligned}\right\} \tag{20}$$

Each gas molecule attempts to modify its position (x_i^d) by using the distance between the current position and $pbest_i^d$ and the distance between the current position and $gbest_i$. The general steps in the KGMO algorithm are summarized by the pseudo-code below:

```
For each gas molecule {
  Repeat initialize gas molecules until it satisfies all constraints
}
Do {
  For each particle {
    Calculate fitness value
    If the calculated fitness value is better than the existing best fitness value (pbest) OR no existing pbest {
      Set current value as the new pbest
    }
  }
  For each particle {
    If particle's pbest is better than the global best fitness value (gbest) so far OR no existing gbest {
      Set current particle's pbest as new gbest
    }
    Calculate the kinetic energy of each gas molecule (Eq. (9))
    Update particle velocity (Eq. (10)) and position (Eq. (19))
  }
} while maximum iterations or minimum error criterion not attained
```

4. Comparative study

To examine how the proposed KGMO is situated compared to other heuristic search optimization algorithms, an experimental comparison with other algorithms was undertaken. Based on the recent results in [24], GSA is reported to perform the best in optima identification for most of the benchmark functions in comparison with other new swarm methods such as the Real Genetic Algorithm (RGA) and Central Force Optimization (CFO) algorithms. The results reported were achieved in 1000 training cycles to minimize the error. On the other hand, PSO has long been used as the standard benchmark algorithm to evaluate the performance of various new optimization algorithms [6,7,29,32–35]. Therefore, in this work, both GSA and PSO were selected as the benchmark algorithms against which we evaluate the performance of the proposed KGMO. The fundamental characteristics of the PSO and GSA are described below.

4.1. PSO algorithm

PSO is a population-based stochastic optimization technique that was developed by Eberhart and Kennedy (1995) [18], which was inspired by the social behavior of bird flocking or fish schooling [11]. PSO updates the population of particles by updating the following equations:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{21}$$

and

$$v_i^d(t+1) = w(t)v_i^d(t) + C_1 rand_i(t)(pbest_i^d - x_i^d(t)) + C_2 rand_i(t)(gbest^d - x_i^d(t)) \tag{22}$$

where $pbest_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^n)$ and $gbest = (gbest^1, gbest^2, \ldots, gbest^n)$ represent the best previous position of the ith particle and the best previous position among all of the particles in the population, respectively. Here, $rand_i(t)$ is a uniform random variable within the interval $[0,1]$ at time t. C_1 and C_2 are two acceleration constants [29]. The initialization process for the velocities and positions of the particles is conducted by using vectors of random numbers that are in the corresponding range, given in [35].

4.1.1. KGMO vs. PSO

In both the proposed KGMO and the benchmark PSO, optimization is obtained by the movement of agents in the search space. However, the movement strategy that is used by both algorithms is based on different laws. Some important differences between the two algorithms are the following:

- KGMO uses kinetic energy as the update rule, and this use is the main feature.
- The distance measure in KGMO is updated with a rule that uses Eq. (19), which involves acceleration; this feature is absent in the movement equation of the PSO. Incorporating acceleration allows improvement in the search speed.
- The temperature of the gas molecules in considered when updating the velocity in KGMO, whereas the PSO does not account for the effects of temperature.
- The temperature $T(t)$ in the KGMO works as a controlling parameter, which is a feature that is absent in the PSO. It allows faster controlled optimization in the KGMO.
- The mass variable m in the KGMO is a random number that is absent in the PSO.
- Based on the behavior of gas molecules, the particles in a KGMO move very fast, which allows for quick exploration of the search space to overcome trapping in local minima.
- The overall search strategies of both algorithms are different. PSO simulates the social behavior of birds, while KGMO simulates gas law theory.

4.2. GSA algorithm

GSA is a population-based search algorithm that is based on the law of gravity and mass interaction [24]. Objects tend to move toward the heavier mass based on the gravitational attraction force between them. There are four specific parameters in the GSA: the position of the mass in the dth dimension, inertia mass, active gravitational mass and passive gravitational mass.

GSA determines the positions of the mass at a specified dimension, with the determined positions reflecting the solution of the problem. It has a randomized initialization process. The inertia parameter represents the resistance of the mass of agents in slowing their movement. The gravitational mass and inertia mass are computed by a fitness evolution of the problem. At each iteration, the positions of the masses are updated. A fixed number of iterations defines the termination condition of the algorithm. After termination, the position of the mass at specified dimensions of the corresponding agent becomes the global fitness for a particular problem [24]. To describe the GSA, consider a system with s masses in which the position of the ith mass is defined as follows:

$$X_i = (x_i^1, \ldots, x_i^d, \ldots x_i^n), \quad \text{for } (i = 1, 2, \ldots, N) \tag{23}$$

where x_i^d represents the position of the ith agent in the dth dimension.

The mass of each agent is calculated after computing the current population's fitness, as given below:

$$q_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{24}$$

Hence,

$$M_i(t) = \frac{q_i(t)}{\sum_{j=1}^{n} q_j(t)} \tag{25}$$

where $fit_i(t)$ and $M_i(t)$ represent the fitness value and mass of the agent i at time t, respectively. For a minimization problem, $worst(t)$ and $best(t)$ are defined as follows:

$$best(t) = \min_{j \in \{1,\ldots,N\}} fit_j(t) \tag{26}$$

and

$$worst(t) = \max_{j \in \{1,\ldots,N\}} fit_j(t) \tag{27}$$

To compute the acceleration of an agent, the total forces from a set of heavier masses applied on an agent should be considered based on the law of gravity (Eq. (28)), which is followed by a calculation of the agent acceleration by using the law of motion (Eq. (29)), as given below:

$$F_i^d(t) = \sum_{j \in kbest, j \neq i} rand_j G(t) \frac{M_j(t) M_i(t)}{R_{ij(t)} + \varepsilon} (x_j^d(t) - x_i^d(t)) \tag{28}$$

and

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = \sum_{j \in kbest, j \neq i} rand_j G(t) \frac{M_j(t)}{R_{ij(t)} + \varepsilon} \left(x_j^d(t) - x_i^d(t) \right) \tag{29}$$

The velocity and position of an agent are updated according to Eq. (30) and Eq. (31), respectively:

$$v_i^d(t+1) = rand_i v_i^d(t) + a_i^d(t) \tag{30}$$

and

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{31}$$

where $rand_i$ and $rand_j$ are two uniformly distributed random numbers in the interval $[0,1]$; ε is a small constant; $R_{ij}(t)$ is the Euclidean distance between two agents i and j, defined as $||X_i(t), X_j(t)||_2$; $kbest$ is the set of first K agents with the best fitness value and largest mass, which is a function of time and is initialized to $K0$ at the beginning and decreases with time. $K0$ is set to N (the total number of agents) and is decreased linearly to 1.

Comparatively, the proposed KGMO algorithm is significantly simpler to implement than the GSA. It incurs less mathematical complexity and a reduced number of equations. These features provide a significant advantage in terms of the processing cost.

5. Experimental results

To fairly and thoroughly evaluate the performance of the developed KGMO algorithm, a comprehensive set of 23 standard benchmark functions, as given in [33], is used. These benchmark functions are presented below, followed by the performance comparison results of KGMO with PSO and GSA.

5.1. Benchmark functions

The standard benchmark functions used, as obtained from [33], are shown in Table 1. These functions, grouped into sets, were designed to test various aspects of algorithms.

- The first set of $f1(x)$ to $f7(x)$ consists of unimodal functions.
- The second set of $f8(x)$ to $f13(x)$ is multimodal high-dimensional functions.
- The third set, given by $f14(x)$ to $f23(x)$, consists of multimodal test functions with fixed dimensions.

The detailed description of all of the test functions is given by Tables A1–A7 in Appendix A.

5.2. Comparison of results

The results of the performance evaluation, as achieved by the three algorithms for the three types of functions, unimodal, multimodal high dimensional and multimodal test functions with fixed dimensions, are described in this section. The result achieved for each function is the average of the best global minimum obtained by each algorithm over 50 runs. In all cases, the population size (N) was set to 50, and the maximum number of iterations was 150.

For the PSO, we set $C_1 = C_2 = 2$, and the inertia factor (w) was decreased linearly from 0.85 to 0.2, which were the default values. In the GSA, G_0 was set to 100, α was set to 20, $K0 = N = 50$ and T = the total number of iterations. In the KGMO, $C_1 = 1$, $C_2 = 3$, $0 < m \leqslant 1$ in each execution (the same for all of the molecules), the inertia factor (w) was decreased linearly from 0.85 to 0.2, and T was decreased exponentially from 0.95 to 0.1.

In [24], it was found that GSA already provides good results in finding the global minimum of the 23 benchmark functions at 1000 iterations. The reader is reminded that the aim of this work was to formulate an improved algorithm that can converge toward the optima in a reduced number of training cycles.

5.2.1. Unimodal high-dimensional test functions

The functions $f1(x)$ to $f7(x)$ are unimodal functions for which the focus is on the convergence rate because current optimization algorithms are already able to present global optima that are close to the actual optima. Therefore, the aim is to obtain the best global minimum in the least number of required iterations [23].

Table 1
Benchmark functions.

Test function	n	s	$f_{\min}$
$f1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0
$f2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	0
$f3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^n$	0
$f4(x)=\max_i\{\lvert x_i\rvert, 1\leqslant i\leqslant n\}$	30	$[-100,100]^n$	0
$f5(x)=\sum_{i=1}^{n-1}[(100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-30,30]^n$	0
$f6(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	30	$[-100,100]^n$	0
$f7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	$[-1.28,1.28]^n$	0
$f8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	$[-500,500]^n$	−12569.5
$f9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12,5.12]^n$	0
$f10(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+e$	30	$[-32,32]^n$	0
$f11(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]^n$	0
$f12(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right]+\sum_{i=1}^{n}u(x_i,10,100,4)\Big\}$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i\prec -a\end{cases}$	30	$[-50,50]^n$	0
$f13(x)=0.1\left\{\begin{array}{l}\sin^2(3\pi x_1)+\\ \sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+\\ (x_n-1)^2[1+\sin^2(2\pi x_n)]\end{array}\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	0
$f_{14}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	$[-65.53,65.53]$	1
$f15(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]$	0.00030
$f16(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2+4x_2^2+4x_2^4$	2	$[-5,5]$	−1.03162
$f17(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$	2	$[-5,10]\times[0,15]$	0.398
$f18(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-5,5]$	3
$f19(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	3	$[0,1]$	−3.86
$f20(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	6	$[0,1]$	−3.32
$f21(x)=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0,10]$	−10
$f22(x)=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0,10]$	−10
$f23(x)=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0,10]$	−10

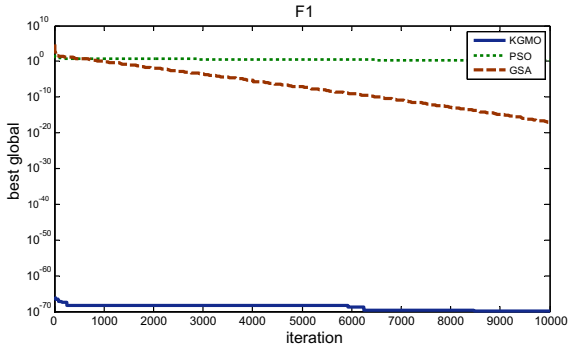

Fig. 2. *Average Gbest* for minimization of $f1(x)$ over 50 runs, each with 10,000 iterations.

Figs. 2–4 present the convergence results that are achieved by the KGMO, GSA and PSO for the functions $f1(x)$, $f2(x)$ and $f3(x)$, respectively, in 10,000 iterations. The figures show that the KGMO can converge toward the global minima extremely quickly, in less than 150 iterations, with the identified optima being very close to the actual minimum value. In contrast, the PSO and GSA were unable to obtain the KGMO result even after 10,000 iterations. With an eye toward time efficiency and this study's objective of achieving quick convergence, the maximum number of iterations for all of the tests and experiments in the remainder of this work is set at 150.

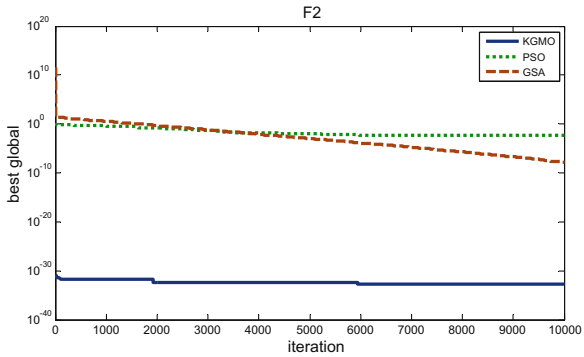

Fig. 3. *Average Gbest* for the minimization of $f2(x)$ over 50 runs, each with 10,000 iterations.

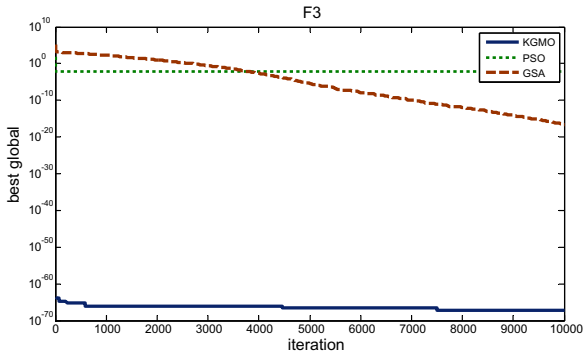

Fig. 4. *Average Gbest* for the minimization of $f3(x)$ over 50 runs, each with 10,000 iterations.

Table 2 presents the results that are achieved by the KGMO, PSO and GSA in finding the minimum value of $f1(x)$ to $f7(x)$ in 150 iterations. The *Average Gbest* and *STD* are the average global best value and standard deviation, respectively, over the 50 runs. In each run, the fitness function values were stored in an array. At the end of the final run, the average and median that were calculated from the generated array are used as the *Average cost* and the *Median cost,* respectively, to show how all of the molecules and particles converged toward the optimum. The minimum value of each function is given by $f(opt)$. The *Average Gbest* in Table 2 shows the closest results to the actual global minimum that were achieved by the PSO, GSA and KGMO.

In addition, by comparing the *STD* that was achieved by all three algorithms, it was found that the KGMO has the smallest *STD* for all of the functions $f1(x)$ to $f7(x)$, which is evidence of its robustness and efficiency in finding the global optima for unimodal functions. The small difference that was observed between the *Median* and *Average Gbest* indicates the algorithm's convergence in identifying the global minimum. This finding means that in the final iteration, all of the molecules converged on the global minimum, in such a way that the median and globally best values were close to each other. For example, in $f1(x)$, KGMO achieved an *Average Gbest* of 1.5177×10^{-67} and a *Median* of 3.8486×10^{-66}, which are very close; this closeness indicates convergence at that position. GSA achieved an *Average Gbest* of 1111 but a median of 1.1115×10^{3}, which is a significantly larger difference than that of the KGMO.

The last column of the table shows the MSE, which is calculated by:

$$MSE = \frac{\sum_{i=m}^{n+m-1} (Gbest_i - f(opt)_i)^2}{n} \tag{32}$$

where *Gbest* is the *Average Gbest* and *f(opt)* is the target in each row. For unimodal functions, $i = (1, 2, \ldots, 7)$, $m = 1$ and $n = 7$; for multimodal high-dimensional functions, $i = (8, 9, \ldots, 13)$, $m = 8$ and $n = 6$; and for multimodal functions with fixed dimensions, $i = (14, 15, \ldots, 23)$, $m = 14$ and $n = 10$.

The MSE in the last row of Table 2 shows that in 150 iterations, KGMO succeeded in decreasing the MSE by 10^7 and 10^{20} times compared to the PSO and GSA, respectively. Based on the findings in [21], for unimodal and multimodal functions with high dimensions, GSA provides good results in 1000 iterations, whereas it finds the closest value to the actual minimum in 500 iterations for multimodal functions that have fixed dimensions. Table A8 in Appendix A provides the results of the global minimum that was obtained by PSO and GSA in 1000 iterations over 50 runs. It is observed that even in 1000 iterations, KGMO achieves the global minimum with less error than PSO and GSA, which indicates that KGMO is at least 600% faster than PSO and GSA in terms of the number of iterations run to accomplish the convergence of the unimodal functions.

Table 2
Average minimization result of unimodal benchmark functions in Table 1 over 50 runs, each with 150 iterations.

	PSO	GSA	KGMO	$f(opt)$
$f1(x)$	Average Gbest: 87.0678 Average cost: 2.2124×10^{3} Median cost: 87.2559 STD: 21.5291	Average Gbest: 1111 Average cost: 2.3437×10^{3} Median cost: 1.1115×10^{3} STD: 7.7814×10^{3}	Average Gbest: 1.5177×10^{-67} Average cost: 1.2171×10^{-63} Median cost: 3.8486×10^{-66} STD: 1.3704×10^{-67}	0
$f2(x)$	Average Gbest: 13.6616 Average cost: 13.9796 Median cost: 13.6642 STD: 4.7100	Average Gbest: 1×10^{7} Average cost: 1.8637×10^{9} Median cost: 0.4029 STD: 2.2826×10^{10}	Average Gbest: 6.6717×10^{-32} Average cost: 4.3344×10^{-30} Median cost: 3.3797×10^{-30} STD: 8.9356×10^{-32}	0
$f3(x)$	Average Gbest: 15.5011 Average cost: 9451.1 Median cost: 16.0331 STD: 27.3169	Average Gbest: 48,000 Average cost: 7888.1 Median cost: 4803.4 STD: 8.3495×10^{7}	Average Gbest: 2.2503×10^{-61} Average cost: 2.4722×10^{-61} Median cost: 1.5414×10^{-58} STD: 4.2420×10^{-62}	0
$f4(x)$	Average Gbest: 19.7695 Average cost: 19.7700 Median cost: 19.7695 STD: 6.3284	Average Gbest: 10.6925 Average cost: 10.6925 Median cost: 10.6925 STD: 4.8745	Average Gbest: 1.4984×10^{-31} Average cost: 1.4485×10^{-31} Median cost: 1.0216×10^{-31} STD: 1.1348×10^{-33}	0
$f5(x)$	Average Gbest: 94.6729 Average cost: 73.2547 Median cost: 28.8522 STD: 14.2206×10^{1}	Average Gbest: 10×10^{4} Average cost: 2.6043×10^{5} Median cost: 26.0451 STD: 7.1841×10^{6}	Average Gbest: 24.8522 Average cost: 24.8522 Median cost: 24.8522 STD: 2.2277×10^{-4}	0
$f6(x)$	Average Gbest: 336 Average cost: 38.022×10^{3} Median cost: 336 STD: 11.5497	Average Gbest: 845 Average cost: 23.104×10^{3} Median cost: 79.3436 STD: 2.5766×10^{3}	Average Gbest: 0 Average cost: 0 Median cost: 0 STD: 0	0
$f7(x)$	Average Gbest: 0.0489 Average cost: 18.6847 Median cost: 0.8032 STD: 10.8253	Average Gbest: 0.2437 Average cost: 0.7 Median cost: 0.7 STD: 23.6470	Average Gbest: 0.1669×10^{-7} Average cost: 0.1621×10^{-6} Median cost: 0.1022×10^{-6} STD: 1.1119×10^{-4}	0
MSE	1.7319×10^{4}	1.4286×10^{17}	1.78×10^{-3}	

5.2.2. Multimodal high-dimensional test functions

The functions in this group are complex because there is a risk of being trapped in many local optima when finding the global optima. Therefore, a good metaheuristic algorithm is one that can find an optimum solution that is close to the actual global minimum with a high convergence rate and escapes from the local optima. As indicated by the evaluation results in Table 3, KGMO achieved better performance for *Average Gbest* in determining the global minima of all of the multimodal functions that have high dimensions, in comparison with the benchmark optimization algorithms. Additionally, this result was achieved with a smaller *STD* compared with PSO and GSA, which indicates quick movement of the algorithm toward the global minimum.

The last row of Table 3 shows that KGMO decreased the MSE by 10^5 and 10^3 times compared to PSO and GSA, respectively. As before, GSA was unable to obtain good results for $f8(x)$. For further illustration of the findings, Figs. 5 and 6 present the convergence results of KGMO, GSA and PSO in 150 iterations for $f8(x)$ and $f11(x)$, respectively. The figures show that KGMO can find the global minimum with a smaller error than PSO and GSA. It also supports that KGMO converges quickly (within 150 training cycles), as confirmed by the figures.

Table A8 in Appendix A shows that even in 1000 iterations, except for $f(12)$ and $f(13)$, KGMO achieved better global minimum results than PSO and GSA for all of the multimodal high-dimensional functions. This finding again proves that KGMO not only obtains better results but also finds those results at speeds of at least 600% faster than PSO and GSA.

5.2.3. Multimodal test functions with fixed dimensions

The third group of test functions is the set of multimodal functions that have fixed dimensions. Functions in this group have a smaller number of local optima, compared to those in the previous section. Table 4 shows the performance that is achieved by the algorithms in finding the minimum of $f14(x)$ to $f23(x)$.

Comparing the performance that was achieved by PSO and KGMO in Table 4, it is found that except for $f18(x)$ and $f20(x)$, where the PSO performed better, KGMO achieved good performance in all of the functions, and there was no significant difference in the performance of both. The very small difference in the performance of the PSO and KGMO for $f14(x)$, $f16(x)$, $f17(x)$ and $f19(x)$ can be ignored because of the randomness specification of the algorithms. For $f15(x)$ and $f21(x)$, KGMO performed better than PSO.

Comparing the performance of the GSA and KGMO, it is observed that for $f16(x)$, $f17(x)$ and $f18(x)$, there was only a small difference in the optima values found by KGMO and GSA. For other functions, KGMO performed better than GSA. From the

Table 3
Average minimization result of the multimodal benchmark functions in Table 1 over 50 runs, each with 150 iterations.

	PSO	GSA	KGMO	$f(opt)$
$f8(x)$	Average Gbest: −6751.8 Average cost: −6470 Median cost: −6649.3 STD: 76.6921×10^{1}	Average Gbest: −2268.5 Average cost: 33.9036 Median cost: 5.3449 STD: 16.3120×10^{1}	Average Gbest: −11,902 Average cost: -24.2020×10^{1} Median cost: 46.5653×10^{1} STD: 39.7420×10^{1}	−12,569
$f9(x)$	Average Gbest: 11.2384×10^{1} Average cost: 14.1680×10^{1} Median cost: 11.2622×10^{1} STD: 20.2530	Average Gbest: 31.5253 Average cost: 31.5259 Median cost: 31.5259 STD: 76.9931	Average Gbest: 5.7085×10^{-8} Average cost: 4.0373×10^{-4} Median cost: 3.0473×10^{-4} STD: 5.9230×10^{-8}	0
$f10(x)$	Average Gbest: 8.2266 Average cost: 8.3730 Median cost: 8.2375 STD: 4.1164	Average Gbest: 5.4014 Average cost: 5.4014 Median cost: 5.4014 STD: 2.6500	Average Gbest: 1.5637×10^{-4} Average cost: 0.0055 Median cost: 0.0045 STD: 4.5806×10^{-5}	0
$f11(x)$	Average Gbest: 6.3108 Average cost: 7.5496 Median cost: 6.3485 STD: 1.6864	Average Gbest: 32.3051×10^{1} Average cost: 49.6143×10^{1} Median cost: 50.5975×10^{1} STD: 44.7624	Average Gbest: 1.1102×10^{-14} Average cost: 5.1434×10^{-10} Median cost: 1.2084×10^{-10} STD: 2.0322×10^{-8}	0
$f12(x)$	Average Gbest: 5.9219 Average cost: 1.7624×10^{6} Median cost: 5.9337 STD: 4.6058	Average Gbest: 6.9332 Average cost: 5.0514×10^{6} Median cost: 6.3917 STD: 4.7665×10^{7}	Average Gbest: 1.6690 Average cost: 1.6741 Median cost: 1.6767 STD: 6.4187×10^{-5}	0
$f13(x)$	Average Gbest: 0.0974 Average cost: 0.5679 Median cost: 0.0974 STD: 0.8491	Average Gbest: 1.2229 Average cost: 1.1398×10^{7} Median cost: 1.2234 STD: 1.0528×10^{8}	Average Gbest: 3.4555×10^{-7} Average cost: 3.2414×10^{-6} Median cost: 1.6023×10^{-6} STD: 3.5447×10^{-6}	0
MSE	5.6430×10^{6}	1.77×10^{4}	11.1445×10^{1}	

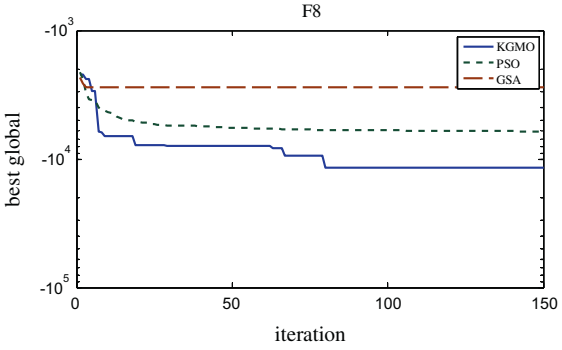

Fig. 5. *Average Gbest* for the minimization of $f8(x)$ over 50 runs, each with 150 iterations.

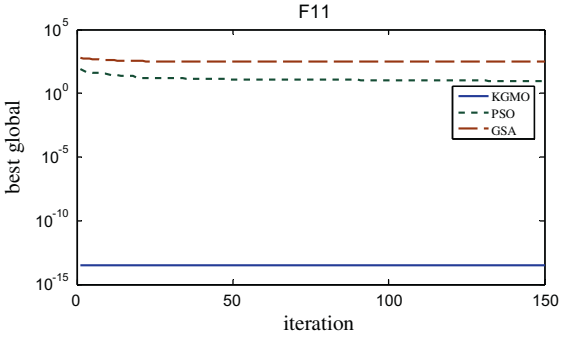

Fig. 6. *Average Gbest* for the minimization of $f11(x)$ over 50 runs, each with 150 iterations.

Table 4
Average minimization result of multimodal with fixed dimensions benchmark functions in Table 1 over 50 runs, each with 150 iterations.

	PSO	GSA	KGMO	$f(opt)$
$f14(x)$	Average Gbest: 1.2012 Average cost: 0.9926 Median cost: 0.7301 STD: 0.3826	Average Gbest: 3.9731 Average cost: 13.6186 Median cost: 13.6186 STD: 0.8365	Average Gbest: 1.3718 Average cost: 7.0274 Median cost: 2.4728 STD: 3.8411×10^{-15}	1
$f15(x)$	Average Gbest: 0.0016 Average cost: 0.0016 Median cost: 0.0016 STD: 0.0098	Average Gbest: 0.0069 Average cost: 100 Median cost: 0.0220 STD: 0.0087	Average Gbest: 0.0012 Average cost: 12.9248 Median cost: 8.0856 STD: 0.0278	0.0003075
$f16(x)$	Average Gbest: −1.0316 Average cost: −1.0316 Median cost: −1.0316 STD: 1.5701×10^{-16}	Average Gbest: −1.0316 Average cost: −1 Median cost: −1 STD: 0.2621	Average Gbest: −1.0113 Average cost: −0.9835 Median cost: −0.9835 STD: 0.4468	−1.03
$f17(x)$	Average Gbest: 0.3979 Average cost: 0.3979 Median cost: 0.3979 STD: 0.7297	Average Gbest: 0.3979 Average cost: 0.3979 Median cost: 0.3979 STD: 0.0638	Average Gbest: 0.4072 Average cost: 34.1087 Median cost: 37.1256 STD: 0.2527	0.398
$f18(x)$	Average Gbest: 3.0000 Average cost: 3.0000 Median cost: 3.0000 STD: 4.0782×10^{-15}	Average Gbest: 3.0000 Average cost: 1 Median cost: 1 STD: 1.3457	Average Gbest: 3.7423 Average cost: 0.3979 Median cost: 0.3979 STD: 13.9212	3
$f19(x)$	Average Gbest: −3.8628 Average cost: −3.8628 Median cost: −3.8628 STD: 2.5640×10^{-16}	Average Gbest: −3.3620 Average cost: −0.3351 Median cost: −0.3351 STD: 0.1459	Average Gbest: −3.8467 Average cost: −0.4679 Median cost: −0.3005 STD: 0.0371	−3.86
$f20(x)$	Average Gbest: −3.2031 Average cost: −3.2031 Median cost: −3.2031 STD: 0.0627	Average Gbest: −2.2192 Average cost: −0.7865 Median cost: −0.7865 STD: 0.0802	Average Gbest: −2.9012 Average cost: −0.0193 Median cost: -3.0415×10^{-4} STD: 0.3584	−3.32
$f21(x)$	Average Gbest: −2.6552 Average cost: −2.6552 Median cost: −2.0552 STD: 3.7474	Average Gbest: −5.0552 Average cost: −5.0552 Median cost: −5.0552 STD: 1.3595	Average Gbest: −10.1532 Average cost: −2.3266 Median cost: −2.1440 STD: 2.0656×10^{-4}	−10
$f22(x)$	Average Gbest: −10.4023 Average cost: −10.4029 Median cost: −10.4029 STD: 2.6371	Average Gbest: −5.0877 Average cost: −5.0877 Median cost: −5.0877 STD: 1.3069	Average Gbest: −10.4023 Average cost: −1.7011 Median cost: −2.1815 STD: 3.2995×10^{-4}	−10
$f23(x)$	Average Gbest: −10.5364 Average cost: −10.5364 Median cost: −10.5364 STD: 4.5749	Average Gbest: −10.5364 Average cost: −10.5364 Median cost: −10.5364 STD: 3.1853	Average Gbest: −10.5352 Average cost: −1.9536 Median cost: −2.2269 STD: 2.5275×10^{-4}	−10
MSE	5.4482	5.9169	0.1337	

comparison of the results achieved for multimodal functions with fixed dimensions, it can be concluded that KGMO suffered considerable error in finding the global optima for $f18(x)$ and $f20(x)$. This weakness for only two functions is justifiable based on the "no free lunch theorem of optimization problems" [32] because one algorithm cannot be the best solution for all optimization problems.

The obtained *STD* with a PSO for the functions $f(15)$ and $f(16)$ is smaller than that of KGMO. However, because the values are not significantly different, the capabilities of the PSO and KGMO are considered to be approximately the same in finding the global optima of these benchmark functions.

The MSE in the last row suggests that KGMO can achieve approximations to the global minimum values that are closer to the actual global minima with a smaller MSE of approximately 0.1, compared to approximately 5 with PSO and GSA. Figs. 7 and 8 present the comparison of the results that are achieved by the PSO and GSA with KGMO for the functions $f21(x)$ and $f22(x)$. The results support that KGMO performs well in achieving the optimal minimum in less than 150 iterations.

Comparing the results in Table A8 with the results obtained for KGMO in Table 4 indicates that except for $f15(x)$ and $f18(x)$, in which PSO obtained a better result, KGMO managed within 150 iterations to outperform PSO and GSA in 500 iterations for the third type of benchmark function. KGMO was at least 300% faster in terms of the number of iterations that were required to obtain the global minimum.

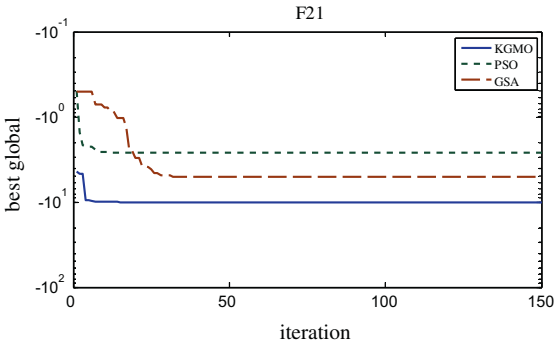

Fig. 7. *Average Gbest* for the minimization of $f21(x)$ over 50 runs, each with 150 iterations.

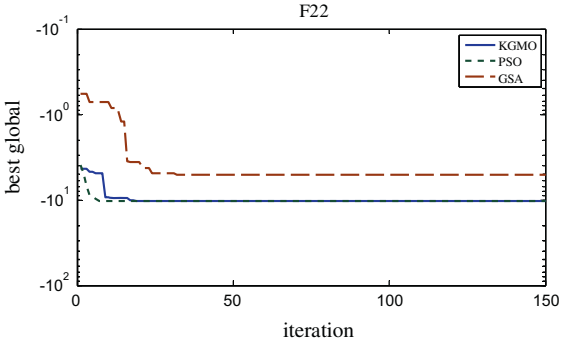

Fig. 8. *Average Gbest* for the minimization of $f22(x)$ over 50 runs, each with 150 iterations.

6. Results and discussion

The Wilcoxon rank sum test is one of the most well-known non-parametric significance tests for assessing whether one of two samples of independent observations tends to have larger values than the other [30]. The Wilcoxon rank sum test for equal medians was undertaken to further evaluate the performance of each algorithm in terms of the achieved *Average Gbest* against the actual optimum solution $f(opt)$. Table 5 shows the results that were obtained for the Wilcoxon rank sum test with $\alpha = 0.05$ [30]. The results achieved by PSO ($P = 0.0068$) and GSA ($P = 0.0052$) are less than 0.05, which indicates that the *Average Gbest* for PSO and GSA were significantly different from $f(opt)$. In comparison, the obtained $P = 0.0505$ for KGMO, which is greater than 0.05, shows that the achieved *Average Gbest* was not significantly different from $f(opt)$; hence, there is evidence of better accuracy being achieved by the KGMO.

Another test for measuring the performance of the proposed KGMO is the Average Ranking (AR) method, in which one rank is assigned to each algorithm for all of the functions, based on the obtained results and, finally, the ranks are averaged. In Table 5, the best rank (i.e., 3) is assigned to the algorithm that has the best *Average Gbest* closest to the actual minimum; rank 2 is assigned to the second algorithm in finding the optima and rank 1 to the worst algorithm. If the algorithms have the same best and worst *Average Gbest*, then rank 3 and 1, respectively, are assigned to them. As Table 5 presents, the average ranking for the results achieved by the KGMO is 2.69, which is higher than 1.91 and 1.56 for the PSO and GSA, respectively. This finding indicates better performance of the KGMO compared to that of the benchmark algorithms, which is again in accordance with previous results.

Consolidating the results of the last rows of Tables 2–4 for the three types of benchmark functions in Table 6, the *Average MSE* for each column is calculated. For clarity and analysis, graphical representations of the results in Table 6 are presented in a logarithmic representation in Figs. 9 and 10. The results in Table 6 and Fig. 9 show that the *Average MSE* for KGMO is approximately 23, which is a decrease of 10^1 and 10^4 times compared to PSO and GSA, respectively. As indicated in Fig. 10, KGMO achieved significantly fewer errors for the three types of functions, compared to PSO and GSA. The logarithmic representation of the Mean Absolute Error (MAE) of the *Average Gbest* and global minima achieved by the algorithms is given in Fig. 11. The results show that KGMO suffered less error than both PSO and GSA for the 23 benchmark functions. The MAE is calculated by the following formula with $n = 23$, $Gbest_i$ being the *Average Gbest* and $f(opt)_i$ being the global minimum for each function:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Gbest_i - f(opt)_i| \tag{33}$$

Table 5
Results of the average ranking test and the Wilcoxon rank sum test for equal medians for *Average Gbest* and $f(opt)$. The obtained P = 0.0505 for KGMO, which is greater than 0.05, shows that the achieved *Average Gbest* was not significantly different from $f(opt)$. In addition, the average ranking for the results achieved by the KGMO is higher than average ranking results for PSO and GSA, which shows the better performance of KGMO.

	PSO	GSA	KGMO	$f(opt)$
$f1(x)$	87.0678	1111	1.52×10^{-64}	0
Rank	2	1	3	
$f2(x)$	13.6616	10,000,000	6.67×10^{-32}	0
Rank	2	1	3	
$f3(x)$	15.5011	48,000	2.25×10^{-61}	0
Rank	2	1	3	
$f4(x)$	19.7695	10.6925	1.50×10^{-33}	0
Rank	2	1	3	
$f5(x)$	94.6729	10,000	28.8522	0
Rank	2	1	3	
$f6(x)$	336	845	0	0
Rank	2	1	3	
$f7(x)$	0.0489	0.2437	1.67×10^{-8}	0
Rank	2	1	3	
$f8(x)$	−6751.8	−2268.5	−11,902	−12,569
Rank	2	1	3	
$f9(x)$	112.3839	31.5253	5.71×10^{-8}	0
Rank	1	2	3	
$f10(x)$	8.2266	5.4014	1.56×10^{-4}	0
Rank	1	2	3	
$f11(x)$	6.3108	323.0513	1.11×10^{-14}	0
Rank	2	1	3	
$f12(x)$	5.9219	10,000	1.669	0
Rank	2	1	3	
$f13(x)$	0.0974	100,000	3.46×10^{-7}	0
Rank	2	1	3	
$f14(x)$	1.2012	3.9731	1.3718	1
Rank	3	1	2	
$f15(x)$	0.0016	0.0069	0.0012	0.0003075
Rank	1	2	3	
$f16(x)$	−1.0316	−1.0316	−1.0113	−1.03
Rank	3	3	1	
$f17(x)$	0.3979	0.3979	0.4072	0.398
Rank	3	3	2	
$f18(x)$	3	3	3.7423	3
Rank	3	3	2	
$f19(x)$	−3.8628	−3.362	−3.8467	−3.86
Rank	3	1	2	
$f20(x)$	−3.2031	−2.2192	−2.9012	−3.32
Rank	3	1	2	
$f21(x)$	−2.6552	−5.0552	−10.1532	−10
Rank	1	2	3	
$f22(x)$	−10.4023	−5.0877	−10.4023	−10
Rank	3	1	3	
$f23(x)$	−10.5364	−10.5364	−10.5352	−10
Rank	3	3	3	
P value	0.0068	0.0052	**0.0505**	
Average rank	1.9133	1.5652	**2.6956**	

Table 6
The average MSE for the 3 models of benchmark functions.

	PSO	GSA	KGMO
Unimodal functions	47.22682	83.8330×10^{4}	2.40
Multimodal high-dimensional functions	59.5014×10^{1}	12.066×10^{3}	66.9
Multimodal functions with fixed dimensions	5.445	5.9168	0.134
Average MSE	21.5895×10^{1}	28.3467×10^{4}	23.1

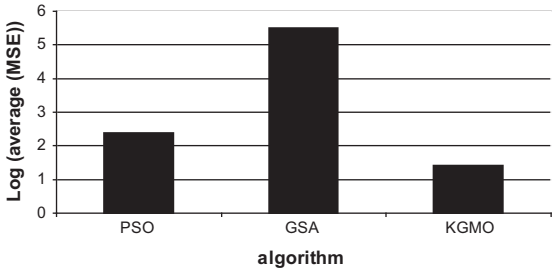

Fig. 9. Average MSE achieved by the algorithms.

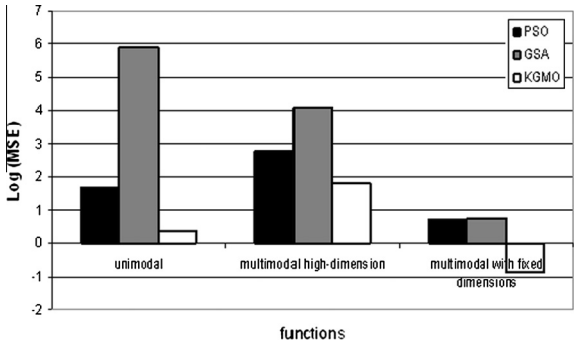

Fig. 10. MSE of the algorithms for each type of benchmark function (Tables 2–4).

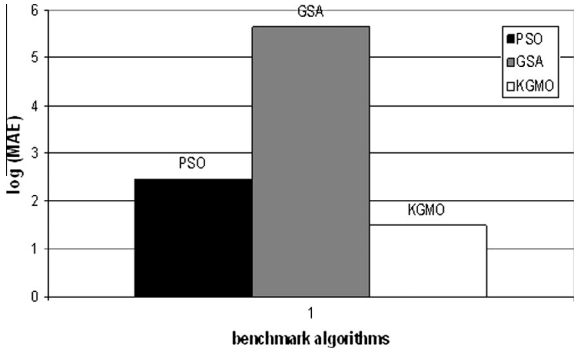

Fig. 11. MAE of the algorithms for the 23 benchmark functions.

7. Conclusions

In this paper, a new optimization algorithm that is based on kinetic energy and the natural motion of gas molecules is proposed. A performance evaluation of the proposed algorithm with other well-known benchmark optimization algorithms using the 23 standard benchmark functions indicates that the proposed KGMO not only can converge toward the global minima quickly in less than 150 iterations but is also more accurate and can decrease the MSE by 10^1 and 10^4 times compared to the PSO and GSA, respectively.

The results obtained in this study indicate that the KGMO is at least 600% faster for the unimodal and multimodal high-dimensional functions and at least 300% faster for the multimodal functions with fixed dimensions. Regarding the equations and complexity associated with these operations, the efficiency of the KGMO is also evident. Additionally, the proposed algorithm has been demonstrated to be better able to converge toward the global optima compared to the GSA and PSO without being trapped in local minima. It is expected that the proposed KGMO algorithm would be able to improve the performance of most systems that require a solution to optimization problems.

Appendix A.

Tables A1–A8 are prepared in this part, as in [24].

Table A1
a_{ij} in $f14$.

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

Table A2
a_i and b_i in $f15$.

i	1	2	3	4	5	6	7	8	9	10	11
a_i	0.1957	0.1947	0.1735	0.16	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 A3
a_{ij} and c_i in $f19$.

i	$a_{ij}, j = 1, 2, 3$			c_i
1	3	10	30	1
2	0.1	10	35	1.2
3	3	10	30	3
4	0.1	10	30	3.2

Table A4
p_{ij} in $f19$.

i	$p_{ij}, j = 1, 2, 3$		
1	0.3689	0.1170	0.2673
2	0.4699	0.4387	0.7470
3	0.1091	0.8732	0.5547
4	0.03815	0.5743	0.8828

Table A5
$a_{ij}$and c_i in $f20$.

i	$a_{ij}, j = 1, 2, 3, 4, 5, 6$						c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

Table A6
p_{ij} in $f20$.

i	$p_{ij}, j = 1, 2, 3, 4, 5, 6$					
1	0.131	0.169	0.556	0.012	0.828	0.588
2	0.232	0.413	0.830	0.373	0.100	0.999
3	0.234	0.141	0.352	0.288	0.304	0.665
4	0.404	0.882	0.873	0.574	0.109	0.038

Table A7
a_{ij} and c_i in $f21$, $f22$ and $f23$.

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 A8
Minimization result of benchmark functions for PSO and GSA.

	PSO	GSA
$f1(x)$	1.22×10^{-8}	2.12×10^{-17}
$f2(x)$	10.3311	2.27×10^{-8}
$f3(x)$	12.2221	238.99
$f4(x)$	6.9021	3.20×10^{-9}
$f5(x)$	52.6255	26.13
$f6(x)$	0	0
$f7(x)$	0.0434	0.0487
$f8(x)$	−9660.9	−2482.8
$f9(x)$	93.3482	17.9093
$f10(x)$	6.1121	3.49×10^{-3}
$f11(x)$	2.6732	2.4271
$f12(x)$	4.7490	0.1
$f13(x)$	5.52×10^{-3}	3.2×10^{-32}
$f14(x)$	0.7301	1.9923
$f15(x)$	5.33×10^{-3}	0.0084
$f16(x)$	−1.0316	−1.0316
$f17(x)$	0.3980	0.3979
$f18(x)$	3.00	3.00
$f19(x)$	−3.86	−3.8347
$f20(x)$	−3.3220	−1.3039
$f21(x)$	−10.1532	−5.0552
$f22(x)$	−10.4023	−5.08
$f23(x)$	−10.5364	−10.5364

Note: Maximum number of iterations for unimodal functions and multimodal high-dimensional functions is 1000 and for multimodal functions with fixed dimensions is 500.

Reference

[1] R. Arkin, Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.
[2] G. Beni, J. Wang, Swarm intelligence in cellular robotics systems, in: Proc. NATO Advanced Workshop on Robots and Biological System, Tuscany, Italy, 1989, p. 102.
[3] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, NY, 1999.
[4] A. Buczak, R. Uhrig, Hybrid fuzzy-genetic technique for multisensor fusion, Intell. Syst. 93 (1996) 265–281.
[5] Y.P. Chen, P. Jiang, Analysis of particle interaction in particle swarm optimization, Theoret. Comput. Sci. 411 (2010) 2101–2115.
[6] L. Chun-an, New dynamic constrained optimization PSO algorithm, in: Proc. Fourth International Conference on Natural Computation, Jinan, China, 2008, pp. 650–653.
[7] S. Devi, D.G. Jadhav, S.S. Pattnaik, PSO based memetic algorithm for unimodal and multimodal function optimization, Swarm Evol. Memetic Comput. 7076 (2011) 127–134. Lecture Notes in Computer Science.
[8] H.K. Dong, A. Abraham, H.C. Jae, A hybrid genetic algorithm and bacterial foraging approach for global optimization, Inf. Sci. 177 (2007) 3918–3937.
[9] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybernet. 26 (1996) 29–41.
[10] R. Eberhart, Y. Shi, Particle swarm optimization: developments, application and resources, in: Proceedings of IEEE Congress of Evolutionary Computation, Seoul, Korea, 2001, pp. 81–86.
[11] R. Eberhart, S. Yuhui, J. Kennedy, Swarm Intelligence; (The Morgan Kaufmann Series in Evolutionary Computation), Academic Press, NY, 2001.
[12] J.D. Farmer, N.H. Packard, A.S. Perelson, The immune system, adaptation and machine learning, Physica D 2 (1986) 187–204.
[13] G. Flake, The Computational Beauty of Nature, MIT Press, Cambridge, MA, 1999.
[14] R.A. Formato, Central force optimization: a new nature inspired computational framework for multidimensional search and optimization, Stud. Computat. Intell. 129 (2008) 221–238.
[15] D. Holliday, R. Resnick, Fundamentals of Physics, John Wiley and Sons, 1993.
[16] S.J. Hopwood, J. Jeans, An Introduction to the Kinetic Theory of Gases, Cambridge University Press, UK, 2009.

[17] L. Idoumghar, M. Melkemi, R. Schott, M.I. Aouad, Hybrid PSO-SA type algorithms for multimodal function optimization and reducing energy consumption in embedded systems, in: Applied Computational Intelligence and Soft Computing, Hindawi, 2011, 12p.
[18] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, vol. 4, 1995, 1942–1948.
[19] R.E. Korf, Artificial Intelligence Search Algorithm, Computer Science Department, University of California, Los Angeles, CA, 1995 (July).
[20] L.B. Loeb, The Kinetic Theory of Gases, Dover Phoenix Editions, NY, 2004.
[21] P. Pareja-Tobes, D. Pelta, A. Sancho-Royo, J.L. Verdegay, Search spaces representation in optimization problems, Exp. Syst. Appl. 34 (2008) 2891–2895.
[22] V.A. Parsegian, Van der Waals Forces: A Handbook for Biologists, Chemists, Engineers, and Physicists, Cambridge University Press, 2006.
[23] D. Pelta, N. Krasnogor, C. Bousono-Calzon, J.L. Verdegay, J. Hirst, E.K. Burke, A fuzzy sets based generalization of contact maps for the overlap of protein structures, J. Fuzzy Sets Syst. 152 (2005) 103–123.
[24] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[25] A. Sancho-Royo, D. Pelta, J.L. Verdegay, J.R. Gonzalez, Evaluation of co-operative strategies in optimization problems, in: Proc. Nisis, 2005, <http://www.nisis.de>.
[26] E. Shaw, The schooling of fishes, Sci. Am. 206 (1962) 128–138.
[27] T. Si1, N.D. Jana1, J. Sil, Constrained function optimization using PSO with polynomial mutation, Swarm, Evol Memetic Comput 7076 (2011) 209–216 (Lecture Notes in Computer Science).
[28] S.N. Sivanandam, S.N. Deepa, Introduction to Genetic Algorithms, Springer, 2007.
[29] G. Venter, R.T. Haftka, Constrained particle swarm optimization using a bi-objective formulation, Struct. Multidiscipl. Optim. 40 (2010) 65–76.
[30] F. Wilcoxon, Individual comparisons by ranking methods, Biomet. Bull. 1 (1945) 80–83.
[31] U. Wilensky, NetLogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999, <http://ccl.northwestern.edu/netlogo>.
[32] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[33] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[34] Y. Liu, Z. Yi, H. Wu, M. Ye, K. Chen, A TABU search approach for the minimum sum-of-squares clustering problem, Inf. Sci. 178 (2008) 2680–2704.
[35] Z.H. Zhan, J. Zhang, Y. Li, H.S.H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybernet.—Part B: Cybernet. 39 (2009) 1362–1381.
[36] J.L. Zne, Y.L. Chou, A hybrid search algorithm with heuristics for resource allocation problem, Informat. Comput. Sci. 173 (2005) 155–167.