Applied Soft Computing 10 (2010) 1132–1140

Contents lists available at ScienceDirect

Applied Soft Computing

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

Grenade Explosion Method—A novel tool for optimization of multimodal functions

Ali Ahrari*, Ali A. Atai

Mechanical Engineering Department, Engineering Faculty, University of Tehran, Tehran, Iran

ARTICLE INFO

Article history:
Received 31 July 2008
Received in revised form 9 November 2009
Accepted 27 November 2009
Available online 5 December 2009

Keywords:
Grenade Explosion Method
Global optimization
Multimodal functions
Evolutionary algorithm

ABSTRACT

This work presents a new optimization technique called Grenade Explosion Method (GEM). The fundamental concepts and ideas which underlie the method are fully explained. It is seen that this simple and robust algorithm is quite powerful in finding all global and some local optima of multimodal functions. The method is tested with several multimodal benchmark functions and the results show it usually converges to the global minima faster than other evolutionary methods such as Genetic Algorithm (GA) and Artificial Bee Colony (ABC). Based on the performance on classical benchmark functions, the efficiency of the method in solving engineering applications can be highly appreciated.

1. Introduction

Among optimization methods, Evolutionary Algorithms (EAs) which are generally known as general purpose optimization algorithms, are capable of finding the near-optimal solution to the numerical real-valued test problems for which exact and analytical methods do not produce the optimal solution within a reasonable computation time [5,7], especially when the global minimum is surrounded by many local minima. Furthermore, EAs don't make use of the derivatives of the objective function. These algorithms are usually devised by observing phenomena happening in nature, like Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) and Bee Colony Optimization.

Genetic Algorithm is based on the genetic process of biological organisms [1,14]. Over many generations, natural populations evolve according to the principles of natural selections, i.e. survival of the fittest. In this method, a potential solution to a problem is represented as a set of parameters. Each independent variable is represented by a gene. Combining the genes, a chromosome is produced which represents a solution. In genetic terminology, the set of parameters represented by a particular chromosome is referred to as an individual. During the reproduction phase, the individuals are selected from the population and recombined. Parents are randomly selected from the population using a scheme, which favors the fitter individuals. Having selected two parents, their chromosomes are recombined; typically using mechanisms of crossover and mutation. Mutation is usually applied to some individuals, to guarantee population diversity.

PSO is a recently developed heuristic algorithm technique, inspired by choreography of a bird flock [11]. The approach can be viewed as a distributed behavioral algorithm that performs a multidimensional search. It makes use of a velocity vector to update the current position of each particle in the swarm [8]. The velocity vector is updated based on the memory gained by each particle, conceptually resembling an autobiographical memory, as well as the knowledge gained by the swarm as a whole. Thus, the position of each particle, which represents a solution to the optimization problem, is updated based on the social behavior of the swarm. In [12] there are some suggestions for choosing the parameters used in PSO.

Artificial Bee Colony (ABC) Algorithm is an optimization algorithm based on the intelligent behavior of honey bee swarm. The colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts [4]. Onlooker bees wait in dance area for making decision to choose a food source, employed bees hover around a discovered food source and scout bees try to locate new sources. Bees come into the hive and share the nectar information of the sources with the bees waiting on the dance area within the hive. Then, every employed bee goes to the food source area visited by her at the previous cycle since that food source exists in her memory, and then chooses a new food source by means of visual information in the neighborhood of the present one [4]. Based on the shared information, an onlooker bee is employed at one of the food sources, where probability of recruitment is proportional to profitability of the food source. In the ABC algorithm, the position

* Corresponding author. Tel.: +98 9125493301; fax: +98 2188013029.
E-mail address: aliahrari1983@gmail.com (A. Ahrari).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.11.032

of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution [4].

Ant colony system was inspired by the foraging behavior of real ants [6]. This behavior enables ants to find the shortest path between food sources and their nest. This functionality of real ant colonies is exploited in artificial ant colonies in order to solve optimization problems [2,3].

Simulated Annealing (SA) was originally proposed to simulate the annealing process [10,13]. After generating an initial solution, it attempts to move from the current solution to one of its neighborhood solutions. If the new solution results in a better objective value, it is accepted. However, if the new solution yields a worse value, it can be accepted according to a probability. By accepting worse solutions, SA can avoid being trapped in local optima.

In these algorithms, except SA, a population of agents [5] searches the space. Each agent represents a solution for the optimization problem. During iterations, a set of new solutions are produced, the best solutions are kept and allowed to produce new solutions in the next iterations. This process goes on until the convergence criterion is met.

One of the most important disadvantage in population based algorithm, is crowding of the agents, which shows the convergence of the algorithm to a point in the crowded region. If it happens in the early iterations of the algorithm, solution to which the algorithm has converged is probably a local minimum, because the design space has not been explored adequately. Furthermore, in the final population, similar agents don't present different solutions, which can be a disadvantage especially when the objective function has several global minima.

In EAs, several random numbers are produced each iteration. Consequently to analyze and predict the performance of these algorithms, knowing the probability of an event is important rather than whether it happens or not. Furthermore, the convergence process and probably the final solution may be different in each independent run. To obtain a reliable solution or test the reliability of an optimization algorithm, several independent runs should be executed.

An objective function may have several global minima, i.e. several points in which the value of the objective function is equal to the global minimum value. Furthermore, it may have some local minima in which the value of the objective function is close to the global minimum value. As there are usually many simplifications in writing a standard mathematical form for an industrial optimization problem, finding all global minima or even these local minima can present a verity of quite optimal solutions for the decision maker.

The present work introduces a novel evolutionary algorithm for optimizing real-valued bounded black-box optimization problems. This method, which we call Grenade Explosion Method (GEM), is based on the clues given in the next sections. In Section 2, the method and the concepts behind it are introduced and the detailed algorithm is presented. Guidelines for selecting related parameters are also given. In Section 3, several optimization tests are performed with this new method and the results are compared with those from the works of others and major advantages of the new method are highlighted. In Section 4, some special advantages of the proposed method which can hardly be seen in other EAs are explained and verified.

2. Grenade Explosion Method (GEM)

In this section, the fundamental ideas behind this method are elaborated and guidelines for its parameter tuning are presented.

2.1. Basic ideas

The idea of the presented algorithm is based on observation of a grenade explosion, in which the thrown pieces of shrapnel destruct the objects near the explosion location. L_e is the length of explosion along each coordinate, in which the thrown piece of shrapnel may destruct the objects. The loss caused by each piece of shrapnel is calculated. A high value for loss per piece of shrapnel in an area indicates there are valuable objects in that area. To make more loss, the next grenade is thrown where the greatest loss occurs. Although the objects near grenade's location are more likely to be damaged, the probability of destruction is still kept for farther objects by choosing a high value for L_e. This process would result in finding the best place for throwing the grenades, even though shrapnel cannot discover this area in early iterations. The loss caused by destruction of an object is considered as the fitness of the objective function at the object's location. Suppose that X is the current location of a grenade:

$$X = \left\{X_m\right\}, \quad m = 1, 2, \ldots, n$$

In which n is the search space dimension, equal to the number of independent variables. Now N_q pieces of shrapnel are produced by the grenade explosion and destruct objects that are in X'_j location:

$$X'_j = \left\{X_m + sign(r_m) \times |r_m|^p \times L_e\right\}, \quad j = 1, 2, \ldots, N_q \tag{1}$$

where r_m is a uniformly distributed random number in $[-1,1]$ and p is a constant. A high value for p lets pieces of shrapnel search the region near the exploded grenade more accurately, while a low one lets them to explore farther regions better.

Considering Eq. (1), it is obvious that exploration for more valuable items performs in an n-dimensional cubic space extended $2L_e$ units along each coordinate and the grenade is located at the center of this cube.

To use this algorithm, an independent variable range is scaled to $[-1,1]$. Using Eq. (1), some produced shrapnel may collide to objects outside the feasible space. To increase the convergence rate and exploration of near-boundary regions more accurately, such a collision location is transported to a new location inside the feasible region according to the following scheme:

$$\text{if } X'_j \notin [-1,1]^n \Rightarrow \left(B'_j = \frac{X'_j}{\left|\text{Largest component of } X'_j \text{ in value}\right|}\right)$$
$$\rightarrow B''_j = r'_j \times \left(B'_j - X\right) + X, \tag{2}$$
$$\begin{cases} j = 1 \text{ to } N_q \text{ (Shrapnel Number)} \\ 0 < r'_j < +1 \text{ (Random Number)} \end{cases}$$

where X'_j is the collision location outside the feasible space and B'' is the new location inside the feasible space. Fig. 1 depicts this procedure schematically.

One of the special concepts of this algorithm which can hardly be seen in other EAs is the agent's territory radius (R_t), which means an agent (in this algorithm agents are grenades) does not let other agents come closer than a specific distance, which is specified by R_t. When several agents are exploring the feasible space, a high value for this parameter makes sure that grenades are spread quite uniformly in the feasible space and the whole space is being explored, while a low value for this parameter lets the grenades get closer to search the local region all together. Fig. 2 depicts the range of explosion and territory radius in the two-dimensional space.

A higher value for the explosion range makes it possible to explore farther regions (better exploration), while a lower one lets the grenades focus on the region nearby (better exploitation). The

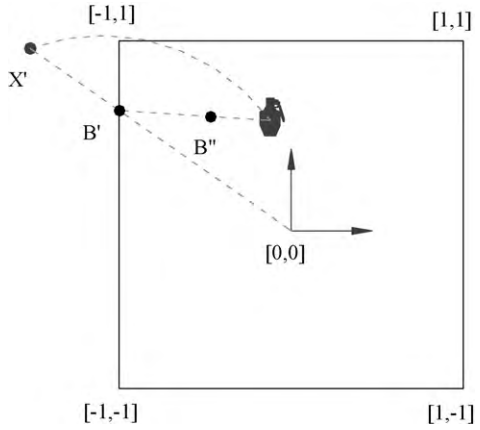

Fig. 1. Transportation of an unfeasible point (X') to a feasible location (B'').

value of exponent p determines the intensity of exploration. This parameter is updated based on the value of T_w:

$$p = \max\left\{1, n \times \frac{\log\left(R_t/L_e\right)}{\log(T_w)}\right\} \tag{3}$$

where T_w is the probability that a produced piece of shrapnel collides an object in an n-dimension hyper-box which circumscribes the grenade's territory.

2.2. Improving the algorithm performance

To increase the global search ability, a High value for R_t should be chosen at the beginning ($R_{t-initial}$) and reduced gradually to let the grenades search the probable global minimum location found altogether for a precise answer. A simple method to reduce R_t, which is used in this article is:

$$R_t = \frac{R_{t-initial}}{(R_{rd})^{(iteration\,No/total\,No\,of\,iterations)}} \tag{4}$$

The value of R_{rd} is set before the algorithm starts. This parameter represents the ratio of the value of R_t in the first iteration to its value in the final iteration. Furthermore, L_e is reduced according to the following equation:

$$L_e = (L_{e\_initial})^m (R_t)^{1-m}, \quad 0 \le m \le 1 \tag{5}$$

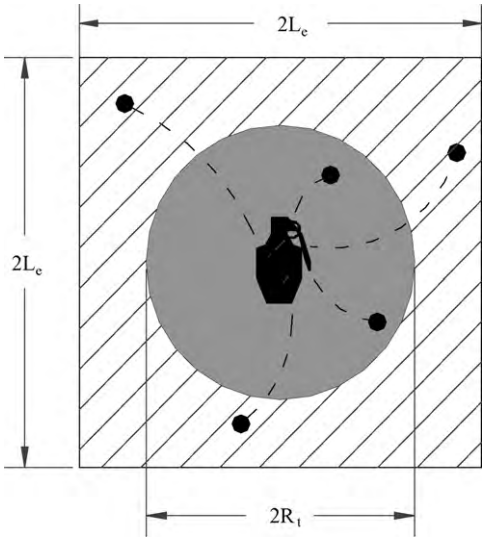

Fig. 2. Range of explosion and territory radius in the two-dimensional space.

Which indicates that L_e is reduced more slowly than R_t during the iterations in order to save the global search ability. m can be constant during the algorithm, or reduced from a higher value to a lower one.

Now the steps in GEM can be stated as follows:

1. Scale the independent variables' ranges to $[-1,1]$.
2. Select problem parameters:
   (N_g, N_q, $L_{e-initial}$, $R_{t-initial}$, R_{rd}, $Max\_iterations$, $m_{\max}$, $m_{\min}$, T_w).
3. Let ($L_e = L_{e-initial}$, $R_t = R_{t-initial}$, $Iteration\_number = 1$).
4. Generate N_g grenades in random locations in an n-dimension space $\vec{X}_i \in [-1, 1]^n$, $(i = 1, \ldots, N_g)$ which are a distance of at least R_t apart from each other.
5. While $Iteration\_number \le Max\_iterations$.
6. Arrange the grenades based on their fitness (The best grenade is the first).
7. Let: $i = 1$.
8. While N_q accepted points are generated around ith grenade.
9. Generate a point (X') around ith grenade according to Eq. (1).
10. If (X') is outside $[-1,1]^n$, transport it into $[-1,1]^n$, according to Eq. (2).
11. If X' is a distance of at least R_t apart from the location of grenades $1, 2, \ldots, i-1$, then X' is accepted.
12. End while % N_q *pieces of shrapnel are thrown*.
13. Calculate the fitness of the new generated points around the ith grenade. If the fitness of the best point is better than current location of the ith grenade, move the grenade to the location of the best point.
14. Let: $i = i + 1$.
15. If $N_g \ge i$ then go to 8.
16. Reduce R_t according to Eq. (4):
17. Calculate exponent m according to the relation:

$$m = m_{\max} - \left(\frac{Iteration\_number}{Max\_iterations}\right)(m_{\max} - m_{\min})$$

18. Update L_e according to Eq. (5).
19. Update exponent p according to Eq. (3).
20. End While % *Number of iterations*.

2.3. Several guidelines for tuning parameters

Unsuitable values for algorithm parameters may result in a low convergence rate, convergence to a local minimum or unreliability of solutions.

- Number of grenades plays the same role as number of bee colonies in BCO. For an ordinary optimization problem, two or three grenades and 10–15 pieces of shrapnel per grenade would be sufficient. For a complicated problem, it is preferred to increase the number of pieces of shrapnel rather than number of grenades. N_q can be raised up to 100 if necessary. When finding several global minima (if available) or local minima whose values are close to the global minimum value, a number of 1.5 or 2 times of the number of minima desired would be sufficient. Increasing the number of grenades more than enough may lead to a low convergence rate.
- The initial value for the radius of the grenade territory ($R_{t-initial}$) should be large enough so that the possibility of grenade crowding would be eliminated in the early iterations of the algorithm. If $R_{t-initial}$ is very large, it takes a long time to put them in an acceptable configuration; larger values may make it impossible. The optimal value for $R_{t-initial}$ depends on the number of grenades and the space dimension. The value of R_{rd} depends on the desired accuracy of independent variables. Furthermore, a high value for

R_{rd} lead to more precise solutions, but it increases the probability of convergence to local minimum.

- Initial range of explosion ($L_{e-initial}$) should be large enough so that the whole $[-1, 1]^n$ space can be searched at the beginning. When there are only a few grenades, let $L_{e-initial} \approx 2$. A lower value can be chosen when there are several grenades, especially when the goal is to locate several different minima.
- Parameter m in Eq. (5) demonstrates the weight of $L_{e-initial}$ in L_e value in each iteration. It makes sure that L_e is kept large in comparison with R_t so that the far regions can be explored for new high-fitted locations and the algorithm would not be trapped in local minima. This, somehow works like mutation operation in GA, or scout bees in BCO, but there's a pattern for the random search, i.e. although farther points have the chance to be explored, but the chance rises for closer points, which means the closer, the more probable. There's no strict rule to distinguish pieces of shrapnel that participate in the random search from those that participate in the local search, i.e. all individuals participate in both global and local search simultaneously; only the weight of each kind of search varies. It is suggested keeping m high especially when the algorithm is trying to find the global minimum region. When it found this region, it can be reduced to find the exact location.
- A higher value for number of total iterations increases the probability of finding near optimum solution or solutions, and at the same time, increases the accuracy of the found solutions.
- A low value for T_w increase the global search and finding near-optimal solution ability, while a high value lets the algorithm to focus on the minimum found and increase the accuracy.

3. Validation of GEM

In this section, seven optimization benchmark functions are optimized by using GEM and the solutions obtained are compared with results of other optimization methods presented in [9]. Table 1 presents specifications of these functions.

The optimization would stop when the difference between the maximum fitness obtained and the global optimum value is less than 0.1% of the optimum value, or less than 0.001, whichever is smaller. In case the optimum value is 0, the solution is accepted if it differs from the optimum value by less than 0.001 [9]. Table 2 presents the values chosen for parameters used in GEM.

Table 3 presents the results obtained by GEM and those by the deterministic Simplex method (SIMPSA), the stochastic simulated annealing optimization procedure (NE-SIMPSA), Genetic Algorithm (GA), the Ant Colony System (ANTS) and Bees Algorithm [9].

The numbers of points visited are average for 100 independent runs. Results for all methods except GEM were directly driven from [9].

Result comparison reveals that in seven out of eight tests, GEM has found the global minimum with the desired accuracy faster than other optimization methods. This superiority is more obvious for functions 3, 4 and 7(at list twice better than any of other methods). Ignoring the results obtained by GEM for the sixth function, only Results of Bee Colony Optimization can compete with those of GEM.

4. Special abilities of GEM

Besides the general properties of an ordinary optimization method, GEM shows special abilities in finding solutions for an optimization problem, which can be considered as finding the global minimum while surrounded by many local minima, finding all global minima of functions with multiple global minima and finding

Table 1
Specifications of seven benchmark functions presented in [9].

No	Function name	Interval	Function	Global optimum
1	De Jong	$[-2.048, 2.048]$	$\max F = 3905.93 - 100(x_1^2 - x_2)^2 - (1 - x_1)^2$	$X(1,1)$ $F = 3905.93$
2	Goldstein and Price	$[-2,2]$	$\min F = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$F = 3$
3	Branin	$[-5,10]$	$\min F = a(x_2 - bx_1^2 + cx_1 - d)^2 + e(1 - f)\cos(x_1) + e \quad a = 1,\ b = \frac{5.1}{4}\left(\frac{7}{22}\right)^2,\ c = \frac{5}{22} \times 7,\ d = 6,\ e = 10,\ f = \frac{1}{8} \times \frac{7}{22}$	$X(-22/7,12.275)$ $X(22/7,2.275)$ $X(66/7,2.475)$ $F = 0.3977272$
4	Martin and Gaddy	$[0,10]$	$\min F = (x_1 - x_2)^2 + \left[(x_1 + x_2 - 10)/3\right]^2$	$X(5,5)$ $F = 0$
5	Rosenbrock	(a) $[-1.2,1.2]$(b) $[-10,10]$	$\min F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$X(1,1)$ $F = 0$
6	Rosenbrock	$[-1.2,1.2]$	$\min F = \sum_{i=1}^{3}\left\{100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2\right\}$	$X(1,1,1,1)$ $F = 0$
7	Hyper sphere	$[-5.12,5.12]$	$\min F = \sum_{i=1}^{6} x_i^2$	$X(0,0,0,0,0,0)$ $F = 0$

Table 2
Parameters used for optimization of benchmark functions presented in Table 1.

Func. no.	N_g	N_q	No. of iterations	L_e	R_t	R_{rd}	$m_{max} : m_{min}$	T_w
1	1	15	80	2	1	400	0.1:0.1	0.45
2	2	5	120	1.5	0.9	200	0.2:0.2	0.7
3	2	5	100	1.5	0.9	300	0.1:0.1	0.7
4	1	5	80	1.5	0.9	200	0.1:0.1	0.6
5a	1	10	150	2	1	250	0.1:0.1	0.35
5b	2	10	250	1.5	0.9	350	0.3:0	0.6
6	2	25	2000	2	1.5	3000	0.05:0	0.7
7	1	7	100	2	1	1500	0.3:0.1	0.6

Table 3
Results from optimization of benchmark functions presented in Table 1. Results for all methods except GEM were directly driven from [9].

Func. no.	SIMPSA		NE-SIMPSA		GA		ANTS		Bee Colony		GEM	
	Succ %	Mean no. of evals	Succ %	Mean no. of evals	Succ %	Mean no. of evals	Succ %	Mean no. of evals	Succ %	Mean no. of evals	Succ %	Mean no. of evals
1	****	****	****	****	100	10,160	100	6000	100	868	100	746
2	****	****	****	****	100	5662	100	5330	100	999	100	701
3	****	****	****	****	100	7325	100	1936	100	1657	100	689
4	****	****	****	****	100	2488	100	1688	100	526	100	258
5a	100	10,780	100	4508	100	10,212	100	6842	100	631	100	572
5b	100	12,500	100	5007	****	****	100	7505	100	2306	100	2289
6	99	21,177	94	3053	****	****	100	8471	100	28,529	100	82,188
7	****	****	****	****	100	15,468	100	22,050	100	7113	100	423

**** Results not available.

Table 4
Schwefel and Rastrigin functions.

No	Function name	Interval	Function	Global optimum	
1	Schwefel	[−500,500]	$f(\vec{x}) = -\sum_{i=1}^{6} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$X_{min} = 420.9 \times (1,1,1,1,1,1) F_{min} = -2513.9$	
2	Rastrigin	[−5.12,5.12]	$f(\vec{x}) = \sum_{i=1}^{50} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$X_{min} = (\vec{0})$	$F_{min} = 0$

high-fitted local minima with a probability which strongly depends on the fitness of that local minimum.

4.1. Finding the global minimum among many local minima

One of the special ability of GEM is finding the global minimum of multimodal functions without being trapped in local minima. To show this ability, two optimization benchmark functions are optimized by using this algorithm. These multimodal functions are the Schwefel function with 6 independent variables [9] and the Rastrigin function with 50 independent variables [5]. Table 4 presents the specifications of these functions. Fig. 3 illustrates the multimodality of these two functions when there are only 2 independent variables. As these figures show, the global minimum is surrounded among many local minima, even for the two-dimensional mode. Table 5 presents the parameter values used for optimization of these functions by using GEM.

To obtain the convergence diagram, 20 independent runs were executed and the mean of the best individual values is calculated which represents the final result. Figs. 4 and 5 illustrate the convergence diagrams of the Schwefel and Rastrigin functions obtained by using Artificial Bee Colony Algorithm [5,9] and GEM. In optimization of the Rastrigin function by GEM, when the population size is 10, only in 16 runs the algorithm could locate the global minimum and hence for this population size the plotted diagram is the average of these 80% successful runs.

Fig. 4 shows that for the Schwefel function, GEM reaches the minimum value with a four-digit accuracy only after 28,600 function evaluations and with a five-digit accuracy only after 41,400

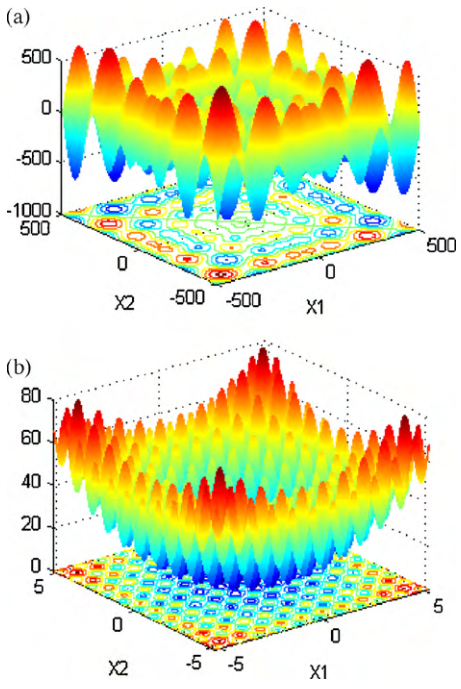

Fig. 3. Surface plot and contour lines for (a) the Schwefel function and (b) the Rastrigin function.

Table 5
Parameter selection used for optimization of the Schwefel and Rastrigin functions.

Function	N_g	N_q	No. of iterations	L_e	R_t	R_{rd}	m_{max}:m_{min}	T_w	No. of VARs
Schwefel	2	5	20,000	2	1.5	100	0.9:0.3	0.5	6
Rastrigin									
(a)	2	50	5000	2	2	1000	1.0:0.5	0.8	50
(b)	2	25	5000	2	2	1000	1.0:0.3	0.8	50
(c)	1	10	5000	2	1	500	1.0:0.7	0.6	50

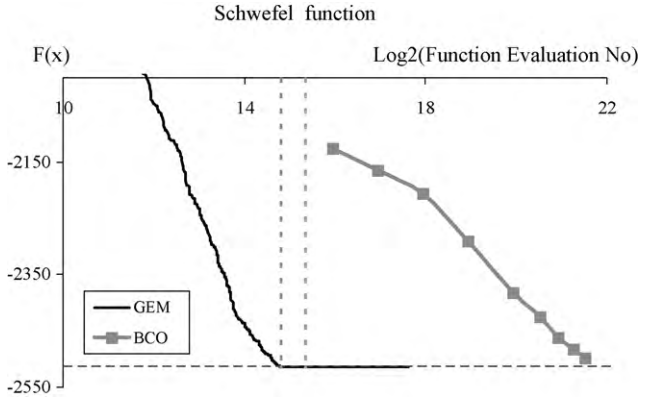

Fig. 4. Convergence diagram for the Schwefel function using Bee Colony Optimization (BCO) [9] and GEM.

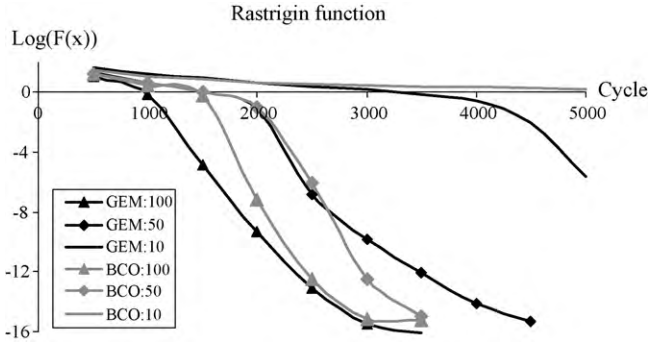

Fig. 5. Convergence diagram for the Rastrigin function using (a) Bee Colony Optimization (BCO) [5] and (b) GEM. For "GEM:10", the plotted diagram is the average of 80% successful runs.

function evaluations, while Bee Colony cannot find the global minimum value with a four-digit accuracy even after 2.8 million function evaluations.

Fig. 5 shows that for the Rastrigin function, Both Algorithms can find the global minimum value accurately when the population size is 50 or 100. BCO has problem finding the global minimum region when the population size is 10, but GEM can still locate the Global minimum and its convergence trend shows it can find the global minimum value accurately if the number of iterations is raised.

4.2. Finding all global minima of functions with multiple global minima

As it was mentioned before, the ability of finding multiple global minima if available and local minima with high fitness can be counted as an advantage for an optimization algorithm. To verify this ability for GEM, four benchmark functions having several global minima are optimized by using this method. Table 6 presents these functions and their specifications (Figs. 6 and 7). Parameter selection for these functions is shown in Table 7 and the obtained results which are the outcome of 20 independent runs for each function are presented in Table 8. The data in Table 8 includes the number of runs the algorithm could locate a global minimum and the average function value for that global minimum.

For the Six Hump and Himmelblau functions, GEM could locate all global minima in all runs. The average value for the objective function is coincident with the minimum value with at least an 11-digit accuracy. For the Himmelblau function, the difference between the minimum value and the final objective function value is less than 1e−10. For the Hansen function, each global minimum is found at least in 19 out of 20 runs and the average final value is coincident with the global minimum value with at least an 11-digit accuracy. For the Shubert function, each global minimum is found at least in 17 out of 20 runs and averagely the algorithm has located 96.7% of all global minima in an independent run. Again the

Table 6
Specifications of some multimodal functions having several global minima.

No.	Function name	Interval	Function	Global minimum
1	Six Hump Camel Back	$x_1 \in [-1.9, 1.9]$ $x_2 \in [-1.1, 1.1]$	$f(X) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2$	$X_{min}^{(1)} = \begin{bmatrix} 0.089842 & -0.712656 \end{bmatrix}$ $X_{min}^{(2)} = \begin{bmatrix} -0.089842 & 0.712656 \end{bmatrix}$ $f_{min} = -1.031628453$
2	Himmelblau	$x_1 \in [-6, 6]$ $x_2 \in [-6, 6]$	$f(X) = \left(x_1^2 + x_2 + -11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$	$X_{min}^{(1)} = \begin{bmatrix} 3 & 2 \end{bmatrix}$ $X_{min}^{(2)} = \begin{bmatrix} -2.805118 & 3.131312 \end{bmatrix}$ $X_{min}^{(3)} = \begin{bmatrix} -3.779310 & -3.283185 \end{bmatrix}$ $X_{min}^{(4)} = \begin{bmatrix} 3.584428 & -1.8488126 \end{bmatrix}$ $f_{min} = 0$
3	Hansen	$x_1 \in [-10, 10]$ $x_2 \in [-10, 10]$	$f(X) = \left(\sum_{i=1}^{5} i\cos((i-1)x_1 + i)\right) \times \left(\sum_{j=1}^{5} j\cos((j+1)x_1 + j)\right)$	See Fig. 6 $f_{min} = -176.54179313$
4	Shubert	$x_1 \in [-10, 10]$ $x_2 \in [-10, 10]$	$f(X) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right) \times \left(\sum_{j=1}^{5} j\cos((j+1)x_1 + j)\right)$	See Fig. 7 $f_{min} = -186.73090883$

Table 7
Parameter selection for optimization of functions presented in Table 6.

Function	N_g	N_q	No. of iterations	L_e	R_t	R_{rd}	m_{max}:m_{min}	T_w	Total no. of function evaluations
Six Hump Camel Back	3	4	200	2	1	800	0.9:0.2	0.7	2400
Himmelblau	5	6	200	2	0.8	3000	0.9:0.0	0.7	6000
Hansen	15	5	800	1.5	0.3	120	0.9:0.3	0.5	60,000
Shubert	25	5	800	1.5	0.2	120	0.9:0.4	0.5	100,000

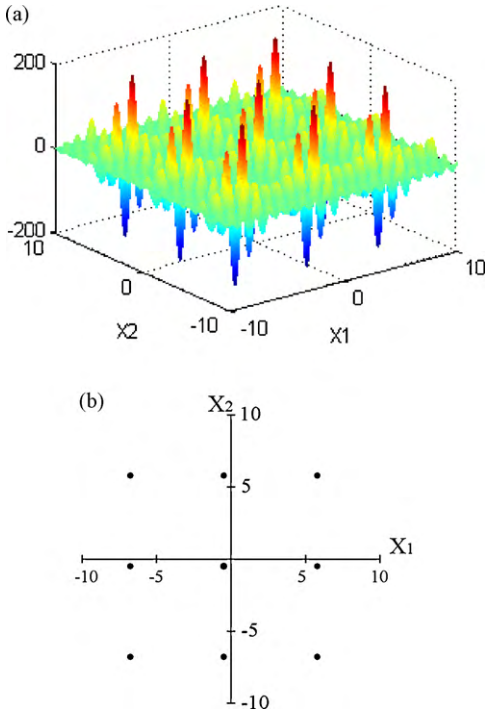

Fig. 6. Hansen function: (a) 3D plot and (b) uniformly distributed global minima.

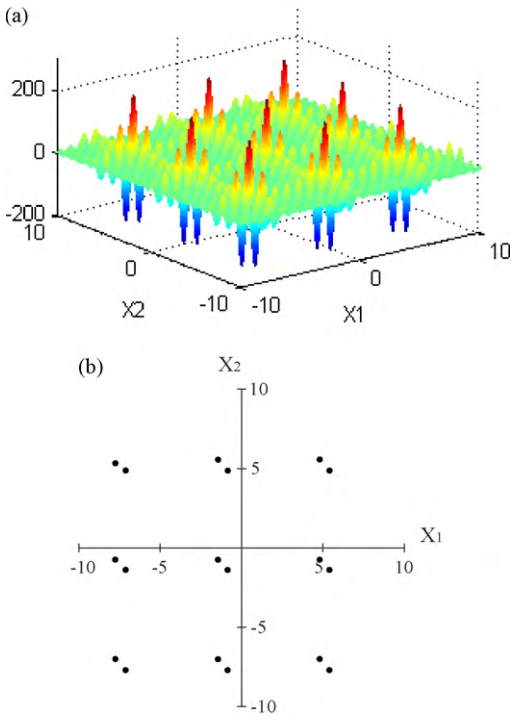

Fig. 7. Two-dimensional Shubert function: (a) 3D plot and (b) uniformly distributed global minima.

final value of the objective function is averagely coincident with the global minimum value with an 11-digit accuracy.

GEM can locate all global minima even when they are crowding in a small region of the search space. To demonstrate this ability of GEM, a modified type of the Hansen function is defined in which the search space is extended to $[-10, 90]^2$ and a positive term is

Table 8
Optimization results obtained from GEM for multimodal functions presented in Table 6.

Global min. no.	Six Hump Camel Back function		Himmelblau function		Hanen function[a]		Shubert function[a]	
	Succ. times	Average final value	Succ. times	Average final value	Succ. times	Average final value	Succ. times	Average final value
1	20	−1.031628453	20	6.40E−11	20	−176.54179313	19	−186.73090883
2	20	−1.031628453	20	7.90E−12	20	−176.54179313	19	−186.73090883
3		–	20	2.80E−11	20	−176.54179313	18	−186.73090883
4		–	20	3.70E−12	19	−176.54179313	17	−186.73090883
5		–		–	19	−176.54179313	20	−186.73090883
6		–		–	19	−176.54179313	20	−186.73090883
7		–		–	20	−176.54179313	19	−186.73090883
8		–		–	20	−176.54179313	20	−186.73090883
9		–		–	20	−176.54179313	20	−186.73090883
10		–		–		–	19	−186.73090883
11		–		–		–	19	−186.73090883
12		–		–		–	19	−186.73090883
13		–		–		–	20	−186.73090883
14		–		–		–	20	−186.73090883
15		–		–		–	20	−186.73090883
16		–		–		–	20	−186.73090883
17		–		–		–	19	−186.73090883
18		–		–		–	20	−186.73090883

[a] Global minima are numbered in an ascending order respective to their x_1 values. When several points have similar x_1 values, they are numbered in an ascending order respective to their x_2 values.

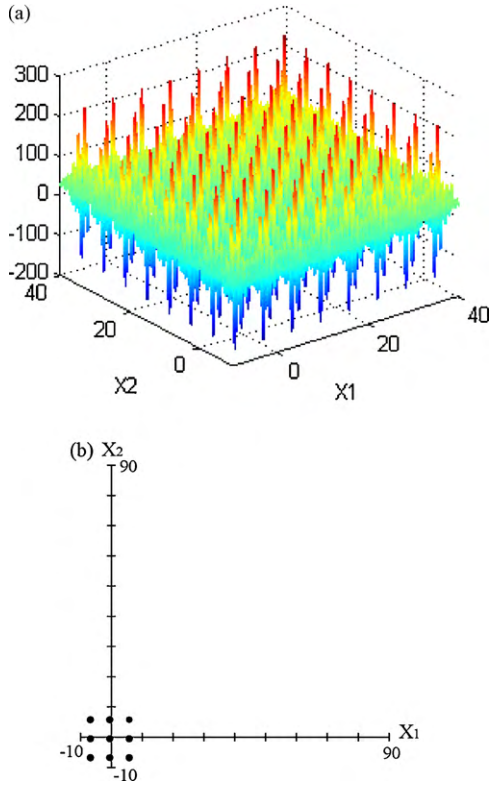

Fig. 8. Function $h(x)$: (a) surface plot in $[-10,40]^2$ and (b) non-uniformly distributed global minima.

added for points outside $[-10, 10]^2$. Hence:

$$h(x) = \begin{cases} f(x) & x_1 \in [-10, 10] \quad \text{and} \quad x_2 \in [-10, 10] \\ f(x) + |x_1 - 10| & x_1 \notin [-10, 10] \quad \text{and} \quad x_2 \in [-10, 10] \\ f(x) + |x_2 - 10| & x_1 \in [-10, 10] \quad \text{and} \quad x_2 \notin [-10, 10] \\ f(x) + |x_1 - 10| + |x_2 - 10| & x_1 \notin [-10, 10] \quad \text{and} \quad x_2 \notin [-10, 10] \end{cases}$$

where

$f(x) \equiv$ *Hansen function*

Fig. 8 depicts a 3D plot of function $h(x)$ in $[-10, 40]^2$ and the location of its global minima in the extended search space. It is seen that most of local minima outside $[-10, 10]^2$ remain and the average value of this function increases gradually while getting away from $[-10, 10]^2$. Consequently, the number of local minima especially those whose values are close to the global minimum value increases significantly.

Parameter selection for this function is: $N_g = 16$, $N_q = 12$, Number of iterations = 1000, $L_e = 2$, $R_t = 0.3$, $R_{rd} = 1000$, $m_{\max} = 1$, $m_{\min} = 0.8$, $T_w = 0.15$. Hence the number of function evaluations is: 192,000.

Table 9 presents the results obtained by GEM for 20 independent runs. Each point has been found at least in 17 out of 20 runs. The algorithm has located averagely 93.9% of total global minima in each run. The average final value of the objective function is coincident with the global minimum value with at least an 11-digit accuracy.

4.3. Finding local minima with high fitness

As it was mentioned in Section 1, finding local minima in which the value of the objective function is close to the global minimum value can be considered as an advantage for an industrial optimization problem. This advantage is more desirous when it presents

Table 9
Results obtained by optimization of the extended Hansen function.

Global Min. No.	Extended Hanen function	
	Succ. times	Average final value
1	20	−176.54179313
2	18	−176.54179313
3	20	−176.54179313
4	19	−176.54179313
5	18	−176.54179313
6	17	−176.54179313
7	20	−176.54179313
8	18	−176.54179313
9	19	−176.54179313

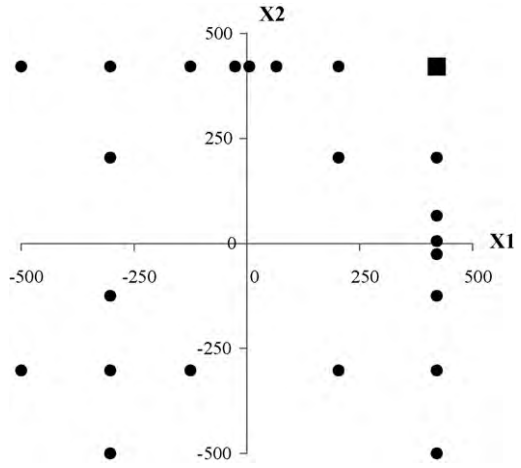

Fig. 9. Minima of the 2D Schwefel function in which the values of the function is less than −400.

entirely different solutions, which means the local minima are far apart the global minimum. To show the proposed algorithm's ability in finding these local minima, the two-dimensional Schwefel function which has many local minima is optimized by GEM. For this function:

$$f(\vec{x}) = -\sum_{i=1}^{2} x_i \sin\left(\sqrt{|x_i|}\right), \qquad x_1, x_2 \in [-500, 500]$$

$$\vec{x}_{\min} = 420.9688 \times [1, 1], \qquad f_{\min} = -837.965775$$

Fig. 9 depicts the location of the minima[1] in which the values of the objective function is less than −400. Parameter selection for optimization of this function is: $N_g = 9$, $N_q = 5$, Number of iterations = 800, $L_e = 1.5$, $R_t = 0.5$, $R_{rd} = 50$, $m_{\max} = .8$, $m_{\min} = 0$, $T_w = 0.6$. Thus the number of total function evaluations is 36,000.

The obtained results which are presented in Table 10 are outcome of 20 independent runs. The algorithm is successful in locating the global minimum (minimum number 1) in all runs. An arbitrary local minimum in which the value of the function is less than −420 is found with a probability which strongly depends on the goodness of the local minimum value. This shows that an appropriate parameter selection make this algorithm capable of finding good local minima besides the global minimum.

[1] The local minima are found by using the Exhaustive Search Method to find the approximate location of minima and running a pattern search for each near-minimum point to find the exact location.

Table 10
Results from optimization of the 2D Schwefel function.

Min. no.	Exact func. value at the minimum location	Succ. times	Average obtained value	Min. no.	Exact func. value at the minimum location	Succ. times	Average obtained value
1	−837.965775	20	−837.965774	11	−502.387771	9	−502.387770
2	−719.527440	20	−719.527439	12	−502.387771	8	−502.387770
3	−719.527440	20	−719.527439	13	−482.617869	0	−482.617869
4	−620.826105	20	−620.826105	14	−482.617869	0	−482.617869
5	−620.826105	19	−620.826105	15[a]	−481.133711	5	−481.133711
6	−601.089105	20	−601.089105	16[a]	−481.133711	2	−481.133711
7[a]	−599.572046	12	−599.572046	17	−443.065847	0	−443.065847
8[a]	−599.572046	6	−599.572046	18	−443.065847	0	−443.065847
9	−541.859061	9	−541.859060	19	−423.420726	1	−423.420726
10	−541.859061	7	−541.859061	20	−423.420726	1	−423.420726

[a] Boundary minimum.

5. Conclusion

Comparison of results obtained from optimization of several benchmark functions using GEM and some other EAs revealed that GEM can usually find the global minimum location faster. Furthermore, this algorithm is able to find the global minimum of multimodal functions without being trapped in local minima.

Another ability of this algorithm is that it can find quite all global minima of functions with multiple global minima. Additionally, this algorithm can locate good local minima with a probability which is strongly dependent on the goodness of minima.

6. Further research

Although GEM shows potentiality to be a comprehensive global optimization algorithm, it can be improved. The reduction pattern of the grenade's territory radius is just a simple suggestion which can be modified. Independent radius of territory for independent variables may be another solution. The suggestions for parameters selection are base on only a few empirical tests. Reducing the number of parameters used in this algorithm can make it a more user-friend algorithm.

References

[1] J.F. Goncalves, J.J.M. Mendes, M.G.C. Resende, A genetic algorithm for the resource constrained multi-project scheduling problem, European Journal of Operational Research 189 (2008) 1171–1190.
[2] Y. Hani, L. Amodeo, F. Yalaoui, H. Chen, Ant colony optimization for solving an industrial layout problem, European Journal of Operational Research 183 (2007) 633–642.
[3] S.M. Hashemi, A. Moradi, M. Rezapour, An ACO algorithm to design UMTS access network using divided and conquer techniques, Engineering Applications of Artificial Intelligence 21 (2008) 931–940.
[4] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.
[5] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (2008) 687–697.
[6] W.H. Liao, Y. Kao, C.M. Fan, Data aggregation in wireless sensor networks using ant colony algorithm, Journal of Network and Computer Applications 31 (2008) 387–401.
[7] J. Ombach, Stability of evolutionary algorithms, Journal of Mathematical Analysis 342 (2008) 326–333.
[8] R.E. Perez, K. Behdinan, Particle swarm approach for structural design optimization, Computers and Structures 85 (2007) 1579–1588.
[9] D.T. Pham, A. Ghanbarzade, E. Koc, S. Otri, S. Rahim, M.Zaidi, The bees algorithm—A novel tool for complex optimization problems, IPROMS 2006, Cardiff, England, pp. 454–461.
[10] E. Rodriguez-Tello, J.K. Hao, J. Torres-Jimenez, An improved simulated annealing algorithm for bandwidth minimization, European Journal of Operational Research 185 (2008) 1319–1335.
[11] S. Suresh, P.B. Sujit, A.K. Rao, Particle swarm optimization approach for multiobjective composite-beam design, Composite Structure 81 (2007) 598–605.
[12] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.
[13] T.H. Wu, C.C. Chang, S.H. Chung, A simulated annealing algorithm for manufacturing cell formation problems, Expert Systems with Applications 34 (2008) 1609–1617.
[14] Z. Wu, G. Ding, K. Wang, M. Fukaya, Application of a genetic algorithm to optimize the refrigerant circuit of fin-and tube heat exchangers for maximum heat transfer or shortest tube, International Journal of Thermal Science 47 (2008) 985–997.