International Journal of the Physical Sciences Vol. 6(25), pp. 5961-5981, 23 October, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.029
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Hybrid differential evolution and gravitation search algorithm for unconstrained optimization

Xiangtao Li*, Minghao Yin and Zhiqiang Ma

College of Computer Science, Northeast Normal University, Changchun, Jilin, P. R. China.

Accepted 27 May, 2011

This paper proposed an algorithm called DE-GSA. The proposed algorithm incorporates both the concepts from Differential evolution algorithm (DE) and Gravitation search algorithm (GSA), updating particles not only by DE operators but also by GSA mechanisms. The proposed algorithm is tested on several benchmark functions including unimodal and multimodal test functions, multimodal test function with fix dimension, and some real life problems. Then, experimental results have shown that the proposed algorithm is both efficient and effective.

Key words: Gravitation search algorithm, differential evolution algorithm, differential evolution with gravitation search algorithm (DE-GSA), hybrid meta-heuristic, algorithm.

INTRODUCTION

Optimization problems play an important role on both industrial application fields and the scientific research world. During the past decade, we have viewed significant progresses on tackling optimization problems. Different kinds of classical techniques have been advanced to handle optimization problems, including branch-and-bound (Lawler and Wood, 1996), meta-heuristic (Glover and Kochenberger, 2003), dynamic programming (Bellman, 1952) and gradient-based methods (Snyman, 2004). Among them, Meta-heuristic based methods, such as simulated annealing algorithm (SA) (Suman, 2004), genetic algorithm (GA) (Horn et al., 1994; Reid, 1996), artificial immune system algorithm (AIS) (Kalinlia and Karabogab, 2005), particle swarm optimization algorithm (PSO) (Bergh and Engelbrecht, 2006; Clerc and Kennedy, 2002; Du and Li, 2008; Kennedy and Eberhart, 1995; Liu et al., 2007), ant colony algorithm (ACO) (Ahmed, 2005; Dorigo et al., 1996; Ellabib et al., 2007; Zhang and Li, 2007), differential evolution algorithm (DE) (Omran et al., 2005, 2006; Qin and Suganthan, 2005; Qian and Li, 2008), gravitation search algorithm (GSA) (Rashedi and Nezamabadi-pour, 2009) and estimation of distribution algorithm (EDA) (Zhang and Muhlenbein, 2004; Zhang and Sun, 2004), may be one of the most popular methods. Particularly, gravitation search algorithm (GSA) is a novel meta-heuristic algorithm introduced recently. The basic idea of GSA is based on the law of gravity and mass interaction. The individuals of the population in this algorithm can be regarded as different masses. Using the gravitational force, information is shared among the individuals to direct the search towards the best location in the search space. Compared with other meta-heuristic methods like PSO, RGA and CFO, GSA usually provides better results.

During the last decade, meta-heuristic methods have been proved to have superior features to other traditional methods, and have been widely applied in the optimization field. However, these meta-heuristic methods often suffer some limitations. In some cases, they may be easy to fall into the local minimum or converge too slowly. Recently, researchers have found that a skilled combination of two meta-heuristic techniques can improve the performance when dealing with real-world and large scale problems (Hendtlass, 2001; Kim et al., 2007; Talibi Bautouche, 2004). Many hybrid heuristic based optimization methods have been investigated in the past few years. Angeline (Angeline, 1998) proposed a new hybrid swarm integrating PSO with tournament selection method. Zhang and Xie (2003) introduced a hybrid particle swarm with differential operators. Shi and Liang (2005) introduced a novel

*Corresponding author. E-mail: ymh@nenu.edu.cn. Tel: +86-0431-84536338. Fax: +86-0431-84536338.

algorithm based on the VPGA and the particle swarm optimization. Omran et al. (2007) proposed a hybrid version combining with PSO and DE. The DE is used to mutate, for each particle, the attractor associated with the particle, defined as a weighted average of its position and best position. Zhang et al. (2009) proposed a novel algorithm for unconstrained optimization. This algorithm updates particles according to the DE operation and mechanism of PSO. Thangaraj et al. (2008) developed a new hybrid algorithm combining with PSO and DE. This proposed algorithm can preserve the strengths of both of the algorithms and so on. Qin (2005) and Omran et al. (2005) proposed a self-adaptive DE (SaDE) algorithm, in which both trail vector generation strategies and their association control parameter values are self-adaptive by learning from their previous experiences in generating promising solution. The experiment results show that the SaDE is more stable with the relatively small standard deviation, and higher success rates. Brest et al. (2006) proposed an efficient technique for adaptive control parameter setting associated with differential evolution. The experimental results show that the self-adaptive DE can obtain better solutions than other algorithms. Sun et al. (2004) proposed a new algorithm combining the DE algorithm and the EDA algorithm, which tried to guide its search toward a promising area by sampling new solution from a probability model. Liu and Lampinen (2005) introduced a new version of the differential evolution algorithm with control parameters-the fuzzy adaptive Differential evolution algorithm (FADE). The algorithm uses the fuzzy logic controllers to adapt the parameters. Ratnaweera et al. (2004) introduced a novel parameter automation strategy for the particle swarm algorithm and two further extensions to improve its performance after a predefined number of generations. Wang and Dang (2007) proposed a novel evolutionary algorithm (EA) for global optimization, an application of Latin squares leads to a new and effective crossover operator. Noman and Iba (2008) proposed a crossover-based adaptive local search (LS) operation for enhancing the performance of standard differential evolution (DE) algorithm. This algorithm presents a LS technique to solve this problem by adaptively adjusting the length of the search, using a hill-climbing heuristic. Zhang and Sanderson (2009) proposed a new differential evolution (DE) algorithm, JADE. The algorithm uses a new mutation strategy "DE/current-to-*p*best" with optional external archive and updating control parameters in an adaptive manner. Jasper et al. (2009) proposed an evolutionary algorithm, entitled "A multialgorithm genetically adaptive method for single objective optimization (AMALGAM-SO)". The algorithm shown that AMALGAM-SO obtained similar efficiencies as existing algorithms on relatively simple unimodal problems, but is superior for more complex higher dimensional multimodal optimization problems. Yao et al. (1999) proposed "fast EP" (FEP) which uses a Cauchy instead of Gaussian mutation as the primary search operator. Lee and Yao (2004) proposed an evolutionary programming algorithm using adaptive as well as nonadaptive Lévy mutations, which applied to multivariate functional optimization. However, this field of study is still in its early days, a large number of future researches are necessary in order to develop hybrid algorithm for optimization problems. Particularly, within our knowledge, there is almost no paper concerning a hybrid heuristic method combining GSA.

Therefore in this paper, we propose a hybrid meta-heuristic algorithm integrating differential evolution heuristic into gravitational search algorithm, as we called DE-GSA. Differential evolution algorithm is first proposed by Storn and Price (1997). This algorithm is a population-based heuristic evolutionary algorithm that is simple to be implemented and has little or no parameters to be tuned. One of the remarkable advantages of DE is that this algorithm can use mutation, cross, select operators to increase the population diversity. In this sense, DE algorithm can be viewed as a complement of GSA algorithm, which is well known for its ability of global search. Therefore, combining these two methods should be a reasonable approach. Specifically, in this paper, the proposed algorithm starts from the DE process, and uses GSA to improve the quality of solution for the global population. Both processes run alternatively until the algorithm meets the stopping criterion.

DIFFERENTIAL EVOLUTION ALGORITHM

Differential evolution (DE) is an evolutionary algorithm first introduced by Storn and Price (1997). Similar to other evolutionary algorithms particularly genetic algorithm, DE uses some evolutionary operators like selection recombination and mutation operators. Different from genetic algorithm, DE uses distance and direction information from current population to guide the search process. The crucial idea behind DE is a scheme for producing trial vectors according to the manipulation of target vector and difference vector. If the trail vector yields a lower fitness than a predetermined population member, the newly trail vector will be accepted and be compared in the following generation. Different kinds of strategies of DE have been proposed based on the target vector selected and the number of difference vectors used. In this paper, we use two strategies, DE/rand/1/bin and DE/best/2/bin, described as follows.

For each target vector $x_i(t)$, trail vector $v_i(t)$, i = 1, ..., NP, let N be the dimension of target vector, and G be the G generation. Two mutant vectors are generated in these two strategies respectively:

For DE/rand/1/bin

$$v_{i,G} = x_{a,G} + F(x_{b,G} - x_{c,G}) \quad (1)$$

For DE/best/2/bin

$$v_{i,G} = x_{best,G} + F(x_{a,G} + x_{b,G} - x_{c,G} - x_{d,G}) \quad (2)$$

Where $a, b, c, d \in [1, \cdots\cdots, NP]$ are randomly chosen integers, and $a \neq b \neq c \neq d \neq i$. F is the scaling factor controlling the amplification of the differential evolution.

The cross-over operator, implements a recombination of the trial vector and the parent vector to produce offspring. This operator is calculated as:

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, (rand_j[0,1] \leq CR) or (j = j_{rand}) \\ x_{j,i,G}, otherwise \end{cases} \quad (3)$$

Where $j = [1, \cdots, D]$; $rand_j \in [0,1]$; $j_{rand} = [1, \cdots, D]$ is the randomly chosen index, CR is the crossover rate $v_{j,i,G}$ is the difference vector of the jth particle in the ith dimension at the Gth iteration, and $u_{j,i,G}$ denotes the trail vector of the jth particle in the ith dimension at the Gth iteration. Selection operator is used to choose the next population between the trail population and the target population:

$$x_{i,G+1} = \begin{cases} u_{i,G}, f(u_{i,G}) < f(x_{i,G}) \\ x_{i,G}, otherwise \end{cases} \quad (4)$$

The standard differential evolution algorithm can be described as the followings:

Step 1: Randomized initialization population, initialize parameters CR, F. and set the current generation number G = 0.
Step 2: Evaluate fitness for every individual.
Step 3: According to the mutation and crossover operations, it can obtain a trail vector for each individual.
Step 4: Evaluate every trail vector.
Step 5: Selection, if the trail vector yields a lower fitness than a predetermined population member, the newly trail vector will be accepted.
Step 6: G = G + 1; Repeat Step 3 to Step 5 until the stop criteria are reached.

GRAVITATION SEARCH ALGORITHM

Gravitation search algorithm is a stochastic, population-based search method introduced by Rashedi and Hossein (2009). The mechanism of GSA got inspired by the law of Newtonian gravity: "In the universe, every particle attracts every other particle with a force, and the force is directly proportional to the product of their masses and inversely proportional to the square of the distance between them." A GSA algorithm maintains a population of individuals, where each individual represents a possible solution. For a D dimension space, the position of a particle can be represented as:

$$X_i = (x_i^1, x_i^2, \cdots, x_i^d, \cdots, x_i^D), i = 1, \cdots, NP.$$

Given a specific time step 't' and arbitrary two individuals 'i' and 'j', the gravity force acting on these individuals can be represented as:

$$F_{ij}^d(t) = G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t)) \quad (5)$$

where M_{aj} is the active gravitational mass, M_{pi} is the passive gravitational mass, $G(t)$ is gravitational constant at time t, ε is a small constant, and $R_{ij}(t)$ is the Euclidian distance between two particles:

$$R_{ij}(t) = \| X_i(t), X_j(t) \|_2 \quad (6)$$

The total force that acts on a given individual i in a dimension d is a randomly sum of dth components of the forces exerted from other agents:

$$F_i^d = \sum_{j=1, j \neq i}^{NP} rand_j F_{ij}^d(t) \quad (7)$$

where $rand_j \in [0,1]$ is a random number.

According to the law of motion, "the current velocity of any mass is equal to the sum of the fraction of its previous velocity and the variation in the velocity or acceleration of any mass is equal to the force acted on the system divided by mass of inertia". Consequently, the acceleration rate of the individual i at time t, and in the direction dth, denoted by $a_i^d(t)$, can be calculated by:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \quad (8)$$

Where M_{ii} is the inertial mass of individual i.

Furthermore, the position and velocity of the individual can be updated as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \quad (9)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (10)$$

Where $rand_i \in [0,1]$ is used to give a stochastic characteristic to the algorithm.

Assuming the equality of the gravitational and inertia mass, the value of mass are calculated using the map of fitness. We update by the following equation:

$$M_{ai} = M_{pi} = M_{ii} = M_i,\ i = 1, \cdots, NP$$
$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}$$
$$M_i(t) = \frac{m_i(t)}{\sum_{i=1}^{NP} m_j(t)} \qquad (11)$$

Where $fit_i(t)$ denotes the fitness value of the agent I at time t and $best(t)$, $worst(t)$ are defined as follow:

$$best(t) = \min_{j \in \{1,\cdots,NP\}} fit_i(t)$$
$$worst(t) = \max_{j \in \{1,\cdots,NP\}} fit_i(t) \qquad (12)$$

The followings are the standard version of this algorithm:

Step 1: Search space identification.
Step 2: Randomized initialization.
Step 3: Fitness evolution of agents.
Step 4: Update the G, best, worst of the population.
Step 5: Calculation of the total force in different directions.
Step 6: Calculate M and a for every agent.
Step 7: Updating agents' velocity and position.
Step 8: Repeat Step 3 to Step 7 until the stop criteria are reached.

PROPOSED DE-GSA ALGORITHM

Here, discusses the structure and relational of this hybrid algorithm. Figure 1 describes the framework of the proposed DE-GSA algorithm. As shown, there are mainly two strategies to update the agent in the proposed algorithm: the DE strategy and the GSA strategy. For the DE updating strategy, we have discussed in differential evolution algorithm. For the GSA updating strategy, we propose a new method to restrain the most and least bounds of the individuals for GSA.

The GSA algorithm assumes that the whole population should be in an isolated and finite space. During the searching process, if there are some individuals that will move out of bounds of the space, the original algorithm stops them on the boundary. In other words, the particle will be assigned a boundary value. The disadvantage is that if there are too many individuals on the boundary, and especially when there exists some local minimum on the boundary, the algorithm will lose its population diversity to some extent.

In order to tackle this problem, there are mainly two improvements made in the proposed GSA updating strategy. First, we restrict the speed of the particles during searching process. The intention is to avoid the individuals moving towards the boundary too fast.

Secondly, when the individual moves outside the boundary, we scatter them in a feasible region away from the boundary, instead of stopping them just on the boundary. The proposed GSA updating strategy is as following:

Algorithm UGSA (GSA updating strategy)

```
Input: $x_i$ [G], $v_i$ [G] ( $x_i$ [G] denotes the position of the ith agent at the Gth iteration, $v_i$ [G] presents the velocity of the ith agent at the Gth iteration)
Outpt: $x_i$ ' [G], $v_i$ ' [G] ( $x_i$ ' [G] denotes the position of the ith agent at the Gth iteration, $v_i$ ' [G] presents the velocity of the ith agent at the Gth iteration)
Step 1: Generate $x_i$ [G], $v_i$ [G] using Equation (9) and (10).
Step 2: Tackle the boundary of the position and velocity.
If ( $x_i$ [G]> $x_{\max}$ ) then
    $x_i'[G] = x_{\max} - c * rand() * x_{\max}$
End if
If ( $x_i$ [G] < $x_{\min}$ ) then
    $x_i'[G] = x_{\min} + c * rand() * (-x_{\min})$
End if
If ( $v_i$ [G]> $v_{\max}$ ) then
    $v_i'[G] = v_{\max} - 2 * rand() * v_{\max}$
End if
If ( $v_i$ [G] < $v_{\min}$ ) then
    $v_i'[G] = v_{\min} + 2 * rand() * (-v_{\min})$
End if
```

Remark

c plays an important role in different test functions. If the value is too high, the individual will not be set in a feasible region. If it is too low, the algorithm will lose population diversity. Through a careful selection based on lots of experiments, c is set to be 0.01 in this paper. We set the $v_{\min}$ is equal to the $x_{\min}$ and the $v_{\max}$ is equal to the $x_{\max}$.

Based on the previous description, the algorithm of DE-GSA can be presented subsequently. In this proposed algorithm, DE process is first called. Then after the mutation operation and cross operation, if DE cannot generate a better solution, GSA is activated to update the current population. This process runs alternatively until the stopping criterion meets.

Algorithm DE-GSA

```
Input: *itermax* (maximal number of generations)
*NP* (population size)
*D* (the dimension of the agent)
Output: *bestvalue () (the best optimal solution)*

Step 1: Initialization.
Generate an initial population P with NP agents.
```

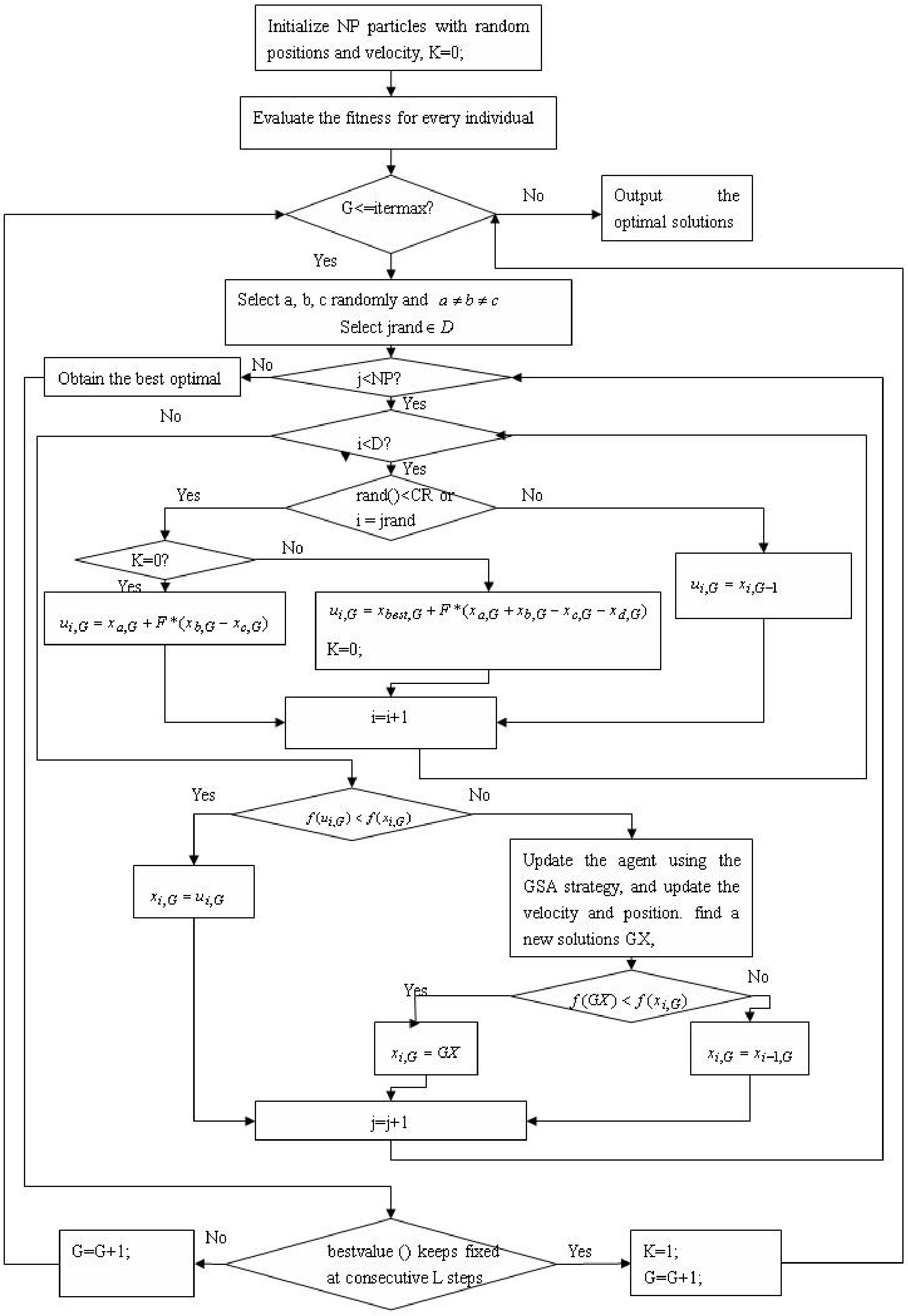
Initialize NP particles with random positions and velocity, K=0;
Evaluate the fitness for every individual
G<=itermax?
No
Output the optimal solutions
Yes
Select a, b, c randomly and $a \neq b \neq c$
Select jrand $\in D$
j<NP?
No
Obtain the best optimal
Yes
i<D?
No
Yes
rand()<CR or i = jrand
Yes
No
K=0?
No
Yes
$u_{i,G} = x_{a,G} + F*(x_{b,G} - x_{c,G})$
$u_{i,G} = x_{best,G} + F*(x_{a,G} + x_{b,G} - x_{c,G} - x_{d,G})$
K=0;
$u_{i,G} = x_{i,G-1}$
i=i+1
Yes
No
$f(u_{i,G}) < f(x_{i,G})$
$x_{i,G} = u_{i,G}$
Update the agent using the GSA strategy, and update the velocity and position. find a new solutions GX,
$f(GX) < f(x_{i,G})$
Yes
No
$x_{i,G} = GX$
$x_{i,G} = x_{i-1,G}$
j=j+1
bestvalue () keeps fixed at consecutive L steps
No
Yes
G=G+1;
K=1;
G=G+1;

Figure 1. The framework of the proposed DE-GSA algorithm.

```
Evaluate the fitness for each agent value (). Set the best position for the bestvalue ().Set the current iteration number G = 0.
Step 2: stopping criterion
If (G==itermax) then
    Output the optimal solutions bestvalue () and stop.
End if
Step 3: mutation and crossover.
Select a, b, c randomly and $a \neq b \neq c$
Select jrand$\in D$
For i = 1 to D do
    if (rand()<CR or i = jrand) then
        $u_{i,G} = x_{a,G} + F*(x_{b,G} - x_{c,G})$
    End if
End for
Step 4: Selection.
Evaluate the fitness for each trail vector for each dimension. Set the fitness for the tempvalue ().
For j=1 to NP do
    If (tempvalue (j) < value (j)) then
        $x_{i,G} = u_{i,G}$
    Else
        GSA active. We can use GSA to find a new solutions, this solution is GX.
    End if
    Evaluate the fitness of GX for GXvalue ().
    If (GXvalue (j) <value (j)) then
        $x_{i,G} = GX$
    Else
        $x_{i,G} = x_{i-1,G}$
    End if
End for
Step 5: Let the best optimal solution to the bestvalue().
If (bestvalue () keeps fixed at consecutive L steps) then
    $u_{i,G} = x_{best,G} + F(x_{a,G} + x_{b,G} - x_{c,G} - x_{d,G})$
End if
Step6: G=G+1; goto Step2.
```

Remark

An important problem that needs to be intentioned is the value of the CR, $CR \in (0,1)$. CR is the crossover rate which affects the diversity of population for the next generation. In our knowledge, hardly a good choice of the crossover rate has been proposed in the literature of DE. If the value of CR is too large, it is conducive to local search and speeds up the convergence rate. However, if the value of CR is too small, it is conducive to the population diversity and the ability of the global search. In our paper, CR is set to be 0.1. This value can enhance the population diversity as well as the convergence rate. The scaling factor F ($F \in (0,2)$) is also important because it affects the differential variation between two individual. In this paper, the value of F is 0.5, for the functions of Table 1. Additionally, for the functions of Table 2, we will assign different values of F for the different functions. For the proposed DE-GSA algorithm, the value of the L is also quite significant, since it is crucial for the proposed algorithm to obtain a optimal solution. If we set up the value too large, the effect of the GSA will be reduced. On the contrary, if the value of L is too little, it will affect the role of DE. Therefore, in this paper, the value of L is set to 15; this value can enhance the effect of both algorithms. The last problem is the G of the GSA algorithm, G is set using the equation as follows:

$$G(t) = G_0 * e^{-\alpha \frac{t}{T}},$$

where G0 is equal to 100, t denotes the current iteration, T represents the maximum iteration. The value of α is also important for it affects the effect of the GSA algorithm, if we set α too high, the stability of the algorithm will become very poor. On the other hand, small value will make the convergence rate slowly. Therefore, we set α to 20.

EXPERIMENTAL RESULTS

To evaluate the performance of our algorithm, we applied it to 20 standards benchmark functions and 2 real life problems. The first five functions are unimodal functions, and the following five functions are multimodal test functions. These ten test functions can be seen in Table 1. Then, ten multimodal test functions with fix dimension are used in our experimental study. Table 3 has shown the details of these functions. We also use two real life problems, gas transmission compressor design and optimal capacity of gas production facilities, to validate the proposed algorithm. So far, these problems have been widely used as benchmarks for study with different methods by many researchers.

The algorithm is coded in MATLAB 7.0, and experiments are made on a Pentium 3.0 GHz Processor with 1.0 GB of memory.

For unimodal and multimodal test functions, we will do three kinds of experiments for F1-F10:

(1) Population size = 20, dimension = 20, run = 30, itermax = 1000
(2) Population size = 40, dimension = 40, run = 20, itermax = 2000
(3) Population size = 80, dimension = 80, run = 15, itermax = 4000

Here, run presents the number of times an algorithm is executed.
For multimodal test function with fix dimensioning, we will experiment for F11 to F20:
Population size = 60, run = 30, itermax = 500

The results of unimodal and multimodal high–dimension test functions

Here, the performance of DE-GSA is compared with other well-know algorithms based on the 10 functions. The results are listed in Table 1. We apply DE-GSA to

Table 1. Unimodal and multimodal high–dimension test functions.

Test function	Range	Optimum
$F_1(x)=\sum_{i=1}^{n} x_i^2$	[-5.12, 5.12]	0
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[-10, 10]	0
$F_3(X)=\sum_{i=1}^{n-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	[-30, 30]	0
$F_4(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	[-100, 100]	0
$F_5(X)=\sum_{i=1}^{n} ix_i^4+random[0,1]$	[-1.28,1.28]	0
$F_6(X)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	[-500,500]	-418.9829*n
$F_7(X)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	[-5.12, 5.12]	0
$F_8(X)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi))+20+e$	[-32, 32]	0
$F_9(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})+(y_n-1)^2]\}+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	[-50, 50]	0
$F_{10}(X)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	[-50, 50]	0

these minimization functions and compared the results with GA, PSO, DE, and GSA. For unimodal function F1 to F5, the objective of the algorithms is to improve their convergence rate, not just to obtain optimal solutions. The results are averaged at least 30 runs for 20 dimension, 20 runs for 40 dimension, 15 runs for 80 dimension and the averaged best-so-far solution are listed in Table 2. As shown in Table 2, DE-GSA usually provides better solutions than other four algorithms expect for the test function F3. Moreover, the optimal convergence rate of the algorithms is shown in Figures 2, 3, and 4. From these figures, we can see that DE-GSA tends to find the optimal solutions faster than other algorithms.

Multimode functions are more difficult to be solved than unimodal functions, because they have many local minima. For these functions, obtaining the optimal solutions is more important because it can reflect the ability of the algorithms to escape from the local minima. As shown in Table 3, DE-GSA can provide better results than other algorithms for F6 to F9. For F10, DE-GSA cannot adjust itself and provide a better solution. Figures 5 and 6 have shown the optimal convergence rate of the algorithms. From both figures, we can find DE-GSA has a higher convergence rate and can generate better solutions than other algorithms.

Table 2. Minimization result of benchmark function in Table 1.

F	Dim	GA	PSO	DE	GSA	DE-GSA
F1	20	0.0862 (0.0651)	2.3921e-008 (1.458e-008)	1.1166e-016 (7.6700e-017)	1.9108e-016 (1.4131e-016)	7.2582e-018 (1.6864e-017)
	40	0.0750 (0.0152)	2.1431e-007 (1.154e-007)	2.4395e-020 (5.3728e-020	5.5061e-017 (2.0107e-017)	1.2268e-021 (1.4505e-021)
	80	0.0546 (0.0131)	1.6600e-006 (5.200e-005)	6.3583e-022 (5.3732e-022	2.3377e-017 (2.6958e-018)	4.4763e-025 (1.5334e-024)
F2	20	1.5415 (0.2228)	3.0915e-006 (2.2590e-006)	5.4799e-009 (1.6384e-009	7.5417e-008 (6.0403e-008)	1.4368e-009 (1.3166e-009)
	40	2.1713 (0.2493)	1.2187e-005 (3.6595e-005	2.5695e-011 (2.6668e-011	4.2746e-008 (4.5332e009)	8.2103e-012 (6.8438e-013)
	80	2.6224 (0.3626)	9.5727e-004 (3.0258e-003	2.1804e-012 (2.6707e-011)	3.9757e-008 (7.0558e-010)	1.2316e-012 (6.8541e-012)
F3	20	892.2516 (5.2561)	28.1872 (7.9226	47.7445 (19.4010)	57.2537 (93.6483)	24.6655 (3.1346)
	40	659.3939 (35.0295)	88.2824 (32.5186)	60.3302 (12.4921)	34.7175 (0.1729)	49.6038 (22.0396)
	80	1757.4654 (45.4340)	203.9393 (69.2797)	77.3435 (13.6890)	72.0420 (0.0439)	75.848 (4.218)
F4	20	23.7364 (10.2218)	2.9861e-010 (2.4076e-010)	7.5301e-014 (6.3810e-014)	2.0798e-013 (1.1161e-013)	1.1925e-014 (1.6710e-014)
	40	23.1347 (4.5841)	6.9759e-009 (1.200e-008)	7.8294e-018 (4.1231e-018)	4.9932e-017 (9.9231e-018)	1.6780e-018 (1.4098e-018)
	80	23.0706 (2.8393)	2.6924e-008 (2.6924e-008)	2.1394e-019 (4.1231e-019)	2.0936e-017 (1.0697e-019)	1.7443e-020 (7.5054e-021)
F5	20	0.1431 (0.0664)	0.0418 (0.0120)	0.0224 (0.0060)	0.6554 (1.5655)	0.0113 (0.0024)
	40	0.1307 (0.0518)	0.0908 (0.0268)	0.0288 (0.0075)	0.0539 (0.0147)	0.0232 (0.0070)
	80	0.1132 (0.0323)	1.5212 (0.0475)	0.0547 (0.0434)	0.0549 (0.0041)	0.0533 (0.0021)

The results of multimodal test function with fix dimension

Here, the performance of DE-GSA is compared with other algorithms based on multimodal test function with fix dimension listed in Table 4. A detailed description of these functions is shown in appendix A. For these functions, finding a better optimal solutions are more important, because it reflects the ability of the algorithms to locate near-global optimum of these functions. The averaged best-so-far solutions are listed in Table 5, from which we can see that DE-GSA performances better than the other algorithms. From Figures 7, 8 and 9, we can also find DE-GSA convergences faster than other algorithms on these functions.

The results of real life problem

The performance of DE-GSA is also compared with algorithm on some real life problems. In this paper, we use two real life problems to validate the performance of DE-GSA: Gas transmission compressor design problem and optimal capacity of gas production facilities problem. Gas transmission compressor design (Beightler and Phillips, 1976).

Table 3. Minimization result of benchmark function in Table 1.

F	Dim	GA	PSO	DE	GSA	DE-GSA
F6	20	-7.1444e+003 (225.1424)	-4.70750e+003 (4.8669e+002)	-8.3151e+003 (59.9792)	-1.9199e+003 (320.8131)	-8.1108e+003 (205.7557)
	40	-1.4285e+004 (334.7862)	-9.7576e+003 (1.1116e+003)	-1.6641e+004 (0.0000)	-2.9486e+003 (498.9667)	-1.6641e+004 (0.0000)
	80	-3.0304e+004 (177.9693)	-1.7004e+004 (1.8959e+003)	3.3479e+004 (68.4178)	-4.3034e+003 (751.4651)	-3.3519e+004 (0.0000)
F7	20	17.3415 (5.4070)	28.3329 (14.3352)	1.6160e-009 (1.4703e-009)	46.9666 (27.7972)	1.0622e-013 (3.4876e-013)
	40	18.0032 (3.0007)	82.7479 (34.5265)	2.3882e-005 (5.4991e-005)	29.6829 (6.9052)	3.3810e-013 (5.2729e-013)
	80	27.9166 (3.6881)	195.4091 (56.3481)	145.2627 (59.7323)	38.3059 (0.7035)	78.1390 (15.6996)
F8	20	3.0165 (0.0589)	0.0127 (0.0147)	1.0706e-007 (4.8870e-008)	1.2346e-008 (3.8539e-009)	9.5664e-009 (1.6838e-008)
	40	2.9816 (2.5227e-004)	0.0098 (0.0102)	6.9785e-010 (3.7265e-010)	5.5114e-009 (5.3301e-010)	1.5410e-010 (1.3703e-010)
	80	2.9812 (4.8999e-005)	0.0012 (0.0030)	8.3041e-011 (3.7348e-012)	2.1960e-009 (5.6204e-011)	1.4788e-012 (4.8736e-013)
F9	20	0.2972 (0.2080)	2.2397e-004 (3.4072e-004)	4.2327e-015 (3.0724e-015)	0.0104 (0.0402)	2.3564e-016 (9.1262e-016)
	40	0.1503 (0.1134)	0.3458 (0.8222)	2.1988e-018 (3.4942e-019)	0.0259 (0.0635)	1.1779e-032 (0.0000)
	80	0.0171 (8.6606e-004)	8.2411e-004 (8.2886e-004)	7.2401e-019 (2.4944e-019)	5.3640e-020 (4.6214e-021)	5.8895e-033 (0.0000)
F10	20	0.0212 (0.0167)	0.3063 (0.6793)	-12.1253 (0.0000)	5.5391e-032 (6.8897e-032)	-12.1253 (0.0000)
	40	0.0027 (6.4061e-004)	0.5000 (0.8927)	-12.1253 (0.0000)	4.4493e-032 (5.5703e-032)	-1.1504 (0.0000)
	80	3.6692e-004 (2.5446e-004)	42.7568 (47.6000)	-12.1253 (0.0000)	1.4998e-032 (1.2726e-032)	-1.1504 (0.0000)

Min $$f(x)=8.61\times10^5\times x_1^{1/2}x_2x_3^{-2/3}(x_2^2-1)^{-1/2}+3.69\times10^4\times x_3 +7.72\times10^8\times x_1^{-1}x_2^{0.219}-765.43\times10^6\times x_1^{-1}$$

Subject to: $10\le x_1\le 55$, $1.1\le x_2\le 2$, $10\le x_3\le 40$.

Optimal capacity of gas production facilities (Beightler and Phillips, 1976):

Min $$f(x)=61.8+5.72x_1+0.2623[(40-x_1)\ln(\frac{x_2}{200})]^{-0.85} +0.087(40-x_1)\ln(\frac{x_2}{200})+700.23x_2^{-0.75}$$

Subject to :

$x_1\ge 17.5$, $x_2\ge 200$, $17.5\le x_1\le 40$, $300\le x_2\le 600$

The experimental results of these real life problems are listed in Tables 6 and 7. From the tables we can draw the conclusion that DE-GSA usually generates better results.

Comparison of self-adaptive DE with fuzzy adaptive differential evolution algorithm

Liu and Lampinen (2005) introduce a new version of the differential evolution algorithm with control parameters

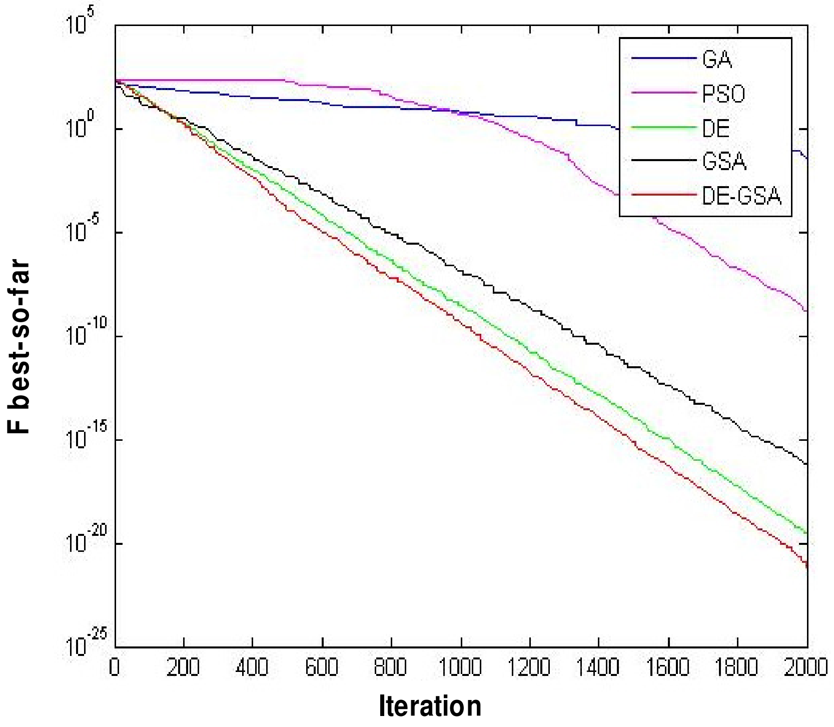

Figure 2. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F1 with NP = 40, D = 40.

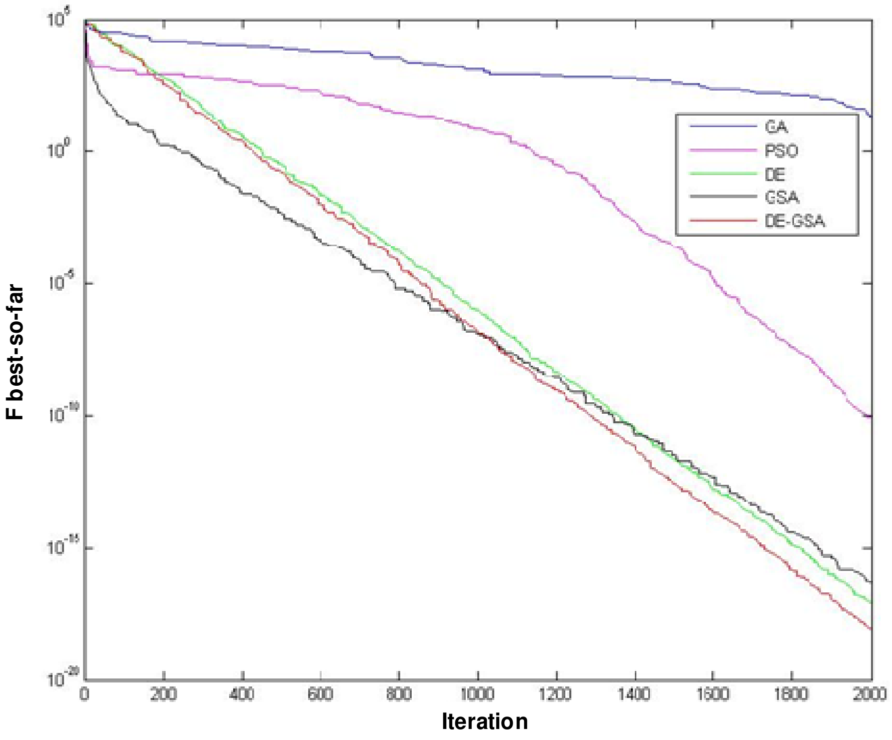

Figure 3. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F4 with NP = 40, D = 40.

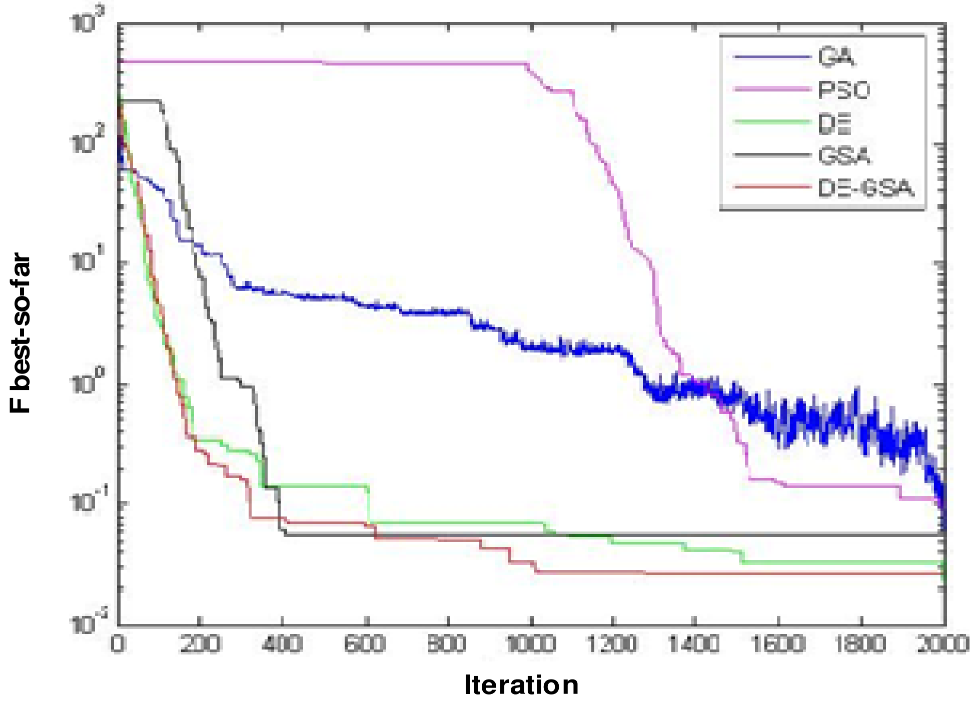

Figure 4. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F5 with NP = 40, D = 40.

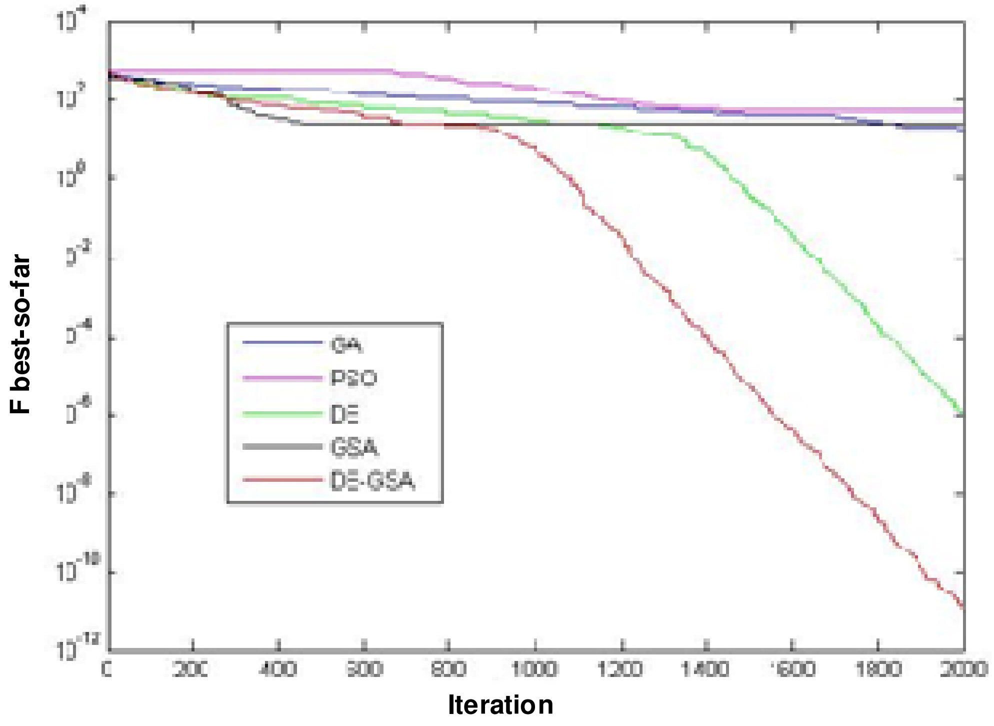

Figure 5. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F7 with NP = 40, D = 40.

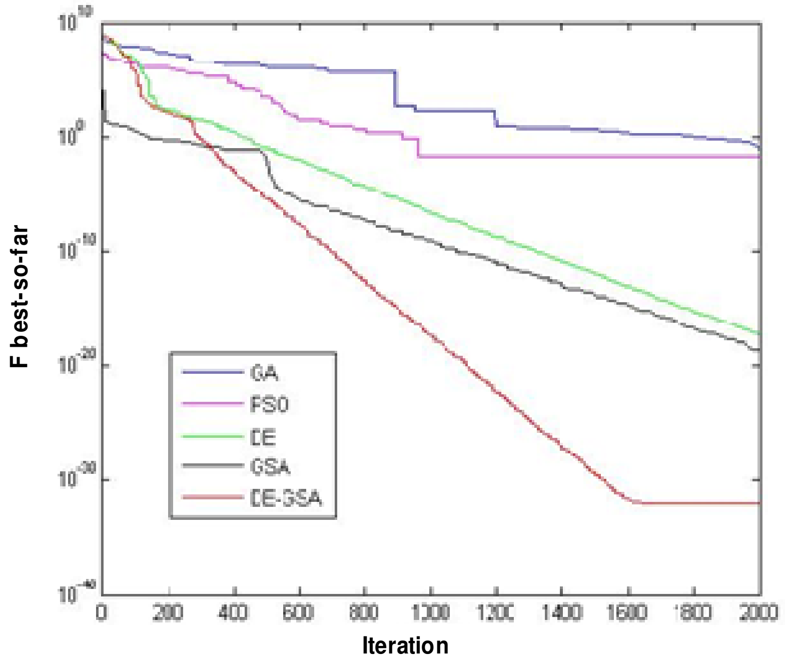

Figure 6. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F9 with NP = 40, D = 40.

Table 4. Multimodal test function with fix dimensioning.

Test function	Range	Dimension
$F_{11}(X)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i+a_{ij})^6})^{-1}$	[-65.53,65.53]	2
$F_{12}(X)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	[-5,5]	4
$F_{13}(X)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	[-5,5]	2
$F_{14}(X)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	[-5,10]*[0,15]	2
$F_{15}(X)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	[-5,5]	2
$F_{16}(X)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	[0,1]	3
$F_{17}(X)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	[0,1]	6
$F_{18}(X)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	[0,10]	4
$F_{19}(X)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	[0,10]	4
$F_{20}(X)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	[0,10]	4

Table 5. Minimization result of benchmark function in Table 4.

F	Dim	GA	PSO	DE	GSA	DE-GSA
F11	2	0.9980 (0.0000)	0.9980 (0.0000)	0.9980 (0.0000)	4.5297 (2.7184)	0.9980 (0.0000)
F12	4	0.0017 (3.6960e-004)	0.0046 (0.0012)	0.0014 (4.5493e-004)	0.0057 (0.0024)	7.1010e-004 (1.8186e-004)
F13	2	-1.0316 (0.0000)	-1.0316 (0.0000)	-1.0316 (0.0000)	-1.0316 (0.0000)	-1.0316 (0.0000)
F14	2	0.3981 (3.8806e-004)	0.3979 (0.0000)	0.3979 (0.0000)	0.3979 (0.0000)	0.3979 (0.0000)
F15	2	3.0036 (0.0028)	3.0000 (0.0000)	3.0222 (0.0014)	3.0000 (0.0000)	3.0000 (0.0000)
F16	3	-3.8620 (0.0018)	-3.8628 (0.0000)	-3.8628 (0.0000)	-3.7989 (0.1406)	-3.8628 (0.0000)
F17	6	-3.2744 (0.0651)	-3.2657 (0.5732)	-3.3220 (0.0000)	-2.4750 (0.7890)	-3.3220 (0.0000)
F18	4	-6.5630 (3.4866)	-5.0551 (0.0000)	-9.9104 (0.5209)	-5.0552 (0.0000)	-10.1532 (0.0000)
F19	4	-6.4789 (3.1670)	-5.0876 (2.1503e-007)	-10.1182 (0.3283)	-9.3399 (2.2007)	-10.4029 (0.0000)
F20	4	-10.5280 (0.0038)	-5.1284 (0.0000)	-10.2201 (0.3967)	-10.5364 (0.0000)	-10.5364 (0.0000)

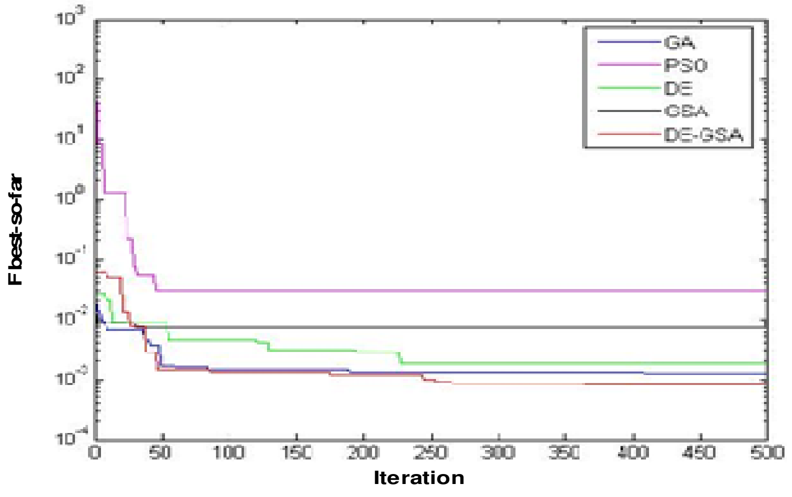

Figure 7. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F12 with NP = 40, D = 40.

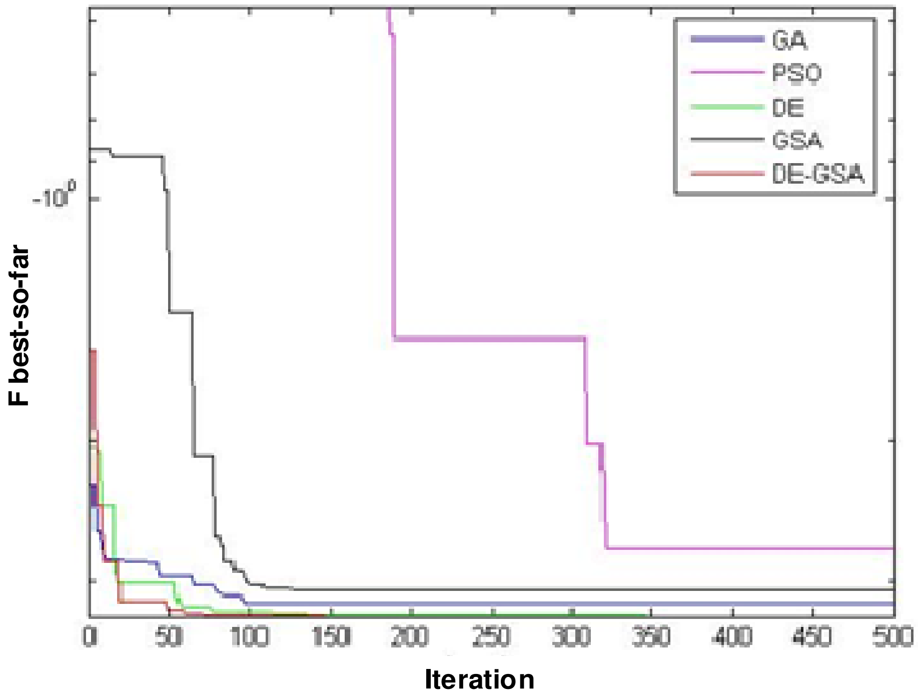

Figure 8. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F17 with NP = 40, D = 40.

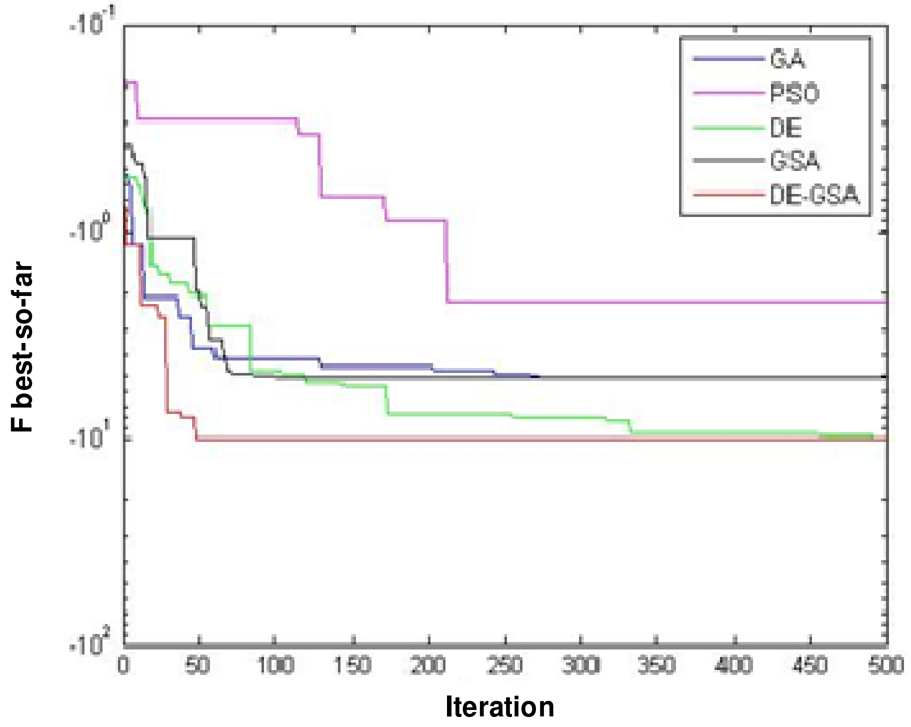

Figure 9. Comparison of performance of GA, PSO, DE, GSA, DE-GSA for minimization of F18 with NP = 40, D = 40.

Table 6. Gas transmission compressor design.

Item	DE	GSA	DE-GSA	(Beightler and Phillips, 1976)
x_1	52.3966	53.0547	53.5080	55
x_2	1.1875	1.1919	1.1901	1.195
x_3	24.6997	24.5070	24.7624	25.026
f(x)	2.96443e+006	2.96449e+006	2.96437e+006	2.96455e+006

Table 7. Gas transmission compressor design.

Item	DE	GSA	DE-GSA	(Beightler and Phillips, 1976)
x_1	17.5	17.5	17.5	17.5
x_2	600	600	600	465
f(x)	169.844	169.844	169.844	173.76

Table 8. Comparison of self-adaptive DE with fuzzy adaptive differential evolution algorithm.

F	#Gen	Fuzzy adaptive DE Mean best (Std Dev.)	DE-GSA Mean best (Std Dev.)
F1	5000	2.35e-10(2.97e-21)	1.0945e-43(1.1289e-43)
F3	7000	4.16e+1(1.82e-2)	4.441e+1 (6.125e-1)
F4	5000	0(0)	0(0)
F5	5000	1.9e+1(2.92e-1)	0(0)
F7	10000	2.58e+2(9.17e+1)	0(0)
F8	5000	5.9e-2(1.23e-6)	7.9936e-15(0)
F12	100	0.9980(2.5e-26)	0.9980(1.0368e-13)
F15	50	3.0001(3.35e-7)	3.000(0)

The fuzzy adaptive differential evolution algorithm. The algorithm uses the fuzzy logic controllers to adapt the parameters.

From the paper, the algorithm is tested with a set of standard test functions, where it outperforms the original DE when the dimensionality of the problem is high. The following parameters are the same with the paper (Liu and Lampinen, 2005): dimension = 50; population size = 10* dimension, itermax: 5000 for F1, F4, F5, F8 and 7000 for 5, F3, 10000 for F7, are listed on Table 8.

As shown in Table 8, The DE-GSA algorithm can perform better than the FADE algorithm for the functions expect the F3. Base on the experimental results, we can find that DE-GSA outperforms the FADE for the high dimension function. For lower dimension function, the DE-GSA algorithm has the similar solution with the FADE algorithm.

Comparison of FEP and CEP algorithms

Here, we will compare our algorithm with the FEP and CEP algorithms (Yao and Liu, 1999). We set the parameters as in Yao and Liu (1999), the following parameters are used in our paper: population size = 100, dimension: 30 for F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, 2 for F11, F13, F14, F15, 4 for F12, F18, F19, F20, 3 for F16, 6 for F17. itermax: 1500 for F1, F4, F8, F9, F10, 2000 for F2, 20000 for F3, 3000 for F5, 5000 for F7 and 100 for F11, F13, F14, F15, F18, F19, F20, 4000 for F12. Therefore, in our paper, we set the same population size and the same iteration as in Yao and Liu (1999), runs = 50 replications are conducted for each function. The averaged best-so-far solutions are listed in Table 9. Form Table 9, we can find the DE-GSA obtain better results than the FEP and CEP algorithm. For unimodal and

Table 9. Comparison of FEP and CEP algorithms.

F	#Gen	FEP	CEP	DE-GSA
		Mean best (Std Dev.)	Mean best (Std Dev.)	Mean best (Std Dev.)
F1	1500	5.7e-4(1.3e-4)	2.2e-4(5.9e-4)	1.1130e-17(8.6734e-18)
F2	2000	8.1e-3(7.7e-4)	2.6e-3(1.7e-4)	8.2379e-15(4.4905e-15)
F3	20000	5.06(5.87)	6.17(13.61)	0(0)
F4	1500	0(0)	577.76(1125.76)	1.5978e-17(9.2810e-18)
F5	3000	7.6e-3(2.6e-3)	1.8e-3(6.4e-3)	0(0)
F7	5000	4.6e-2(1.2e-2)	89.0(23.1)	0(0)
F8	1500	1.8e-2(2.1e-3)	9.2(2.8)	1.2150e-13(1.5632e-13)
F9	1500	9.2e-6(3.6e-6)	1.76(2.4)	9.3668e-10(4.0617e-10)
F10	1500	1.6e-4(7.3e-5)	1.4(3.7)	1.1930e-18(3.7725e-18)
F11	100	1.22(0.56)	1.66(1.19)	0.9980(1.0951e-12)
F12	4000	5.0e-4(3.2e-4)	4.7e-4(3.0e-4)	3.0749e-04(1.2122e-19)
F13	100	-1.03(4.9e-7)	-1.03(4.9e-7)	-1.0316(4.3205e-09)
F14	100	0.398(1.5e-7)	0.398(1.5e-7)	0.3979(2.9893e-06)
F15	100	3.02(0.11)	3.0(0)	3.000(0)
F18	100	-5.52(1.59)	-6.86(2.67)	-10.1532(4.6389e-5)
F19	100	-5.52(2.12)	-8.27(2.95)	-10.4029 (9.5130e-8)
F20	100	-6.57(3.14)	-9.10(2.92)	-10.5364(8.8497e-12)

multimodal function, the DE-GSA algorithm can gives better solution than FEP and CEP for all functions. Especially, for the multimodal function, the final results are more important because of this function can reflect the algorithm's ability to escape form poor local optima and obtain the near-global optimum.

Comparison of DEPSO and BBDE algorithms

Here, compares the performance of the DE-GSA with the DEPSO algorithm (Zhang and Xie, 2003) and BBDE algorithm (Omran et al., 2007). The DEPSO provides the bell-shaped mutations with consensus on the population diversity along with the evolution, while keeps the self-organized particle swarm dynamics. The DEPSO is shown to outperform the PSO and DE for a set of benchmark functions. The BBDE presents a new population-based algorithm, as a hybrid of the barebones particle swarm optimizer (PSO) and differential evolution (DE). The particle position update is changed to probabilistically base a new position on a mutation of the particle attractor, or a randomly selected best position. The BBDE does not make use of the standard PSO parameters and also removes the DE scale parameter. The only parameter is the probability of recombination, for which it was shown empirically that the BBDE is insensitive. In the experiment, we used the same function set and the parameter as in Omran et al. (2007), the results reported here are the average and the deviations for 30 independent runs. In Omran et al. (2007), 13 test functions were used, and 7 of them are the same as the benchmark function in Omran et al. (2007) and in our paper, population size is 30 and iteration is 100000 function evaluations. The experimental results are listed in Table 10. As shown in Table 10, the DE-GSA performs better than the DEPSO and BBDE. It can conclusion that the DE-GSA combines the DE algorithm and GSA is very meaningful. For F10, The DEGSA algorithm cannot give the optimal solution than other algorithm, but based on the result, the robustness of the DEGSA is very good.

Comparison of adaptive LEP and best lévy algorithm

Here, we will compare our algorithm with the adaptive LEP and best lévy algorithm (Lee and Yao, 2004). We set the parameters as in Lee and Yao (2004), the following parameters were used in our paper: population size = 100, dimension: 30 for F1, F3, F7, F8, F9, F10, 2 for F13, F15, 4 for F18, F19, F20. itermax: 1500 for F1, F3, F7, F8, F9, F10, 30 for F13, F15, 100 for F18, F19, F20. The experimental results are listed in Table 11, From Table 11, for the unimodal function F1 and F3, the DE-GSA can gives the better solution than adaptive LEP and best lévy algorithm. For multimodal functions F7 to F10 with many local minima, the final results are more important because of this function can reflect the algorithm's ability to escape form poor local optima and obtain the near-global optimum. The DE-GSA provided better solutions than other algorithms. For F13 and F15, the dimension of the function is very small. Therefore, all the algorithms

Table 10. Comparison of DEPSO and BBDE algorithms.

F	#Gen	DEPSO Mean Best (Std Dev.)	BBDE Mean Best (Std Dev.)	DE-GSA Mean best (Std Dev.)
F1	100000	0.339409e-10(0.255639e-10)	0(0)	0(0)
F3	100000	30.243866(3.11839)	14.295707(0.948028)	0.1511(0.0589)
F4	100000	0.692243e-9(0.278905e-9)	0(0)	0(0)
F5	100000	0.436603e-14(0.348727e-12)	0(0)	0(0)
F7	100000	40.970572(2.021865)	72.185823(3.018019)	0(0)
F8	100000	13.435463(0.550129)	2.136173(0.159471)	4.4409e-15(0)
F11	100000	0(0)	0(0)	0.9980(0)

Table 11. Comparison of adaptive LEP and best lévy algorithm.

F	#Gen	Adaptive LEP Mean best (Std dev)	Best levy Mean best (Std dev)	DE-GSA Mean best (Std dev)
F1	1500	6.32e-4(7.6e-5)	6.59e-4(6.4e-5)	1.1130e-17(8.6734e-18)
F3	1500	43.40(31.52)	57.75(41.60)	30.1137(5.8759)
F7	1500	5.85(2.07)	12.50(2.29)	1.2150e-13(1.5632e-13)
F8	1500	1.9e-2(1.0e-3)	3.1e-2(2.0e-3)	9.3668e-10(4.0617e-10)
F9	1500	6.0e-6(1.0e-6)	3.0e-5(4.0e-6)	1.1930e-18(3.7725e-18)
F10	1500	9.8e-5(1.2e-5)	2.6e-4(3.0e-5)	-7.4218(5.8663)
F13	30	-1.031(0.0)	-1.031(0.0)	-1.0316(4.2308e-07)
F15	30	3.000(0)	3.000(0)	3.0000(0)
F18	100	-9.54(1.69)	-9.95(0.99)	-10.1532(4.6389e-5)
F19	100	-10.30(0.74)	-10.40(1.0e-4)	-10.4029 (9.5130e-8)
F20	100	-10.54(4.9e-5)	-10.54(3.1e-3)	-10.5364(8.8497e-12)

find optimal solutions for these two functions. For F18 to F20, The DE-GSA can provide all the optimal solution. The algorithm performs superiorly better than adaptive LEP and best lévy algorithm.

Comparison of MPSO-TVAC and HPSO-TVAC algorithms

Ratnaweera et al. (2004) introduced a novel parameter automation strategy for the particle swarm algorithm and two further extensions to improve its performance after a predefined number of generations. Firstly, time-varying acceleration coefficients (TVAC) are introduced in addition to the time-varying inertia weight factor in particle swarm optimization (PSO) to efficiently control the local search and convergence to the global optimum solution, the concept of "mutation" is introduced to the particle swarm optimization along with TVAC (MPSO-TVAC), Secondly, we introduce a novel particle swarm concept "self-organizing hierarchical particle swarm optimizer with TVAC (HPSO-TVAC)." This algorithm selected modulus of the velocity vector of a random particle by predefined probability which added a small perturbation to it randomly. In the experiment, we used the same function set and the parameter as in Ratnaweera et al. (2004), the results reported here are the average and the deviations for 50 independent runs.

In Ratnaweera et al. (2004), 5 test functions were used, and 3 of them are the same as the benchmark function in Ratnaweera et al. (2004) and in our paper, the following parameters were used in our experimental: population size is 40, dimension: 10, 20, 30 for the functions. The iterations are listed in Table 12, all functions have the global optimal value of 0.0. The stopping criteria are setting to 0.01. For F1, all algorithms can convergence to 0.01 for the 50 runs. For F3, The DE-GSA can give better solution than the MPSO-TVAC and HPSO-TVAC for 10 and 20 dimensions. For 30 dimensions, the algorithm gives the worst solution. For F7, the DE-GSA algorithms can convergence to 0.01 for three different dimensions. The experimental results show that the DE-GSA performs superiorly better than the MPSO-TVAC and HPSO-TVAC.

Table 12. Comparison of MPSO-TVAC and HPSO-TVAC algorithms.

F	Dimension	#Gen	Mean best (Std dev)		
			MPSO-TVAC	**HPSO-TVAC**	**DE-GSA**
F1	10	1000	0.01	0.01	0.01
	20	2000	0.01	0.01	0.01
	30	3000	0.01	0.01	0.01
F3	10	3000	4.247(7.961)	12.967(11.538)	0.0222(0.0225)
	20	4000	17.7148(60. 306)	14.093(9.641)	3.0755(0.5737)
	30	5000	18.633(25.122)	13.666(11.006)	52.5691(28.2315)
F7	10	3000	0.01(0.0033)	0.01	0.01
	20	4000	0.3415(0.588)	0.01	0.01
	30	5000	2.050(1.910)	0.044(0.196)	0.01

CONCLUSION

This paper presented a hybrid differential evolution with gravitation search algorithm called DE-GSA. The proposed algorithm is tested on several benchmark functions including unimodal and multimodal test functions, multimodal test function with fix dimension, and some real life problems. We have compared the performance of DE-GSA with other evolution algorithm. Experimental results have shown that the proposed algorithm is more effective in obtaining better quality solution, which are more robust with the relatively smaller standard deviation.

ACKNOWLEDGEMENTS

This work is fully supported by the National Natural Science Foundation of China under Grant Nos. 60473042, 60573067, 60803102.

REFERENCES

Ahmed AA (2005). Feature Subset Selection Using Ant Colony Optimization. Int. J. Comput. Intell., 2:53-58.
Angeline PJ (1998). Using selection to improve particle swarm optimization, IEEE Inter. Conf. on Evol. Comput.. Anchorage, 5:84-89.
Beightler CS, Phillips DT (1976). Applied Geometric Programming, John Wiley and sons.
Bellman R (1952). On the theory of dynamic programming. Proceedings of the National Academy of Sciences. 38:716-719.
Bergh FVD, Engelbrecht AP (2006). A study of particle swarm optimization particle trajectories. Info. Sci., 176:937-971.
Brest J, Zumer V, Maucec MS (2006). If-adaptive differential evolution algorithm in constrained real-parameter optimization Proc IEEE Congr. Evol. Comput.. (Vancouver, BC), pp. 215-222.
Clerc M, Kennedy J (2002). The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. IEEE Trans. Evol. Comput., 6:58-73.
Dorigo M, Maniezzo V, Colorni A (1996). The ant system: optimization by a colony of cooperating agents. IEEE Trans. Systems, Man, Cybernetics - Part B, 26 (1):29-41.
Du W, Li B (2008). Multi-strategy ensemble particle swarm optimization for dynamic optimization. Infor. Sci., 178:3096-3109.
Ellabib I, Calamai P, Basir O (2007). Exchange strategies for multiple ant colony system. Infor. Sci., 177:1248-1264.
Glover F, Kochenberger G (2003). Handbook of Meta-heuristics, Kluwer, Boston.
Hendtlass T (2001). A Combined Swarm Differential Evolution Algorithm for Optimization Problems. In Proceedings of the Fourteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, In: Lecture Notes in Computer Science, Springer-Verlag, 2070:11-18.
Jasper AV, Bruce AR, James M (2009). Hyman Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces. IEEE Trans. Evol. Comput., 13 (2):243-259.
Kalinlia A, Karabogab N (2005). Artificial immune algorithm for IIR filter design. Engineering Applications of Artificial Intelligence, 18:919-929.
Kennedy J, Eberhart RC (1995). Particle Swarm Optimization, In Pro. of the IEEE Inter. Joint Conf. on Neu. Net., 4:1942-1948.
Kim DH, Abraham A, Cho JH (2007). A hybrid genetic algorithm and bacterial foraging approach for global optimization. Info. Sci., 177:3918-3937.
Lawler EL, Wood E (1966). Branch-and-bound methods: A survey. Operations Research, 14:699-719.
Lee CY, Yao X (2004). Evolutionary programming using mutations based on the levy probability distribution. IEEE Trans. Evol. Comput, 8(1): 1-13.
Liu H, Abraham A, Zhang W (2007). A Fuzzy Adaptive Turbulent Particle Swarm Optimization. Int. J. Innovative Comput. Appl., 1: 39-47.
Liu J, Lampinen J (2005). A fuzzy adaptive differential evolution algorithm. Soft Comput.: Fusion Found,. Methodologies Applicat., 9(6): 448-462.
Noman N, Iba H (2008). Accelerating Differential Evolution Using an Adaptive Local Search. IEEE Trans. Evol. Comput., 12(1):107-125.
Omran MGH, Engelbrecht AP, Salman A (2006). Empirical Analysis of Self-Adaptive Differential Evolution. Euro. J. Oper. Res., 12:785-804.
Omran MGH, Engelbrecht AP, Salman A (2007). Differential Evolution based Particle Swarm Optimization. IEEE Swarm Intel. Symp,(SIS 2007). 4:112-119.
Omran MGH, Salman A, Engelbrecht AP (2005). Self-Adaptive Differential Evolution. In Lecture Notes in Artificial Intelligence, 3801:192-199.

Qian WY, Li AJ (2008). Adaptive differential evolution algorithm for multiobjective optimization problems. Applied Mathematics and Computation, 5:431-440.

Qin AK, Suganthan PN (2005). Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the IEEE Congress Evol. Comput., 2:1785-1791.

Storn R, Price K (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous space. J. Global Optim., 11: 341-359.

Rashedi E, Nezamabadi-pour H (2009). GSA: A Gravitational Search Algorithm. Infor. Sci., 7:2232-2248.

Ratnaweera A, Halgamuge SK, Watson HC (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evol. Comput., 8(3): 240-255.

Reid DJ (1996). Genetic algorithms in constrained optimization. Mathematical, 3:87-111.

Shi XH, Liang YC (2005). An improved GA and a novel PSO-GA-based hybrid algorithm. Infor. Proc. Let., 93:255-261.

Snyman JA (2004). Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Kluwer Academic Publishers, Dordrect, the Netherlands.

Suman B (2004). Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. Computers & Chemical Engineering, 8:1849-1871.

Sun J, Zhang Q, Tsang E (2004). DE/EDA: A new evolutionary algorithm for global optimization. Info. Sci., 169:249-262.

Talibi H, Bautouche M (2004). Hybrid Particle Swarm with Differential Evolution for Multimodal Image Regression. IEEE Inter. Conf. on Ind. Tech, 3:1567-1573.

Thangaraj R, Pant M, Abraham A, Grosan C (2008). Hybrid differential evolution-Particle Swarm Optimization algorithm for solving global optimization problems. Digital Information Management, 2008, ICDIM 2008, Third International Conference on,11:8-24.

Wang YP, Dang CY (2007). An Evolutionary Algorithm for Global Optimization Based on Level-Set Evolution and Latin Squares. IEEE Trans. Evol. Comput., 11(5): 579-595.

Yao X, Liu Y, Lin G (1999). Evolutionary programming made faster. IEEE Trans. Evol. Comput., 3(2): 82-102.

Zhang BJ, Li SY (2007). Ant colony optimization algorithm and its application to neuro-fuzzy controller design. J. Syst. Eng. Elec., 18:603-610.

Zhang CS, Ning JX, Lu S, Ouyang DT, Ding TN (2009). A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. Operations Research Letters, 37:17-122.

Zhang JQ , Sanderson AC (2009). JADE: Adaptive Differential Evolution with Optional External Archive, IEEE Trans. Evol. Comput.. 13(5) 945-958.

Zhang Q, Muhlenbein H (2004). On the convergence of a class of estimation of distribution algorithms. IEEE Trans on Evol. Comput., 8:127-136.

Zhang Q, Sun J, Tsang E, Ford J (2004). Hybrid estimation of distribution algorithm for global optimization. Eng. Comput., 21: 91-107.

Zhang WT, Xie XF (2003). DEPSO: hybrid Particle Swarm with Differential Evolution Operator. IEEE International Conference on Systems Man and Cybernetics, 4:3816-3821.

Appendices

Table A.1. a_{ij} in F_{14}

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

Table A.2. a_i and b_i in F_{15}

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

Table A.3. a_{ij} and c_i in F_{19}

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	35	3.2

Table A.4. p_{ij} in F_{19}

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	0.5574
4	0.0315	0.5743	0.8828

Table A.5. The planning and control components.

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 A.6. p_{ij} in F_{20}

i	p_{ij} ,j = 1, 2, 3					
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 A.7. a_{ij} and c_i in F_{21}, F_{22}, F_{23}.

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