Computers and Mathematics with Applications 63 (2012) 191–200

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

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

Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization

Amir Hossein Gandomi [a,*], Xin-She Yang [b], Siamak Talatahari [c], Suash Deb [d]

[a] Young Researchers Club, Central Tehran Branch, Islamic Azad University, Tehran, Iran
[b] Mathematics and Scientific Computing, National Physical Laboratory, Teddington TW11 0LW, UK
[c] Marand Faculty of Engineering, University of Tabriz, Tabriz, Iran
[d] Department of Computer Science & Engineering, C. V. Raman College of Engineering, Bidyanagar, Mahura, Janla, Bhubaneswar 752054, India

ARTICLE INFO

Article history:
Received 22 February 2011
Received in revised form 6 September 2011
Accepted 4 November 2011

Keywords:
Eagle strategy
Differential evolution
Unconstrained and constrained optimization
Global optimization
Benchmark

ABSTRACT

The performance of an optimization tool is largely determined by the efficiency of the search algorithm used in the process. The fundamental nature of a search algorithm will essentially determine its search efficiency and thus the types of problems it can solve. Modern metaheuristic algorithms are generally more suitable for global optimization. This paper carries out extensive global optimization of unconstrained and constrained problems using the recently developed eagle strategy by Yang and Deb in combination with the efficient differential evolution. After a detailed formulation and explanation of its implementation, the proposed algorithm is first verified using twenty unconstrained optimization problems or benchmarks. For the validation against constrained problems, this algorithm is subsequently applied to thirteen classical benchmarks and three benchmark engineering problems reported in the engineering literature. The performance of the proposed algorithm is further compared with various, state-of-the-art algorithms in the area. The optimal solutions obtained in this study are better than the best solutions obtained by the existing methods. The unique search features used in the proposed algorithm are analyzed, and their implications for future research are also discussed in detail.

1. Introduction

The emerging paradigm of computational modeling and optimization as a problem-solving approach has shaped practice in scientific computing and engineering design and applications. This so-called third approach complements the conventional theoretical and experimental approaches to problem-solving. The essence of such revolutionary progress is the efficient numerical methods and search algorithms. It is no exaggeration to say that how numerical algorithms perform will largely determine the performance and usefulness of modeling and optimization tools [1,2]. Among all optimization algorithms, metaheuristic algorithms are becoming powerful for solving tough nonlinear optimization problems [3–8]. Though most metaheuristic algorithms have relatively high efficiency in terms of finding global optimality, this may be at the expense that there is no guarantee that global optimality can always be found.

The aim of developing modern metaheuristic algorithms is to increase/improve the capability of carrying out global search and to increase the accessibility of the global optimality. Particle swarm optimization is one of the most widely used,

* Corresponding author. Tel.: +1 234 788 0619.
E-mail addresses: a.h.gandomi@gmail.com, ag72@uakron.edu (A.H. Gandomi), xin-she.yang@npl.co.uk, xy227@cam.ac.uk (X.-S. Yang), siamak.talat@gmail.com, talatahari@tabrizu.ac.ir (S. Talatahari), suashdeb@gmail.com (S. Deb).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.11.010

though it only first appeared in 1995 [3], while differential evolution is also versatile and efficient with vectorized genetic operators [4]. Probably the most widely used algorithms in the last three decades have been the genetic algorithms [9] with a huge amount of literature. On the other hand, cuckoo search [5] is one of the latest development, which has demonstrated promising efficiency in solving nonlinear global optimization [6]. Other algorithms such as ant colony and harmony search are now well established in many areas [1,2].

The search efficiency of metaheuristic algorithms can be attributed to the fact that they are designed to imitate the best features in nature with many different sources (e.g., physic-inspired charged system search [7] and music-inspired harmony search [8]). Biological systems are a principal source for proposing new nature-inspired approaches, based on the selection of the fittest, the adaptation to changes, and genetic mechanism in biological systems which have evolved by natural selection over millions of years. There are some algorithms for stochastic optimization, and for example, the Eagle Strategy (ES), developed by Yang and Deb, is one of such algorithms for dealing with stochastic optimization [10].

In this paper, we will investigate the ES further in greater detail by hybridizing it with differential evolution (DE) as a two-stage strategy to enhance its search efficiency, and the proposed algorithm can be called ES–DE. The new two-stage hybrid search method is first verified by using 20 benchmark unconstrained problems. As further validation, we have also tested the algorithm against a well-selected set of constrained problems, and then subsequently applied to thirteen classical benchmarks and three benchmark problems in engineering, reported in the specialized literature. The performance of the proposed algorithm is further compared with various algorithms, state of the art representatives in this area. The optimal solutions obtained in this study are significantly better than the best solutions obtained by the existing methods. The unique search features used in the proposed algorithm are analyzed, and their implications for future research are also discussed in detail.

2. Eagle strategy

Eagle strategy developed by Yang and Deb [10] is a two-stage method for optimization. It uses a combination of crude global search and intensive local search via a balance combination of different algorithms to suit different purposes. In essence, the strategy first explores the search space globally using Lévy flight random walks**;** if it finds a promising solution or a set of promising solutions, then an intensive local search is employed using a more efficient local optimizer such as hill-climbing and the downhill simplex method. Then, the two-stage process restarts again with new global exploration followed by a local search in a new or more promising region.

In the first stage, a population of search agents such as those in used in PSO and differential evolution are initialized with solutions that are generated by Lévy flights in the search space. Then, these solutions are evaluated, and the solutions with the best objective values are recorded as promising solutions. Then, a more intensive local search algorithm is used at the second stage around the recorded best solutions. Iteratively, a new population can then be generated in a new region, which is followed by another local search. In the simplest case, ES is like a random restart hill climbing (RRHC) method. In RRHC, the first step is to generate a good initial point, then hill climbing begins at this point. If the final solution is not good, then a new, different initial point can be generated, which is again followed by another hill-climbing. However, there are some fundamental differences in ES. Firstly, it becomes a strategy, rather than a method. Secondly, at different stages, different algorithms can be used. Thirdly, the two stages can be switched on and off according to the quality of the solutions found. Finally, this ES strategy can mimic the balance of exploration and exploitation in successful metaheuristics such as genetic algorithms and cuckoo search [5].

The advantage of such a combination is to use a balanced tradeoff between global search (which is often slow) and a fast local search. Some tradeoff and balance as well as parameter tuning are important to almost all metaheuristic algorithms. This balance is controlled by a parameter to be introduced later. Another advantage of this method is that we can use any algorithms we like at different stages of the search or even at different stages of iterations. This makes it easier to combine the advantages of various algorithms so as to produce better results.

It is worth pointing out that this is a methodology or strategy, not an algorithm. In fact, we can use different algorithms at different stages and at different times during iterations. The algorithm used for the global exploration should have enough randomness so as to explore the search space diversely and effectively. This process is typically slow initially, and should speed up, as the system gradually converges (or no better solutions can be found after a certain number of iterations). On the other hand, the algorithm used for the intensive local exploitation should be an efficient local optimizer. The idea is to try to reach the local optimality as quickly as possible, ideally with the minimal number of function evaluations. This stage should be fast and efficient.

3. Differential evolution

Differential evolution (DE) was developed by Storn and Price [4]. It is a vector-based evolutionary algorithm, and can be considered as a further development to genetic algorithms. It is a stochastic search algorithm with self-organizing tendency and does not use the information of derivatives. Thus, it is a population-based, derivative-free method. Another advantage of differential evolution over genetic algorithms is that DE treats solutions as real-number strings, thus no encoding and decoding is needed. As in genetic algorithms, design parameters in a d-dimensional search space are represented as vectors, and various genetic operators are operated over their bits of strings or entries of the solution vectors. However, unlike genetic algorithms, differential evolution carries out operations over each component (or each dimension of the solution). Almost everything is done in terms of vectors. For example, in genetic algorithms, mutation is carried out at one site or multiple sites of a chromosome, while in differential evolution, a difference vector of two randomly-chosen population vectors is used to perturb an existing vector as mutation. Such vectorized mutation can be viewed as a self-organizing search, directed towards optimality. This kind of perturbation is carried out over each population vector, and thus can be expected to be more efficient. Similarly, crossover is also a vector-based component-wise exchange of chromosomes or vector segments.

For a d-dimensional optimization problem with d parameters, a population of n solution vectors are initially generated, we have x_i where $i = 1, 2, \ldots, n$. For each solution x_i at any generation t, we use the conventional notation as

$$x_i^t = (x_{1,i}^t, x_{2,i}^t, \ldots, x_{d,i}^t), \tag{1}$$

which consists of d-components in the d-dimensional space. This vector can be considered as the chromosomes or genomes.

Differential evolution consists of three main steps: mutation, crossover and selection. Mutation is carried out by the mutation scheme. For each vector x_i at any time or generation t, we first randomly choose three distinct vectors x_p, x_q and x_r at t, and then generate a so-called donor vector by the mutation scheme

$$v_i^{t+1} = x_p^t + F(x_q^t - x_r^t), \tag{2}$$

where the parameter F lies in the range of [0, 2], often referred to as the differential weight. This requires that the minimum number of population size is $n \geq 4$. In principle, $F \in [0, 2]$, but in practice, a scheme with $F \in [0, 1]$ is more efficient and stable. The perturbation $\delta = F(x_q - x_r)$ to the vector x_p is used to generate a donor vector v_i, and such perturbation is directed and self-organized.

The crossover is controlled by a crossover probability $C_r \in [0, 1]$ and actual crossover can be carried out in two ways: binomial and exponential. The binomial scheme performs crossover on each of the d components or variables/parameters. By generating a uniformly distributed random number $r_i \in [0; 1]$, the jth component of v_i is manipulated as

$$u_{j,i}^{t+1} = v_{j,i} \quad \text{if } r_i \leq C_r \tag{3}$$

otherwise it remains unchanged. This way, it can be decided randomly whether each component exchanges with the donor vector or not.

Selection is essentially the same as that used in genetic algorithms. It is to select the fittest, and for the minimization problem, the minimum objective value.

Most studies have focused on the choice of F, C_r and n as well as the modification of (2). In fact, when generating mutation vectors, we can use many different ways of formulating (2), and this leads to various schemes with the naming convention: DE/x/y/z where x is the mutation scheme (rand or best), y is the number of difference vectors, and z is the crossover scheme (binomial or exponential). The basic DE/Rand/1/Bin scheme is given in (2). For a detailed review on different schemes, please refer to [4].

4. Eagle strategy combined with differential evolution

As ES is a two-stage strategy, we can use different algorithms at different stages. The large-scale coarse search stage can use randomization via Lévy flights. In the context of metaheuristics, the so-called Lévy distribution [11] is a distribution of the sum of N identically and independently distribution random variables whose characteristic function can be written as the following Fourier transform

$$F_N(k) = e^{[-N|k|^\beta]}. \tag{4}$$

The inverse to get the actual distribution $L(s)$ is not straightforward, as the integral

$$L(s) = \frac{1}{\pi}\int_0^\infty \cos(\tau s) e^{-\alpha\tau^\beta} d\tau, \quad (0 < \beta \leq 2) \tag{5}$$

does not have analytical forms, except for a few special cases. Here $L(s)$ is called the Lévy distribution with an index β. For most applications, we can set $\alpha = 1$ for simplicity. Two special cases are $\beta = 1$ and $\beta = 2$. When $\beta = 1$, the above integral becomes the Cauchy distribution. When $\beta = 2$, it becomes the normal distribution. In this case, Lévy flights become the standard Brownian motion.

For the second stage, we can use differential evolution as the intensive local search, rather than the gradient-based methods such as hill-climbing. We know DE is essentially a global search algorithm, it can easily be tuned to do an efficient local search by limiting new solutions locally around the most promising region. Two distinct advantages of DE are: vectorization and derivative-free. As all evolutionary operations are vectorized, it is easy to implement in MATLAB TM or any other programming languages. DE only uses function values, not the derivatives, and it is suitable for a wide range of optimization functions, including continuous, discrete, and even discontinuous and mixed.

The combination in ES may produce better results than those by using pure DE only, as we will demonstrate later. Obviously, the balance of local search (intensification) and global search (diversification) is very important, and so is the balance of the first stage and second stage in the ES. The basic steps of eagle strategy with different evolution (ES–DE) are shown in Fig. 1. From this figure, we can see that at each iteration loop, a global search is carried out first, followed by the local search, there is a 50–50 possibility for each stage, and thus two stages are perfectly balanced.

```
Objective functions f(x)
Initialization and random initial guess x_{t = 0}
while (stop criterion)
    Global exploration by randomization using Levy flights
    Evaluate the objectives and find a promising solution
    Intensive local search around a promising solution via differential evolution
        if (a better solution is found)
                Update the current best
        end
    Update t = t + 1
end
Post-process the results and visualization.
```

Fig. 1. Simplified pseudo code of the ES–DE.

5. Numerical examples

In this section, some benchmark unconstrained and constrained problems are optimized using the proposed ES–DE. The final results are then compared with the solutions of other methods to demonstrate the efficiency of the present approach. The dimensions of these test functions can be varied from lower dimensions such as 1D to very high dimensions such as 5000D. However, in order to validate the performance of each algorithm and thus extrapolate to real design problems, we will use dimensions between 4 and 20 which are in the similar ranges as those used later in actual design problems. This allows us first to see how ES performs against test functions, in comparison with other algorithms. Then we can confirm and see if these results still hold for real-world design problems. Therefore, we can analyze any potential differences and see how different types of problems may affect the performance of each algorithm, and consequently identify the advantages and disadvantages of each algorithm.

There are four parameters in the eagle strategy with differential evolutions, and they are: population size, Lévy exponent, differential weight F and crossover probability. For the Lévy exponent (β), we used a fixed value $\beta = 1.5$, though we did some parameter studies by varying it from 0 to 2. The weight parameter F is taken in the range 0.5–1.0, and we found $F = 0.7$ works best for our simulations. The crossover probability $C_r = 0.9$ is used in all the simulations. For the population size, we have tried to vary it from 10 to 100, and found that the best range is 25–50. Here, we used the population size of 50 for all the problems.

As for the stopping criterion, we can either use the total fixed number of iterations or some given tolerance. In order to compare with other results in the literature, we found the most convenient way is to use a fixed number of iterations. It is an important to have enough iterations as the search accuracy largely depends on the number of iterations. However, as from our parametric study, we see that ES is very efficient, and thus we can set a relatively low number of fixed iterations, compared with other algorithms. Therefore, the number of iterative stages in the eagle strategy Lévy search stage is set to three. The algorithms are coded in MATLAB ™.

5.1. Unconstrained Benchmark Problems

There are many unconstrained test functions in the literature. Here we have chosen the some benchmark optimization problems presented in [12]. These problems have been solved by different evolutionary algorithms by Ma [13], including the genetic algorithm (GA), evolutionary strategy (ESt), particle swarm optimization (PSO), ant colony optimization (ACO), and differential evolution (DE). The main characteristics of the employed problems are presented in Table 1.

The best results obtained in this study for Ucf1–Ucf20 are presented in Table 2. As seen, the proposed algorithm is able to find the global minima in all cases with a very good performance.

Table 3 compares the best results obtained by the proposed algorithm and other famous algorithms (GA, ESt, PSO, ACO and DE) with the global optimums. It should be noticed that all best and mean values are shifted to 0 for easy comparisons. The ES–DE finds the best results in most cases. In these problems, all the algorithms terminated after 10,000 function evaluations for high-dimensional functions and 1000 function evaluations for dimensions lower ones. Based on these results, we carried out a basic Student t-test. For the 95% confidence level, ES–DE is significantly superior to ESt, ACO, and DE, but marginally better than GA, however, it is not statistically significantly different from PSO, though the results obtained by ES–DE are better for most test functions (about 75% of the cases), while sometimes PSO can obtained better results (about 20% of the time). In about 5% of the cases, GA obtained better results, so a t-test suggested the t-statistic for PSO and ES–DE is about 0.4919 for the required $\theta = 1.671$.

Table 1
Characteristics of the unconstrained benchmark problems.

Function type	Function ID	Name	Global Dimension	Optimum	Domain
High-dimensional unimodal functions					
	Ucf01	Sphere function	20	0	[−100 100]
	Ucf02	Schwefel's 2.22 function	20	0	[−10 10]
	Ucf03	Schwefel's 1.2 function	20	0	[−100 100]
	Ucf04	Schwefel's 2.21 function	20	0	[−100 100]
	Ucf05	Rosenbrock's valley function	20	0	[−30 30]
	Ucf06	Step function	20	0	[−100 100]
	Ucf07	Quartic function	20	0	[−1.28 1.28]
High-dimensional multimodal functions					
	Ucf08	Schwefel's function	20	0	[−500 500]
	Ucf09	Rastrigin's function	20	0	[−5.12 5.12]
	Ucf10	Ackley's function	20	0	[−32 32]
	Ucf11	Griewank's function	20	0	[−600 600]
	Ucf12	Penalized function 1	20	0	[−50 50]
	Ucf13	Penalized function 2	20	0	[−50 50]
Low-dimensional standard functions					
	Ucf14	Shekel's Foxholes function	2	1	[−65.536 −65.536]
	Ucf15	Kowalik's function	4	0.003075	[−5 5]
	Ucf16	Six-Hump Camel-Back function	2	−1.0316285	[−5 5]
	Ucf17	Branin's function	2	0.398	x_1:[−5 10], x_2:[0 15]
	Ucf18	Goldstein–Price's function	2	3	[−2 2]
	Ucf19	Hartman's function 1	3	−3.86	[0 1]
	Ucf20	Hartman's function 2	6	−3.32	[0 1]

Table 2
Best results for the benchmark problem by ES–DE.

ID	Best	Mean	Median	Worst	SD	Ave. Time
High-dimensional unimodal functions						
Ucf01	2.50E−06	1.44E−05	1.16E−05	5.15E−05	1.07E−05	4.099
Ucf02	2.36E−03	5.84E−03	5.53E−03	1.12E−02	2.13E−03	4.669
Ucf03	1.74E−01	8.93E−01	7.48E−01	2.55E+00	5.56E−01	5.172
Ucf04	5.15E−03	1.29E−02	1.17E−02	2.90E−02	4.57E−03	4.297
Ucf05	1.05E+01	1.48E+01	1.48E+01	1.83E+01	1.62E+00	4.793
Ucf06	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.269
Ucf07	1.33E−01	5.55E−01	4.60E−01	2.10E+00	3.57E−01	5.220
High-dimensional multimodal functions						
Ucf08	3.53E+03	4.27E+03	4.30E+03	4.78E+03	2.82E+02	6.631
Ucf09	8.64E+01	1.07E+02	1.08E+02	1.32E+02	8.70E+00	4.117
Ucf10	2.09E−03	1.83E−02	8.58E−03	3.42E−01	4.77E−02	5.016
Ucf11	2.42E−04	2.58E−02	9.07E−03	1.84E−01	3.72E−02	5.468
Ucf12	1.17E−07	3.15E−03	5.08E−06	1.56E−01	2.20E−02	9.637
Ucf13	4.17E−06	1.41E−04	6.72E−05	5.76E−04	1.49E−04	7.697
Low-dimensional standard functions						
Ucf14	9.98E−01	1.11E+00	9.98E−01	1.99E+00	2.78E−01	0.730
Ucf15	3.82E−04	8.45E−04	7.89E−04	1.69E−03	2.40E−04	0.536
Ucf16	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	5.33E−09	0.564
Ucf17	3.98E−01	3.98E−01	3.98E−01	3.98E−01	9.31E−07	0.503
Ucf18	3.00E+00	3.00E+00	3.00E+00	3.00E+00	4.52E−11	0.505
Ucf19	−3.86E+00	−3.86E+00	−3.86E+00	−3.86E+00	3.75E−10	0.638
Ucf20	−3.32E+00	−3.30E+00	−3.31E+00	−3.20E+00	3.00E−02	0.617

5.2. Constrained numerical examples

Constrained handling approach

From the implementation point of view, a major barrier is the proper handling of nonlinear constraints for a given problem. For simple limits and bounds, we can simply use

$$P_{i,j}^{LB} \prec P_i \prec P_{i,j}^{UB} \tag{6}$$

where a variable or parameter P_i is always between an upper bound (UB) and lower bound (LB). Here $\prec$ means that inequalities hold in an entry-wise manner. In implementation, we have to ensure that all the generated solutions are within these bounds.

Table 3
Statistical features of the best solutions obtained by different methods for unconstrained problems.

ID		ESt	GA	PSO	ACO	DE	This study
Ucf01	Best	3.26E+03	6.64E−04	8.32E−02	3.66E+01	2.91E+02	**2.50E−06**[a]
	Mean	8.24E+04	9.42E−03	2.76E−01	8.07E+01	7.59E+03	**1.44E−05**
	Stdev	4.67E+03	2.11E−04	6.90E−01	3.15E+01	7.84E+02	**1.07E−05**
Ucf02	Best	3.75E+02	1.19E+02	2.51E+01	6.32E+01	3.67E+01	**2.36E−03**
	Mean	7.32E+03	3.83E+02	3.26E+02	3.95E+02	3.49E+02	**5.84E−03**
	Stdev	9.04E+02	5.67E+03	8.94E+02	7.21E+02	5.34E+02	**2.13E−03**
Ucf03	Best	3.29E+02	2.71E+02	2.64E+02	1.12E+02	1.65E+02	**1.74E−01**
	Mean	9.24E+02	1.09E+03	6.01E+02	5.09E+02	5.21E+02	**8.93E−01**
	Stdev	2.31E+02	4.78E+03	9.55E+03	6.67E+02	8.04E+01	**5.56E−01**
Ucf04	Best	3.97E+01	9.52E−01	3.97E−02	2.07E+01	7.01E+00	**5.15E−03**
	Mean	4.53E+01	7.05E+00	1.55E−01	2.98E+01	6.85E+01	**1.29E−02**
	Stdev	7.21E+00	7.96E−01	2.66E−01	9.04E+00	1.28E+01	**4.57E−03**
Ucf05	Best	2.51E+02	2.40E+01	5.75E−01	7.60E+01	1.88E+01	**1.05E+01**
	Mean	7.58E+03	4.39E+01	3.11E+00	9.91E+02	1.93E+01	**1.48E+01**
	Stdev	3.36E+03	6.68E+00	4.35E+00	9.61E+01	6.54E+00	**1.62E+00**
Ucf06	Best	9.93E+02	1.02E+00	0.00E+00	3.64E+00	4.23E+01	**0.00E+00**
	Mean	4.04E+03	2.91E+00	0.00E+00	5.11E+00	1.10E+02	**0.00E+00**
	Stdev	3.78E+02	1.24E+00	0.00E+00	2.44E+00	1.86E+01	**0.00E+00**
Ucf07	Best	3.31E+01	4.77E+01	5.22E+00	3.22E+01	1.05E+01	**1.33E−01**
	Mean	1.27E+02	1.63E+02	9.64E+00	2.56E+02	1.39E+01	**5.55E−01**
	Stdev	6.45E+02	7.38E+01	8.14E+00	5.53E+02	7.59E+01	**3.57E−01**
Ucf08	Best	2.45E+01	9.52E+01	**4.65E−01**	4.91E+01	5.56E+01	3.53E+03
	Mean	7.01E+01	4.32E+02	**8.81E−01**	6.28E+02	8.74E+01	4.27E+03
	Stdev	2.45E+02	8.56E+02	**1.95E−01**	2.67E+02	7.90E+01	2.82E+02
Ucf09	Best	1.86E+01	**3.37E+01**	1.23E+02	2.46E+01	1.23E+01	8.64E+01
	Mean	6.32E+02	**4.57E+01**	4.71E+02	2.64E+01	3.28E+02	1.07E+02
	Stdev	1.57E+02	**7.48E+02**	7.02E+02	5.56E+01	2.34E+02	8.70E+00
Ucf10	Best	9.20E−01	3.29E−02	5.61E−01	5.57E−01	1.49E−01	**2.09E−03**
	Mean	3.39E+00	4.73E−02	7.58E−01	8.20E−01	5.29E+00	**1.83E−02**
	Stdev	1.07E−01	6.65E−03	7.84E−02	4.45E−01	3.65E−01	**4.77E−02**
Ucf11	Best	1.03E+01	7.28E+01	1.98E+00	3.23E+01	3.76E+01	**2.42E−04**
	Mean	7.80E+01	5.51E+02	3.30E+00	4.55E+01	8.12E+01	**2.58E−02**
	Stdev	2.01E+01	9.94E+02	1.47E+00	5.56E+01	7.76E+01	**3.72E−02**
Ucf12	Best	3.01E−07	7.11E−33	**3.30E−33**	3.23E−08	8.49E−06	1.17E−07
	Mean	6.86E−07	7.98E−32	**2.62E−32**	9.81E−08	1.03E−05	3.15E−03
	Stdev	7.54E−08	4.66E−32	**1.83E−32**	9.77E−07	2.35E−06	2.20E−02
Ucf13	Best	6.45E−09	1.12E−32	**8.54E−33**	4.05E−02	4.94E−08	4.17E−06
	Mean	1.85E−07	3.11E−31	**5.78E−32**	4.12E−01	5.15E−07	1.41E−04
	Stdev	1.80E−08	4.43E−31	**7.85E−32**	9.93E−02	6.67E−07	1.49E−04
Ucf14	Best	4.56E−02	1.09E−02	6.14E−02	1.56E−02	2.32E−04	**3.84E−06**
	Mean	5.92E−02	2.23E−02	4.03E−01	1.60E−02	1.48E−03	**1.14E−01**
	Stdev	4.32E−02	1.75E−02	7.43E−02	9.06E−03	1.22E−04	**2.78E−01**
Ucf15	Best	1.23E−01	1.45E−01	2.71E−02	2.10E−01	1.57E−01	**−2.69E−03**[b]
	Mean	1.86E−01	6.12E−01	8.47E−02	4.32E−01	3.64E−01	**−2.23E−03**
	Stdev	5.54E−01	3.87E−01	9.64E−02	6.89E−01	1.21E−01	**2.40E−04**
Ucf16	Best	9.60E−01	6.71E−03	3.64E−05	2.67E−03	1.17E−06	**4.65E−08**
	Mean	1.27E+00	1.84E−01	1.25E−04	6.06E−03	2.54E−04	**4.93E−08**
	Stdev	7.56E+00	8.17E−01	4.07E−05	3.44E−03	2.12E−04	**5.33E−09**
Ucf17	Best	2.25E−04	6.71E−08	**1.43E−11**	1.77E−10	8.23E−09	1.13E−04
	Mean	6.74E−04	4.60E−07	**2.78E−11**	8.45E−10	7.14E−08	1.12E−04
	Stdev	1.34E−04	6.67E−07	**3.90E−12**	4.65E−10	8.43E−08	9.31E−07
Ucf18	Best	1.53E−02	4.93E−04	4.05E−04	3.91E−03	6.11E−04	**−1.02E−14**
	Mean	1.99E−02	7.91E−04	1.55E−03	3.57E−02	8.25E−04	**2.17E−11**
	Stdev	4.45E−03	7.97E−05	6.87E−04	2.78E−02	3.88E−04	**4.52E−11**
Ucf19	Best	4.68E+00	5.71E+00	1.90E+00	2.74E+00	2.19E+00	**−2.78E−03**
	Mean	7.64E+00	8.07E+00	2.35E+00	4.07E+00	2.56E+00	**−2.78E−03**
	Stdev	2.83E+00	6.92E+00	5.76E+00	3.44E+00	5.89E+00	**3.75E−10**
Ucf20	Best	2.34E+00	1.85E+00	1.68E+00	1.64E+00	2.05E+00	**−2.34E−03**
	Mean	2.71E+00	2.48E+00	2.39E+00	2.44E+00	2.41E+00	**1.83E−02**
	Stdev	5.78E+00	2.90E+00	2.54E+00	5.43E+00	2.25E+00	**3.00E−02**

[a] Bold sets are best sets.
[b] A negative value shows that it is less than the related global optimum presented in Table 1.

For nonlinear equality constraints φ_i and inequality constraints ψ_j, a common method is the penalty method. The idea is to define a penalty function so that the constrained problem is transformed into an unconstrained problem. Now we define

$$\Pi(x, \mu_i, \upsilon_j) = f(x) + \sum_{i=1}^{M} \mu_i \phi_i^2(x) + \sum_{j=1}^{N} \upsilon_j \psi_j^2(x) \quad (7)$$

Table 4
Characteristics of the mathematical constrained problems.

ID	Function type	Problem type	No. variables	No. constrain	No. active constraineds	$\rho(\%)$[a]	Global optimum
Cf1	Quadratic	min	13	9 (I) +0 (E)	6	0.0003	−15
Cf2	Nonlinear	max	20	2 (I) +0 (E)	1	99.9962	−0.803619
Cf3	Polynomial	max	10	0 (I) +1 (E)	1	0.0002	1
Cf4	Quadratic	min	5	6 (I) +0 (E)	2	26.9089	−30665.539
Cf5	Cubic	min	4	2 (I) +3 (E)	3	0	5126.4981
Cf6	Cubic	min	2	2 (I) +0 (E)	2	0.0065	−6961.81388
Cf7	Quadratic	min	10	8 (I) +0 (E)	6	0.0001	24.30621
Cf8	Nonlinear	max	2	2 (I) +0 (E)	0	0.8488	0.095825
Cf9	Polynomial	min	7	4 (I) +0 (E)	2	0.5319	680.6300573
Cf10	Linear	min	8	6 (I) +0 (E)	3	0.0005	7049.248
Cf11	Quadratic	min	2	0 (I) +1 (E)	1	0.0099	0.75
Cf12	Quadratic	min	3	9^3 (I) +0 (E)	0	4.7452	1
Cf13	Exponential	min	5	0 (I) +3 (E)	3	0	0.0539498

[a] The ratio of the size of the feasible search space to the size of the entire search space.

Table 5
Characteristics of the engineering constrained problems.

ID	Variable type	No. variables	No. ineq. constraints	No. active constraints	Global optimum
Ecf1	Continues	4	5	4	≈ 2.38
Ecf2	Mixed	4	4	3	6059.71
Ecf3	Continues	3	4	2	0.01267

Table 6
Results of ES–DE for constrained functions.

Function type	Function ID	Best	Mean	Median	Worst	SD	Ave. time
Standard mathematical test problems							
	Cf01	−15.000	−14.851	−15.000	−13.000	5.02E−01	7.89
	Cf02	−0.80	−0.74	−0.76	−0.53	6.67E−02	67.73
	Cf03	1.00	1.00	1.00	1.00	0.00E+00	12.50
	Cf04	−30665.54	−30665.54	−30665.54	−30665.54	2.20E−11	6.59
	Cf05	5126.50	5127.29	5127.23	5129.42	1.17E+00	6.86
	Cf06	−6961.81	−6961.81	−6961.81	−6961.81	2.18E−12	6.40
	Cf07	24.31	24.31	24.31	24.31	3.97E−04	6.89
	Cf08	−0.10	−0.10	−0.10	−0.10	7.80E−17	6.67
	Cf09	680.63	680.63	680.63	680.63	4.46E−13	8.29
	Cf10	7049.25	7049.42	7049.34	7050.23	1.88E−01	8.42
	Cf11	0.75	0.75	0.75	0.75	0.00E+00	2.89
	Cf12	1.00	1.00	1.00	1.00	0.00E+00	74.93
	Cf13	0.05	0.05	0.05	0.05	5.40E−05	9.17
Standard Engineering Test Problems							
	Ecf1	2.3804	2.3804	2.3804	2.3804	1.79E−15	5.65
	Ecf2	6059.71	6059.71	6059.71	6059.71	9.77E−12	14.90
	Ecf3	0.012665	0.012665	0.012665	0.012665	3.58E−09	4.47

where $1 \leq \mu_i$ and $0 \leq \upsilon_i$ which should be large enough, depending on the solution quality needed. As we can see, when an equality constrained is met, its effect or contribution to $\prod$ is zero. However, when it is violated, it is penalized heavily as it increases $\prod$ significantly. Similarly, it is true when inequality constraints become tight or exactly. It should be mentioned that generation and ramp rate limits are similar type of constraints. These constraints together state the overall upper/lower generation limits of the units.

Constrained benchmark problems

At first, ES–DE was also benchmarked using 13 well-known constrained mathematical problems (Cf1–Cf13). These benchmark problems have been proposed in [14] and extended in [15,16]. The main characteristics of the employed problems are presented in Table 4. The considered diversity for the characteristics of the problems is to cover many kinds of difficulties of constrained global optimization problems that may be encountered in engineering applications [17]. The formulation of these problems can also be found in [18].

Most real-world engineering optimization problems are highly nonlinear with complex constraints, sometimes the optimal solutions of interest do not even exist. In order to see how ES–DE performs, we also test it against some well-known benchmark design problems including welded beam design (Ecf1), pressure vessel design (Ecf2), and the spring design problem (Ecf3). The main characteristics of the employed problems are presented in Table 5. The approximate global optima and formulations of these benchmark engineering problems can also be found in [13].

Table 7
Statistical features of the best solutions obtained by different methods for mathematical constrained problems.

ID & optimum		ESt [19]	GA [20]	SA[a] [21]	PSO [22]	BA[b] [23]	DE [24]	This study
Cf1	Best	−15.0000	−14.9977	−14.9991	−15.0000	−15.0000	**−15.0000**[c]	−15.0000
−15.0000	Mean	−14.7920	−14.9850	−14.9933	13.2734	−14.6582	**−15.0000**	−14.8511
	Worst	−12.7430	−14.9467	−14.9800	−9.7012	−12.4531	**−15.0000**	−13.0000
	S.D.	N.A.	1.40E−02	4.81E−03	1.41E+00	7.05E−01	**2.00E−06**	5.02E−01
Cf2	Best	0.803619	0.802959	0.754913	0.803620	0.803619	0.803619	**0.803311**
0.803619	Mean	0.746236	0.764494	0.371708	0.777143	0.753470	0.724886	**0.738181**
	Worst	0.302179	0.722109	0.271311	0.711603	0.562553	0.590908	**0.530496**
	S.D.	N.A.	2.60E−02	9.80E−02	1.91E−02	5.40E−02	7.01E−02	**6.67E−02**
Cf3	Best	1.0000	0.9997	1.0000	1.0004	1.0000	0.9954	**1.0000**
1.0000	Mean	0.6400	0.9972	0.9992	0.9936	0.9896	0.7886	**1.0000**
	Worst	0.0290	0.9931	0.9915	0.6674	0.9364	0.6399	**1.0000**
	S.D.	N.A.	1.40E−03	1.65E−03	4.71E+02	1.71E−02	1.15E−01	**0.00E+00**
Cf4	Best	−30665.54	−30665.52	−30665.54	−30665.54	**−30665.54**	**−30665.54**	**−30665.54**
−30665.54	mean	−30592.15	−30664.4	−30665.47	−30665.54	**−30665.54**	**−30665.54**	**−30665.54**
	Worst	−29986.21	−30660.31	−30664.69	−30665.53	**−30665.54**	**−30665.54**	**−30665.54**
	S.D.	N.A.	1.60E+00	1.73E−01	6.83E+04	**0.00E+00**	**0.00E+00**	**2.20E−11**
Cf5	Best	5126.497	5126.500	5126.498	5126.647	5126.499	5126.571	**5126.500**
5126.498	Mean	5218.729	5507.041	5126.498	5495.239	5129.425	5207.411	**5127.290**
	Worst	5502.410	6112.075	5126.498	6272.742	5181.474	5327.390	**5129.420**
	S.D.	N.A.	3.50E+02	0.00E+00	4.05E+02	1.09E+01	6.92E+01	**1.17E+00**
Cf6	Best	−6961.814	−6956.251	**−6961.814**	−6961.837	**−6961.814**	**−6961.814**	**−6961.814**
−6961.814	mean	−6367.575	−6740.288	**−6961.814**	−6961.837	**−6961.814**	**−6961.814**	**−6961.814**
	Worst	−2236.950	−6077.123	**−6961.814**	−6961.836	**−6961.814**	**−6961.814**	**−6961.814**
	S.D.	N.A.	2.70E+02	**0.00E+00**	2.61E+04	**0.00E+00**	**0.00E+00**	**2.18E−12**
Cf7	Best	24.3060	24.8820	24.3106	24.3278	24.3062	**24.3062**	24.3062
24.3062	Mean	104.5990	25.7460	24.3795	24.6996	24.3065	**24.3062**	24.3065
	Worst	1120.5410	27.3810	24.6444	25.2962	24.3080	**24.3062**	24.3077
	S.D.	N.A.	7.00E−01	7.16E−02	2.52E+01	3.80E−04	**1.00E−06**	3.97E−04
Cf8	Best	0.095825	0.095825	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
0.095825	Mean	0.091292	0.095819	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
	worst	0.027188	0.095808	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
	S.D.	N.A.	4.40E−06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**7.80E−17**
Cf9	Best	680.6300	680.7260	680.6301	680.6307	**680.6301**	**680.6301**	**680.6301**
680.6301	Mean	692.4720	681.3470	680.6364	680.6391	**680.6301**	**680.6301**	**680.6301**
	Worst	839.7800	682.9650	680.6983	680.6671	**680.6301**	**680.6301**	**680.6301**
	S.D.	N.A.	5.70E−01	1.45E−02	6.68E+03	**0.00E+00**	**0.00E+00**	**4.46E−13**
Cf10	Best	7049.248	7114.743	7059.864	7090.452	7049.261	**7049.248**	7049.253
7049.248	Mean	8442.660	8785.149	7509.321	7747.630	7049.471	**7049.248**	7049.418
	Worst	15580.370	10826.090	9398.649	10533.666	7051.782	**7049.248**	7050.226
	S.D.	N.A.	1.00E+03	5.42E+02	5.52E+02	4.91E−01	**1.67E−04**	1.88E−01
Cf11	Best	0.7500	0.7500	0.7500	0.7499	0.7500	0.7499	**0.7500**
0.7500	Mean	0.7600	0.7520	0.7500	0.7673	0.7500	0.7580	**0.7500**
	Worst	0.8700	0.7570	0.7500	0.9925	0.7500	0.7965	**0.7500**
	S.D.	N.A.	2.50E−03	0.00E+00	6.00E−02	1.00E−08	1.71E−02	**0.00E+00**
Cf12	Best	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
1.00	Mean	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	Worst	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	S.D.	**N.A.**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Cf13	Best	0.05387	N.A.	0.05395	0.05941	**0.05395**	0.05618	**0.05395**
0.05395	Mean	0.74723	N.A.	0.29772	0.81335	**0.05395**	0.28832	**0.05395**
	Worst	2.25988	N.A.	0.43885	2.44415	**0.05397**	0.39210	**0.05397**
	S.D.	N.A.	N.A.	1.89E−01	3.81E+01	**5.65E−06**	1.67E−01	**5.40E−06**

[a] SA is the simulated annealing.
[b] BA is the Bat Algorithm.
[c] Bold sets are best sets.

The proposed ES–DE was executed to find the global optima of the benchmark constrained problems. The best results obtained by ES–DE for the constrained problems are presented in Table 6. In these problems, the search terminated after 10,000 function evaluations, except for the Cf2 with the maximum function evaluation equal to 100,000.

Tables 7 and 8 compares the best results obtained by the proposed algorithm with those obtained by other well-known algorithms for the constrained benchmark mathematical and engineering problems, respectively. As it seen, the proposed algorithm is able to find the global minima in all cases with a very good performance in comparison with other algorithms.

Table 8
Statistical features of the best solutions obtained by different methods for engineering constrained problems.

ID		PSO [25]	ESt [26]	SCA[a] [27]	GA [28]	TCA[b] [29]	DE [30]	This study
Ecf1	Best	2.381	2.5961	2.3847	2.38335	2.38113	2.38097	**2.38036**[c]
	Mean	2.38193	10.1833	2.9607	4.056	2.71041	2.41746	**2.38036**
	Worst	N.A.	4.33259	5.01142	2.993	2.43981	3.3318	**2.38036**
	S.D.	5.24E−03	1.29E+00	N.A.	2.02E−01	9.31E−02	1.45E−01	**1.79E−15**
Ecf2	Best	0.01267	0.01268	0.01267	0.01267	0.01267	0.01267	**0.012665**
	Mean	0.01292	0.0178	0.01292	0.01532	0.01331	0.0127	**0.012665**
	Worst	N.A.	0.01399	0.01672	0.01313	0.01273	0.01345	**0.012665**
	S.D.	4.12E−04	1.27E−03	5.92E−05	6.28E−04	9.40E−05	1.14E−04	**3.58E−09**
Ecf3	Best	6059.71	6832.58	6171	6059.86	6390.55	N.A.	**6059.71**
	Mean	6289.93	8012.62	6335.05	7388.16	7694.07	N.A.	**6059.71**
	Worst	N.A.	7187.31	N.A.	6545.13	6737.07	N.A.	**6059.71**
	S.D.	3.06E+02	2.67E+02	N.A.	1.24E+02	3.57E+02	N.A.	**9.77E−12**

[a] Society and civilization algorithm.
[b] T-Cell algorithm.
[c] Bold sets are best sets.

In particular, for all the constrained engineering optimization problems, the proposed algorithm has found the global optimums in all runs, which is really interesting.

6. Conclusion and discussions

We have carried out benchmark validations for unconstrained and constrained optimization problems using the recently developed eagle strategy in combination with differential evolution. Differential evolution has been demonstrated to be highly suitable for a wider range of optimization problems with very good convergence property. We have used this advantage further in the framework of a two-stage eagle strategy, which tends to carry out a balanced search both globally and locally.

As all metaheuristic algorithms require a certain balance between local intensification and global diversification, we intend to make sure this balance is maintained at each iteration stage. To assist the global exploration more efficiently, we use Lévy flights to help generate diverse new solutions, while, at the same time, we use the excellent convergence property of differential evolution to speed up the local search. The vector-based nature of differential evolution means that the mixing between different solutions is efficient and can be implemented in a straightforward manner. All this ensures a good overall performance of the ES–DE.

From many differential benchmarks of mathematical and engineering problems solved by different methods (ESt, ACO, GA, PSO, DE, SA, BA, SCA, TCA and ES–DE), we found that the ES–DE obtained better optimal solutions in most cases, compared with the results obtained in the literature. This present study suggests that metaheuristic algorithms such as differential evolution and eagle strategy are very efficient; however, this study shows that a proper combination of these two can produce even better performance for solving unconstrained and constrained optimization problems.

Further studies can focus on the sensitivity studies of the parameters used in ES–DE so as to identify optimal parameter ranges for most applications. Another possible improvement is to introduce a switch parameter to control the on and off of each stage. As the global optimality approaches, it may be time-saving to switch to local searches more frequently as the iterations proceed to optimality.

References

[1] T. Baeck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, Taylor & Francis, 1997.
[2] X.S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley & Sons, 2010.
[3] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proc. of IEEE International Conference on Neural Networks, Piscataway, NJ. 1995, pp. 1942–1948.
[4] R. Storn, K.V. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[5] X.S. Yang, S. Deb, Cuckoo search via Levy flights, in: Proceeding of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), India, IEEE Publications, USA, 2009, pp. 210–214.
[6] A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Engineering with Computers, in press (doi:10.1007/s00366-011-0241-y.
[7] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mechanica 213 (3–4) (2010) 267–289.
[8] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[9] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Anbor, 1975.
[10] X.S. Yang, S. Deb, Eagle strategy using Levy walk and firefly algorithms for stochastic optimization. in: J.R. Gonzalez et al. (Eds.), Nature Inspired Cooperative Strategies for Optimization, NICSO 2010, vol. 284, 2010, pp. 101–111.
[11] M. Gutowski, Levy flights as an underlying mechanism for global optimization algorithms, June 2001. ArXiv Mathematical Physics e-Prints.
[12] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions On Evolutionary Computation 3 (2) (1999) 82–102.
[13] H. Ma, An analysis of the equilibrium of migration models for biogeography-based optimization, Information Sciences 180 (2010) 3444–3464.
[14] W. Hock, K. Schittkowski, Test Examples for Nonlinear Programming Codes, Springer-Verlag, Berlin Heidelberg, 1981.

[15] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computation 4 (1) (1996) 1–32.
[16] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 284–294.
[17] A. Hedar, M. Fukushima, Derivative-free filter simulated annealing method for constrained continuous global optimization, Journal of Global Optimization 35 (2006) 521–549.
[18] A.H. Gandomi, X.S. Yang, in: X.Y. Yang, S. Koziel (Eds.), Benchmark Problems in Structural Optimization, in: Computational Optimization and Applications, vol. 356/2011, Springer, 2011, pp. 259–281.
[19] E.M. Montes, C.A.C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, Technical Report EVOCINV-04-2003, Evolutionary Computation Group at CINVESTAV, Secci'on de Computaci'on, Departamento de Ingenierıa El'ectrica, CINVESTAV-IPN, Mexico D.F., Mexico, 2003.
[20] A. Amirjanov, The development of a changing range genetic algorithm, Computer Methods in Applied Mechanics and Engineering 195 (2006) c2495–c2508.
[21] M. Atiqullah Mir, S.S. Rao, Simulated annealing and parallel processing: an implementation for constrained global design optimization, Engineering Optimization 32 (5) (2000) 659–685.
[22] J.C.F. Cabrera, C.A.C. Coello, Handling constraineds in particle swarm optimization using a small population size, in: Gelbukh Alexander, Angel Fernando Kuri Morales (Eds.), in: MICAI 2007: Advances in Artificial Intelligence, 6th International Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, vol. 4827, Springer, Aguascalientes, Mexico, 2007, pp. 41–51.
[23] A.H. Gandomi, X.S. Yang, A.H. Alavi, S. Talatahari, Bat Algorithm for Constrained Optimization Tasks (submitted for publication).
[24] R.L. Becerra, C.A.C. Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering 195 (33–36) (2006) 4303–4322.
[25] S. He, E. Prempain, Q.H. Wu, An improved particle swarm optimizer for mechanical design optimization problems, Engineering Optimization 36 (5) (2004) 585–605.
[26] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Transactions on Evolutionary Computation 4 (3) (2000) 284–294.
[27] T. Ray, K.M. Liew, Society and civilization: an optimization algorithm based on the simulation of social behavior, IEEE Transactions on Evolutionary Computation 7 (4) (2003) 386–396.
[28] H.S. Bernardino, H.J.C. Barbosa, A.C.C. Lemonge, L.G. Fonseca, A new hybrid AIS-GA for constrained optimization problems in mechanical engineering, in: 2008 Congress on Evolutionary Computation, CEC'2008, IEEE Service Center: Piscataway, NJ, USA, Hong Kong 2008, pp. 1455–1462.
[29] V.S. Aragon, S.C. Esquivel, C.A.C. Coello, A modified version of a T-cell algorithm for constrained optimization problems, International Journal for Numerical Methods in Engineering 84 (2010) 351–378.
[30] M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, Information Sciences 178 (15) (2008) 3043–3074.