DIFFERENTIAL EVOLUTION WITH DYNAMIC PENALTY FUNCTION FOR SOLVING CONSTRAINED CONTINUOUS OPTIMIZATION PROBLEMS

Shu-Kai S. Fan
Department of Industrial Engineering and Management
Yuan Ze University
Taiwan (R.O.C.)
simonfan@saturn.yzu.edu.tw

Kuo-Chih Yeh
Yu-Chiang Chuang*
Department of Industrial Engineering and Management
Yuan Ze University
Taiwan (R.O.C.)
{wilddog, *jack}@r2r.iem.yzu.edu.tw

ABSTRACT

This paper presents a differential-evolution-type algorithm for solving constrained continuous optimization problems. The proposed differential evolution (DE) algorithm is developed based upon the penalty function approach, where constraint violation is penalized by placing the constraints into the objective function. Penalty functions can deal both with equality and inequality constraints; in this study, equality constraints are transformed into inequality ones. In addition, to handle infeasibility during DE search, a random re-initialization procedure is executed to produce a new potential solution inside the allowable ranges. Three different types of increasing penalty factors are compared for their performance on convergence. The performance measure includes the best objective value achieved and the number of function evaluations required. The recommendation for the selection of parameter setting in the new algorithm is given through a series of simulation optimizations. The experimental results obtained by solving a variety of benchmark functions are used to demonstrate the effectiveness and efficiency of the penalty-function DE algorithm.

KEYWORDS

Constrained optimization problem; Evolutionary algorithms; Differential evolution (DE); Meta-heuristics; Adaptive penalty function.

1. INTRODUCTION

Evolutionary algorithms have proved a very competitive technique in a wide range of engineering applications. Renowned meta-heuristic algorithms include simulated annealing (SA), genetic algorithm (GA), Tabu search (TS), particle swarm optimization (PSO), among others. However, the vast majority of promising results of EAs are devoted to unconstrained optimization, but most practical problems in real-world are the ones with constraints. In the literature, there have several ways to deal with constraints. Constrained function optimization is an extremely useful tool that can be utilized in almost every facet of engineering, operations research, mathematics, etc. Therefore, there is always a necessity to create alternative ways of dealing with both equality and inequality constraints more effectively. A general formulation for constrained optimization is as follows: (Price et al., 2005)

Find $\vec{x} = (x_0,\ x_1,\ldots,\ x_{D-1}),\ \vec{x} \in \Re^D$
Minimize： $f(\boldsymbol{x})$, subject to：
Inequality constraints： $\gamma_m(\vec{x}) \le 0,\ m = 1,\ 2,\ldots,\ M$,
Equality constraints： $\varphi_n(\vec{x}) = 0,\ n = 1,\ 2,\ldots,\ N$,
and boundary constraints：
$x_{j,L} \le x_j \le x_{j,U},\ j = 0,\ 1,\ldots,\ D-1$.

Early analogies between the mechanism of natural selection and a learning process led to the development of the so-called “evolutionary algorithms” (EAs, Back, 1996), in which the main goal is to simulate the evolutionary process in a computer. Constraints generally make optimization problems the way harder for EAs in that they can cause forbidden regions on the objective function landscape, thus restricting the arbitrary movement of search vectors. The search strategy in our approach is to use differential evolution (DE) algorithms combining with a dynamic penalty function for continuous constrained optimization.

The organization of this paper is organized as follows. In section 2, we briefly review the methods of constraint handling from recent literature, and the basic differential evaluation algorithm is revisited.

Sections 3-4 present the proposed algorithm and evaluate the performance among several EAs for constrained problems. At the end, the conclusion is drawn and the directions of further research are provided.

2. BACKGROUND

Differential evolution (DE) algorithm was first introduced by Storn and Price (1997) to optimize various continuous nonlinear functions. In essence, DE generates new vectors by adding the weighted difference between two population vectors to a third vector, and this operation is called mutation. Afterwards, crossover is used to mix mutated vector's parameters with the parameters with another predetermined vector, the target vector, to yield the so-called trial vector. The last operation is called "selection". Each population vector has to serve once as the target vector so that exactly the number-of-population (NP) competitions take place in every generation.

Some previous algorithms have been proposed for solving constrained problems with DE. Storn proposed constraint adaptation in 1999 (Storn, 1999), in which all the constraints of the problem at hand are relaxed, so that all the individuals in the initial population become feasible. The constraints are reduced toward their original versions at each generation, but the individuals must always remain feasible. Another constraint-handling technique is proposed by Lampinen (2002) (see also Price et al., 2005). Since weight selection tends to be a trial and error optimization problem in its own right, simpler direct constraint handling methods have been designed that do not require the user to "tune" penalty weights. This idea was extended by Storn as CADE (Constraint Adaptation with Differential Evolution) (Storn, 1999) to enhance DE's range of applications. Lampinen has devised a similar method that shows improved convergence speed when compared to CADE. Different from standard DE, each population vector is assigned not just one, but an array of objective values. The array contains both each vector's objective function's value and its constraint function values. He described some rules for the replacement made during the selection procedure. Lampinen's constraint handling approach can reduce the computational effort on evaluating vectors. Not only the objective function has not to be evaluated when one or both vectors are infeasible, but also a vector can also be rejected before all its constraint violations have been computed. So, the merit of this method is saving time.

The most common approach adopted to deal with constrained search space is penalty functions. Penalty function was proposed by Courant (1943) originally and later expanded by Carroll (1961) and Fiacco and McCormick (1968). It gets the idea that is to transform a constrained-optimization problem into an unconstrained one by adding a large value to the objective function based on the amount of constraint violation present in a certain solution. In classical optimization, two kinds of penalty functions are considered: exterior and interior. In the case of exterior method, all vectors start from infeasible solution and move towards the feasible region. The solution sequentially searched approaches asymptotically the optimal point. In the case of interior methods, the penalty term is chosen such that its value will be small at points away from the constraint boundaries and will tend to infinity as the constraint boundaries are approached. Then if starting from a feasible point, the subsequent points generated will always lie within the feasible region since the constraint boundaries act as barriers during the optimization process (Rao, 1996).

Most researchers in the EA community tend to choose exterior penalties, because they do not require an initial solution be feasible. However, it is also an important drawback of interior penalties. The researchers need to find an initial feasible solution when they use interior penalties for constrained optimization problem.

The general formulation of the exterior penalty function can be given by

$$\phi(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{n} r_i \times G_i + \sum_{j=1}^{p} c_j \times L_j \right],$$

where $\phi(\vec{x})$ is the Lagrange-type objective function to be optimized, and G_i and L_i are functions of the constraints $g_i(\vec{x})$ and $h_j(\vec{x})$, respectively. r_i and c_j are positive constants normally called "penalty factors". The most common forms of G_i and L_i are $G_i = \max\left[0, g_1(\vec{x})\right]^{\beta}$ and $L_j = \left|h_j(\vec{x})\right|^{\gamma_i}$ where β and γ are normally chosen 1 or 2 (Coello Coello, 2002).

Penalty functions can deal both with equality and inequality constraints, and the normal approach is to transform an equality to an inequality of the form $\left|h_j(\vec{x})\right| - \varepsilon \le 0$ where ε is the tolerance allowed (a very small value). When using a penalty

function, the amount of constraint violation is calculated to penalize an infeasible solution so that feasible solutions are favored by the selection process (Mezura-Montes et al., 2005).

Despite the popularity of penalty functions in constrained optimization, they have several drawbacks from which the main one is that they require a careful fine turning of the penalty factors that accurately estimates the degree of penalization to be applied as to approach efficiently the feasible region. This is due to the fact that if the penalty is too high or too low, then the problem might become more difficult to be solved by EAs. If setting a large quantity of the penalty, more emphasis is placed on obtaining feasibility and the solution will move very quickly toward a feasible region. Then, the vectors will tend to converge to feasible point even if it is far from optimal. However, if the penalty is set to a small quantity, less emphasis is placed on feasibility, and the vectors may never converge to a feasible solution (Joines and Houck, 1994). So how to select a suitable penalty factor for optimization with constraints is a key issue in this study.

3. DIFFERENTIAL EVOLUTION WITH DYNAMIC PENALTY FUNCTION

The algorithm to be presented is to apply DE to solving constrained optimization coupled with a dynamic penalty function, termed DE_DP. DE is a relatively new EA proposed by Price and Storn (1997). It can be classified as a population-based algorithm and purely heuristic in computation. The algorithm uses the special mutation and crossover operators, based on the linear combination of three individuals, and the selection process is performed via deterministic tournament selection between the parent (target) and the child (trial) created by it. Because the original DE is not well suited for constrained problems, a dynamic penalty function is particularly devised, aiming to enhance the convergence speed and accuracy. Dynamic penalty method produces penalty function values that changes with iteration, that is, strictly increasing during the optimization process. In the initial stage of optimization, penalty values are set very low and, in one of the three cases, almost close to zero. Subsequently, it increases as the population evolves. In the early stage, increasing penalty will make the search with EAs accept nearly all solutions whether feasible or infeasible, and the solution only moves to one that has a better function value. In the later stage,

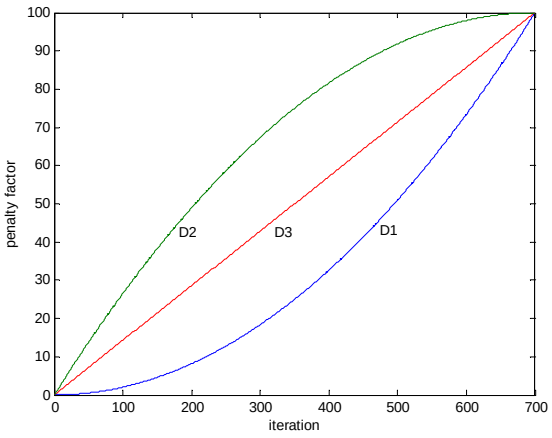

Figure 1 Three different increasing trends of dynamic penalty factors if C_k is 100.

the penalty of infeasible solutions is given very high, and solutions would not accept another infeasible one and the search will stay in several disjointed feasible regions. This method would make the search progress of EAs explore the feasible region (including the global optimum) at the early time, and then the solutions in the population will converge to the global optimum eventually if the dynamic penalty function is appropriately chosen.

Note again that, in this study, the equality constraint is transformed into the inequality one by the function $\left|h_j(\vec{x})\right| - \varepsilon \le 0$ where ε is a tolerance allowed that is always a very small value (say, 10^{-4} opted in this paper). Then, the dynamic penalty function used herein can be expressed by

$$\phi(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m} \left(D_{k,i} \times \max\left[0,\ g_i(\vec{x})\right] \right),$$

where $f(\vec{x})$ is the original objective function, m is number of constraints, $g_i(\vec{x})$ is the constrained function violation, and D is the dynamic penalty factor. The subscript k denotes the type of penalty factors. To gain more knowledge concerning the penalty factor design, three strategies of increasing penalty factors D_1 、 D_2 and D_3 are considered as below:

$$D_1 = g_i^{\alpha} \Big/ g_{\max}^{\alpha} \times C_1,\ i = 1, \ldots,\ g_{\max},$$
$$D_2 = \left(1 - (G - g_i)^{\alpha} \Big/ g_{\max}{}^{\alpha}\right) \times C_2,\ i = 1, \ldots, g_{\max},$$
$$D_3 = g_i \times (C_3 / g_{\max}),\ i = 1, \ldots,\ g_{\max},$$

where g_i is the current iteration (i.e., generation in evolutionary computation), $g_{\max}$ is the maximum

iteration, α is the exponential coefficient, and C is the maximum penalty factor in the last iteration. D_1 and D_2 are of exponential type, referred to as the nonlinear inertia weight variation in particle swarm optimization borrowed from Chatterjee and Siarry (2006), and D_3 is a typical linear growth function. The designed increasing penalty function would prevent the pitfall of premature convergence resulting from using a fixed penalty factor. Figure 1 shows the plot of these three kinds of dynamic penalty factors where the maximum penalty factor is given 100. Henceforth, DE algorithms for constrained optimization with three different dynamic penalty factors are dubbed DE_DP1, DE_DP2 and DE_DP3 for short, respectively.

As the solution violates the constraints over the allowable range, the boundary constraint handling mechanism is triggered to reinitialize the input variables. The boundary constraint handling method is defined by

$$x_i = x_L + rand(0,1) \times (x_U - x_L),$$
$$\text{if } (x_i < x_L) \vee (x_i > x_U),$$

where x_L and x_U are the lower and the upper bounds of the variable x. The pseudo-code of DE_DP is presented in the following:

```
Initialize (Vectors, Parameters)
Evaluate (Vectors, f(x))
t=0
do:
Update (Penalty_factor)
  for (i=1 to NP)
  Mutate & Recombine (Vectors)
  Evaluate(Vectors, f(x) ,Constraints)
  Select (Vectors)
  end for
t=t+1
while (stopping criterion is met).
```

All vectors are initialized in line 1 and evaluated with the objective function, and then enter the while loop until reaching the stopping criterion. In every iteration, the penalty factor is updated first, then mutation and crossover operations are performed to seek vectors with better objective values by selection operation after evaluating all vectors.

Table 1 Parameters setting for each test problem.

Problem	*NP*	*D*	*F*	*CR*	max_iter	C_1	C_2	C_3	α
G1	10	13	0.8	0.01	1000	100	100	100	2
G2	70	20	0.5	0.2	2000	30	30	32	2
G4	10	5	0.8	0.5	800	7000	5000	5000	2
G6	10	2	0.8	0.5	300	15000	5000	5000	2
G8	10	2	0.7	0.7	100	700	200	200	2
G13	50	5	0.9	0.7	2000	0.5	0.5	0.5	2

4. COMPUTATIONAL EXPERIMENTS

4.1. Parameters setting of DE_DP

The major variables of DE contain NP, F and CR. NP is the population size of DE, and it is kept fixed during the optimization process. According to a reasonable choice suggested by Price et al. (2005), NP is between 5*D and 10*D but NP must be at least 4 to ensure that DE will have enough mutually different vectors with which to work. To the best of our knowledge, no optimal choice of the scaling factor F has been suggested in the literature of DE (Kaelo and Ali, 2006). For instance, Price (1999) suggested that F is a value inside [0.4, 0.8], and dynamically calculated values are suggested by Ali (2002). However in original DE, F value is so chosen as to fall in [0, 2]. CR is the crossover constant between [0, 1], and it is found that CR = 0.5 is a good choice by Kaelo and Ali (2006). A good, first choice for CR is 0.1. Yet, since a large CR often speeds up convergence, to first try CR = 0.9 or CR = 1.0 is appropriate in order to see if a quick solution is possible (Storn and Price, 1997). Then C_k is the maximum penalty factor value at the last iteration, and α is the exponential coefficient.

A suite of six well-known test problems named G1 to G13 (see details in Hedar and Fukushima, 2006) attached in Appendix A are used to compare three EAs. The parameters used in our method are listed in table 1. The characteristics of those test problems are deliberately selected diverse enough to cover a variety of complexities that constrained optimization problems might face. DE_DP are applied to solving each problem for 30 independent runs with random initial solutions. The existing EAs taken from the recent literature for the comparison purpose are the

Table 2 Results of FSA, CDE and DE with three kinds of dynamic penalty function for G1-G13.

Pr.	Type	Best known		FSA	CDE	DE_DP1	DE_DP2	DE_DP3
G1	min	-15	Best	-14.999105	-15	-15	-15	-15
			Av.	-14.993316	-14.999996	-14.9999911	-15	-15
			Worst	-14.979977	-14.999993	-14.9999142	-14.9999998	-14.9999998
			SD	0.004813	0.000002	2.26363E-05	3.65333E-08	3.18504E-08
			NFE	205748	100000	10000	10000	10000
G2	max	0.803619	Best	0.7549125	0.803619	0.80307114	0.80314778	0.803192843
			Av.	0.3717081	0.724886	0.802151962	0.802332575	0.801520381
			Worst	0.271311	0.590908	0.792424334	0.800544502	0.792411578
			SD	0.098023	0.070125	0.001861753	0.000510268	0.002453063
			NFE	227832	100000	140000	140000	140000
G4	min	-30665.539	Best	-30665.538	-30665.5387	-30665.5387	-30665.5387	-30665.5387
			Av.	-30665.4665	-30665.5387	-30665.5357	-30665.5385	-30665.5366
			Worst	-30664.688	-30665.5387	-30665.4812	-30665.5374	-30665.4958
			SD	0.173218	0	0.01039085	0.00025683	0.007730471
			NFE	86154	100000	8000	8000	8000
G6	min	-6961.81388	Best	-6961.81388	-6961.81388	-6961.81387	-6961.81388	-6961.81388
			Av.	-6961.81388	-6961.81388	-6961.60674	-6961.81388	-6961.81387
			Worst	-6961.81388	-6961.81388	-6956.19553	-6961.81387	-6961.81376
			SD	0	0	1.023213676	1.20882E-07	2.29632E-05
			NFE	44538	100000	3000	3000	3000
G8	max	0.095825	Best	0.095825	0.095825	0.095825041	0.095825041	0.095825041
			Av.	0.095825	0.095825	0.091367627	0.095825041	0.095825041
			Worst	0.095825	0.095825	0.029143804	0.095825041	0.095825041
			SD	0	0	0.016914438	1.1977E-16	4.61655E-14
			NFE	56476	100000	1000	1000	1000
G13	min	0.0539498	Best	0.0539498	0.05618	0.053949856	0.053949989	0.053949879
			Av.	0.2977204	0.288324	0.053949961	0.053954978	0.05395059
			Worst	0.4388511	0.3921	0.053950353	0.053987567	0.053953035
			SD	0.188652	0.167095	1.37788E-07	9.08251E-06	8.75937E-07
			NFE	120268	100000	100000	100000	100000

derivative-free filter simulated annealing method (FSA) proposed by Hedar and Fukushima (2006) and the cultured differential evolution (CDE) proposed by Becerra and Coello Coello (2006).

4.2. Experimental results

The comparison results of DE_DP in terms of test problems are reported in table 2. All test problems are minimization problems except problems G2 and G8. In problem G1, DE_DP can reach the minimum stably with much fewer number of function evaluations (NFE) than FSA and CDE. The results of all three dynamic penalty factor strategies in G2 dictate a better stability in convergence than FSA and CDE with small standard deviations. Nonetheless, DE_DP requires more NFE than CDE. In the problems G4 and G6, DE_DP could converge to minimum swiftly with much fewer NFE than other two methods. DE_DP2 could attain stably to the maximum in problem G8 as one hundredth of NFE as CDE is required. In problem G13, the three DE_DP algorithms outperform FSA and CDE in average objective values and the standard deviations. In Problems G1, G2 and G13, an improvement upon

the average objective value is reported by using the proposed algorithm. For the remaining problems, equally good performance on the average objective value is observed as FSA and CDE. On the whole, the proposed algorithms outperform dramatically FSA and CDE in NFE. It can be concluded from the experimental results that the second strategy of increasing penalty factor is more suitable than other two versions for testing these six problems. The reason is that the penalty factor in DE_DP2 owns a higher increasing rate than the other two strategies in the early generations, and gradually, the increasing rate turns flat. By the growth trend of D_2 , the population vectors could explore as nearly feasible region as possible in every generations.

5. CONCLUSION AND FURTHER RESEARCH

A differential evolution based algorithm in combination with three different increasing strategies of dynamic penalty factor is presented for handling nonlinear constraint functions in this paper. The proposed algorithm is demonstrated by solving a suite of six well-known test problems taken from the open literature. The comparison with the existing two EAs, FSA and CDE, is reported as well. It can be shown from the experimental results that DE_DP exhibits competitive efficiency and effectiveness while solving 6 constrained optimization problems. The penalty function using the second strategy (DE_DP2) is suggested due to its excellent performance. Another prominent merit of the proposed algorithm is the economy of the number of function evaluations (NFE) required.

APPENDICES

A.1. problem G1

$$\min\ f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

$$\begin{aligned}
\text{st.}\quad & g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0,\\
& g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0,\\
& g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0,\\
& g_4(x) = -8x_1 + x_{10} \le 0,\\
& g_5(x) = -8x_2 + x_{11} \le 0,\\
& g_6(x) = -8x_3 + x_{12} \le 0,\\
& g_7(x) = -2x_4 - x_5 + x_{10} \le 0,\\
& g_8(x) = -2x_6 - x_7 + x_{11} \le 0,\\
& g_9(x) = -2x_8 - x_9 + x_{12} \le 0,\\
& x_i \ge 0,\ \ i = 1,\ldots,\ 13,\\
& x_i \le 1,\ \ i = 1,\ldots,\ 13.
\end{aligned}$$

The bounds：
$\mathbf{U} = (1,1,1,1,1,1,1,1,1,100,100,100,1)$ and
$\mathbf{L} = (0,\ldots,0)$.
Global minimum：
$\mathbf{x}^* = (1,1,1,1,1,1,1,1,1,3,3,3,1),\ f(\mathbf{x}^*) = -15$

A.2. problem G2

$$\max\ f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

$$\begin{aligned}
\text{s.t.}\quad & g_1(x) = -\prod_{i=1}^{n} x_i + 0.75 \le 0\\
& g_2(x) = \sum_{i=1}^{n} x_i + 0.75n \le 0
\end{aligned}$$

The bounds： $\mathbf{U} = (10,\ldots,10)$ and $\mathbf{L} = (0,\ldots,0)$.
Best known value： $f(\mathbf{x}^*) = 0.803619$, for $n = 20$.

A.4. problem G4

$$\begin{aligned}
\min\ f(x) &= 5.3578547x_3^2 + 0.8356891x_1x_5\\
&\quad + 37.293239x_1 - 40792.141
\end{aligned}$$

$$\begin{aligned}
\text{st.}\quad & g_1(x) = u(x) - 92 \le 0,\\
& g_2(x) = -u(x) \le 0,\\
& g_3(x) = u(x) - 110 \le 0,\\
& g_4(x) = -\upsilon(x) + 90 \le 0,\\
& g_5(x) = \omega(x) - 25 \le 0,\\
& g_6(x) = -\omega(x) + 20 \le 0,
\end{aligned}$$

where

$$\begin{aligned}
u(x) &= 85.334407 + 0.0056858x_2x_5\\
&\quad + 0.0006262x_1x_4 - 0.0022053x_3x_5,\\
\upsilon(x) &= 80.51249 + 0.0071317x_2x_5\\
&\quad + 0.0029955x_1x_2 + 0.0021813x_3^2,\\
\omega(x) &= 9.300961 + 0.0047026x_3x_5\\
&\quad + 0.0012547x_1x_3 + 0.0019085x_3x_4.
\end{aligned}$$

The bounds： $\mathbf{U} = (102, 45, 45, 45, 45)$ and
$\mathbf{L} = (78, 33, 27, 27, 27)$
Global minimum：
$\mathbf{x}^* = (78, 33, 29.995256025682, 45, 36.775812905788)$,
$f(\mathbf{x}^*) = -30665.539$.

A.6. problem G6

The bounds： $\mathbf{U} = (100,100)$ and $\mathbf{L} = (13,0)$.
Global minimum：
$\mathbf{x}^* = (14.095, 0.84296),\ f(\mathbf{x}^*) = -6961.81388.$

A.8. problem G8

$$\max\ f(x)=\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$$

$$\text{s.t.}\quad g_1(x)=x_1^2-x_2+1\le 0$$

$$g_2(x)=1-x_1+(x_2-4)^2\le 0.$$

The bounds： $\mathbf{U}=(10,10)$ and $\mathbf{L}=(0,0)$.

Global maximum：

$\mathbf{x}^*=(1.2279713,4.2453733)$, $f(\mathbf{x}^*)=0.095825.$

A.13. problem G13

$$\min\ f(x)=e^{x_1x_2x_3x_4x_5}$$

$$\text{s.t.}\quad h_1(x)=x_1^2+x_2^2+x_3^2+x_4^2+x_5^2-10=0,$$

$$h_1(x)=x_2x_3-5x_4x_5=0,$$

$$h_1(x)=x_1^3+x_2^3+1=0.$$

The bounds： $\mathbf{U}=(2.3,2.3,3.2,3.2,3.2)$ and $\mathbf{L}=(-2.3,-2.3,-3.2,-3.2,-3.2)$.

REFERENCES

Ali, M. M., Torn, A., (2002), "Topographical differential evolution using pre-calculated differentials", in Demyda, G., Saltenis, V., and Zilinskas, A., (eds.), Stochastic and Global Optimization, Kluwer Academic Publisher, London, pp. 1-17.

Back, T., (Ed.), (1996), "Evolutionary Algorithms in Theory and Practice", Oxford University Press, New York.

Carroll, C. W., (1961), "The created response surface technique for optimizing nonlinear restrained systems", Operations Research, Vol. 9, pp. 169-184.

Chatterjee, A., Siarry, P., (2006), “Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization”, Computers & Operations Research, Vol. 33, pp. 859-871

Coello Coello, C. A., (2002), "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art", Computer Methods in Applied Mechanics and Engineering, Vol. 191, pp. 1245-1287.

Courant, R., (1943), "Variational methods for the solution of problems of equilibrium and vibrations", Bull. Am. Math. Soc 49, pp. 1-23.

Fiacco, A. V., McCormick, G. P., (1968), "Extension of SUMT for nonlinear programming： Equality constraints and extrapolation", Manage. Sci. Vol. 12 No. 11, pp. 816-828.

Hedar, A., Fukushima, M., (2006), "Derivative-free filter simulated annealing method for constrained continuous global optimization", Journal of Global Optimization, Vol. 35, pp. 521-549.

Joines, J. A., Houck, C. R., (1994), "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's", IEEE Conference on Evolutionary Computation-Proceeding, pp. 579-584.

Kaelo, P., Ali, M. M., (2006), "A numerical study of some modified differential evolution algorithms", European Journal of Operational Research Vol. 169, pp. 1176-1184.

Lampinen, J., (2002), "A constraint handling approach for the differential evolution algorithm", Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002), vol. 2, Piscataway, pp. 1468-1473.

Landa Becerra, R., Coello Coell, C.A., (2006) "Cultured differential evolution for constrained optimization, Comput", Methods Appl. Mech Engrg. Vol. 195, pp. 304-311.

Mezura-Montes, E., Velazquez-Reyes, J., and Coello Coello, C. A., (2005) "Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization", GECCO, pp. 225-232.

Price, K. V., Storn, R. M., and Lampinen, J. A., (2005), "Differential Evolution: A Practical Approach to Global Optimization", Springer, pp. 201-227.

Price, K., (1999), "An introduction to differential evolution", in Corne, D., Dorigo, M., and Glover, F., (eds.), New Ideas in optimization, McGraw-Hill, London, pp. 79-108.

Rao, S. S., (1996), "Engineering Optimization", third ed., Wiley, New York.

Ricardo Landa Becerra, Carlos A. Coello Coello, Cultured differential evolution for constrained optimization, Comput. Methods Appl. Mech Engrg. 195 (2006) 4303-4322.

Storn, R., (1999), "System desine by constraint adaptation and differential evolution", IEEE Trans. Evol. Comput. 3(1), pp. 22-34.

Storn, R., Price, K., (1997), "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, Vol. 11, pp. 341-359.