Turk J Elec Engin, VOL.12, NO.1 2004, © TÜBİTAK

A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm

Derviş KARABOĞA
Department of Computer Engineering, Erciyes University, Kayseri-TURKEY
e-mail:karaboga@erciyes.edu.tr
Selçuk ÖKDEM
Department of Computer Engineering, Erciyes University, Kayseri-TURKEY
e-mail:okdem@erciyes.edu.tr

Abstract

Differential Evolution (DE) algorithm is a new heuristic approach mainly having three advantages; finding the true global minimum regardless of the initial parameter values, fast convergence, and using few control parameters. DE algorithm is a population based algorithm like genetic algorithms using similar operators; crossover, mutation and selection. In this work, we have compared the performance of DE algorithm to that of some other well known versions of genetic algorithms: PGA, Grefensstette, Eshelman. In simulation studies, De Jong's test functions have been used. From the simulation results, it was observed that the convergence speed of DE is significantly better than genetic algorithms. Therefore, DE algorithm seems to be a promising approach for engineering optimization problems.

Key Words: *Optimization, Genetic Algorithm, Differential Evolution, Test Functions.*

1. Introduction

In the optimization process of a difficult task, the method of first choice will usually be a problem specific heuristics. These techniques using expert knowledge achieve a superior performance. If problem specific technique is not applicable due to unknown system parameters, the multiple local minima, or non-differentiability, Evolutionary Algorithms (EAs) have the potential to overcome these limitations [1]. EAs are a class of direct search algorithms. A conventional direct search method uses a strategy that generates variations of the design parameter vectors. Once a variation is generated, the new parameter vector is accepted or not. The new parameter vector is accepted in the case it reduces the objective function value. This method is usually named the greedy search. The greedy search converges fast but can be trapped by local minima. This disadvantage can be eliminated by running several vectors simultaneously. This is the main idea of differential evolution (DE) algorithm. The most popular EA is genetic algorithm. Although many genetic algorithm versions have been developed, they are still time consuming. In order to overcome this disadvantage, the evolution strategy called DE has been recently proposed by Storn [2, 3]. It has been applied to several engineering problems in different areas [4, 5, 6, 7]. The main difference between the genetic algorithm and

DE is the mutation scheme that makes DE self adaptive and the selection process. In DE, all solutions have the same chance of being selected as parents without dependence of their fitness value. DE employs a greedy selection process: The better one of new solution and its parent wins the competition providing significant advantage of converging performance over genetic algorithms.

DE algorithm is a stochastic optimization method minimizing an objective function that can model the problem's objectives while incorporating constraints. The algorithm mainly has three advantages; finding the true global minimum regardless of the initial parameter values, fast convergence, and using a few control parameters [2]. Being simple, fast, easy to use, very easily adaptable for integer and discrete optimization, quite effective in nonlinear constraint optimization including penalty functions and useful for optimizing multi-modal search spaces are the other important features of DE [8, 9].

The convergence speed is one of the main issues indicating the performance of an EA. There have been some studies to increase the convergence speed of the DE algorithm [8, 9]. In this work, we have applied the DE algorithm to a number of De Jong's functions and compared the speed of convergence of the DE algorithm to that of some popular genetic algorithms such as PGA, Grefensstette and Eshelman algorithms [10]. In the second section the principles of the DE algorithm are described in detail. Section 3 presents the test functions used. The simulation results and comparisons are given in Section 4.

2. Differential Evolution Algorithm

The DE algorithm is a population based algorithm like genetic algorithms using the similar operators; crossover, mutation and selection. The main difference in constructing better solutions is that genetic algorithms rely on crossover while DE relies on mutation operation. This main operation is based on the differences of randomly sampled pairs of solutions in the population.

The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space. The DE algorithm also uses a non-uniform crossover that can take child vector parameters from one parent more often than it does from others. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling the search for a better solution space.

An optimization task consisting of D parameters can be represented by a D-dimensional vector. In DE, a population of NP solution vectors is randomly created at the start. This population is successfully improved by applying mutation, crossover and selection operators.

The main steps of the DE algorithm is given below:

Initialization
Evaluation
Repeat
    Mutation
    Recombination
    Evaluation
    Selection
Until *(termination criteria are met)*

2.1. Mutation

For each target vector $x_{i,G}$, a mutant vector is produced by

$$v_{i,G+1} = x_{i,G} + K \cdot (x_{r1,G} - x_{i,G}) + F \cdot (x_{r2,G} - x_{r3,G}) \tag{1}$$

where $i, r_1, r_2, r_{r3} \in \{1, 2, ..., NP\}$ are randomly chosen and must be different from each other. In Eq(1), F is the scaling factor which has an effect on the difference vector $(x_{r2,G} - x_{r3,G})$, K is the combination factor.

2.2. Crossover

The parent vector is mixed with the mutated vector to produce a trial vector $u_{ji,G+1}$

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} \text{ if } (rnd_j \leq CR) \text{ or } j = rn_i, \\ q_{ji,G} \quad \text{if } (rnd_j > CR) \text{ and } j \neq rn_i \end{cases} \tag{2}$$

where j= 1,2, ... ,D; $r_j \in [0, 1]$ is the random number; CR is crossover constant $\in [0, 1]$ and $rn_i \in$(1,2, ... ,D) is the randomly chosen index.

2.3. Selection

All solutions in the population have the same chance of being selected as parents without dependence of their fitness value. The child produced after the mutation and crossover operations is evaluated. Then, the performance of the child vector and its parent is compared and the better one is selected. If the parent is still better, it is retained in the population.

Figure 1 shows DE's process in detail: the difference between two population members (1,2) is added to a third population member (3). The result (4) is subject to the crossover with the candidate for replacement (5) to obtain a proposal (6). The proposal is evaluated and replaces the candidate if it is found to be better.

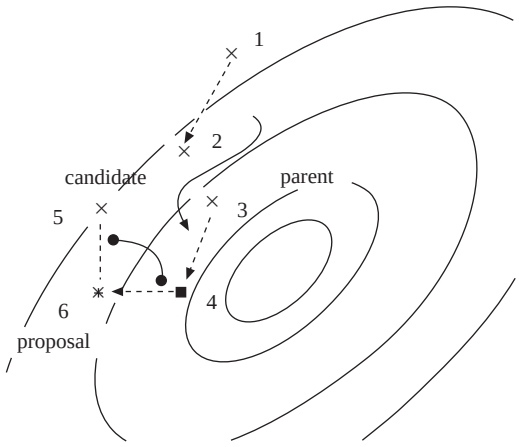

Figure 1. Obtaining a new proposal in DE.

3. Test Functions Used in Simulation Studies

Five test functions (F1-F5) presented in Table 1 have been firstly proposed by De Jong [11] to measure the performance of GAs. After Jong, they have been extensively used by GA researchers and other algorithm

communities. The test environment includes functions which are convex (F1), non-convex (F2), discontinuous (F3), stochastic (F4) and multimodal (F5).

Table 1. Test functions.

Function Number	Function	Limits
F1	$\sum_{i=1}^{3} x_i^2$	$-5.12 \leq x_i \leq 5.12$
F2	$100(x_1^2 - x_2)^2 + (1 - x_1)^2$	$-2.048 \leq x_i \leq 2.048$
F3	$\sum_{i=1}^{5} int(x_i)$	$-5.12 \leq x_i \leq 5.12$
F4	$\sum_{i=1}^{30} ix_i^4 + Gauss(0,1)$	$-1.28 \leq x_i \leq 1.28$
F5	$\left[0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}\right]$	$-65.536 \leq x_i \leq 65.536$

F1 (Sphere) : The first function is smooth, unimodal, strongly convex, symmetric (Figure 2).

F2 (Banana): Rosenbrock's valley is a classical optimization function, also known as Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used to assess the performance of the optimization algorithms (Figure 3).

F3 (Step): This function represents the problem of flat surfaces. Flat surfaces are obstacles for optimization algorithms because they do not give any information as to which direction is favorable. Unless an algorithm has variable step sizes, it can get stuck on one of the flat plateaus (Figure 4).

F4 (Stochastic): This is a simple unimodal function padded with noise. In this type of function, the algorithm never gets the same value on the same point. Algorithms which are not good on this test function will do poorly on noisy data (Figure 5).

F5 (Foxholes): Function F5 is an example of a function with many local optima. Many standard optimization algorithms get stuck in the first peak they find (Figure 6).

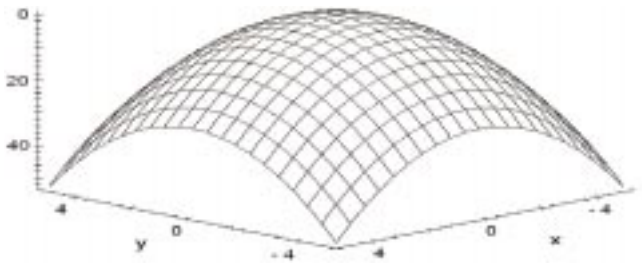

Figure 2. F1 Sphere Function.

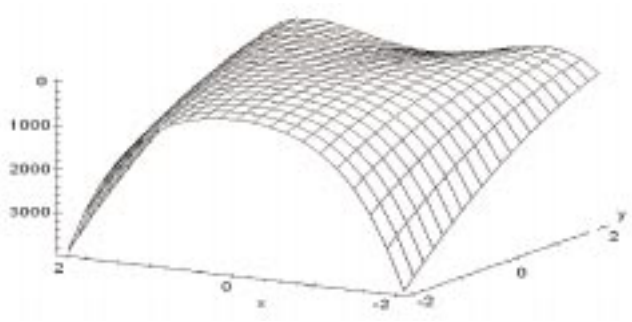

Figure 3. F2 Rosenbrock's Valley Function.

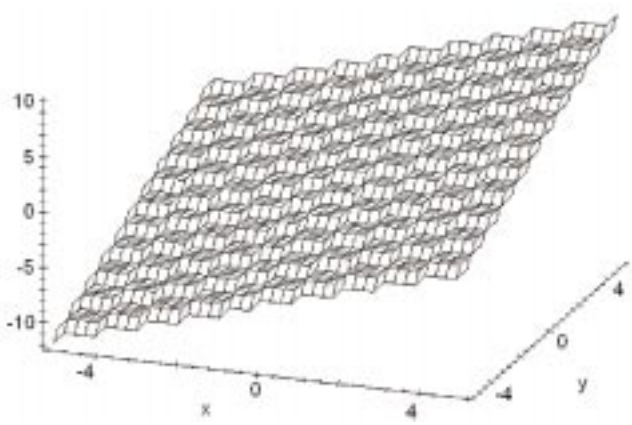

Figure 4. F3 Step Function.

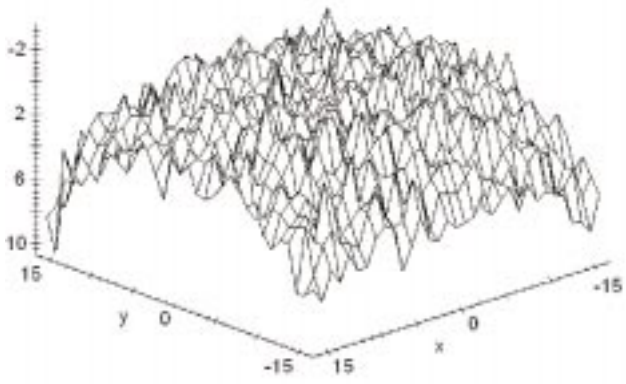

Figure 5. F4 Stochastic Function.

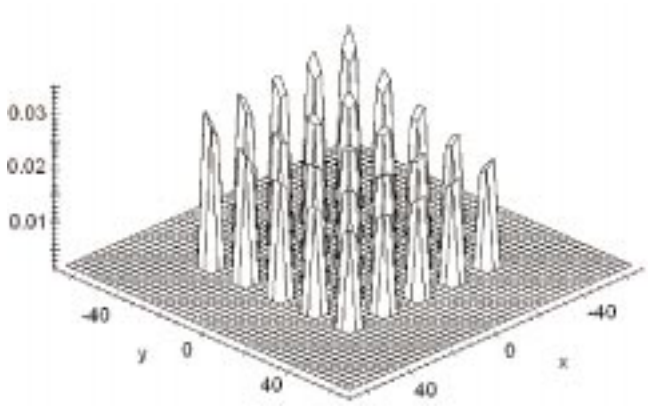

Figure 6. F5 Foxholes Function.

4. Simulation Results

The DE algorithm has a few control parameters: number of population NP, scaling factor F, combination coefficient K, and crossover rate CR. The problem specific parameters of the DE algorithm are the maximum generation number $G_{\max}$ and the number of parameters defining the problem dimension D. The values of these two parameters depend on the problem to be optimized. In the simulations, it was observed that the value of scaling factor significantly affected the performance of DE. This can be seen in Table 2. In the table, 10000^{+}represents the number of generations which are higher than 10000. In order to get the best performance from the DE, the scaling factor value, F, must be optimally tuned for each function. Of course, this is a time-consuming task. For the simplicity and flexibility, the value of F was randomly chosen between -2 and 2 for each generation instead of using a constant value. The results produced for this selection are shown in Table 3.

The average number of generations presented in Table 2-3 provides the minimum function values in three-digit accuracy. We have compared the performance of the DE algorithm to that of other some well known genetic algorithms: PGA, Grefensstette and Eshelman [10]. These algorithms were run 50 times and the DE algorithm was run 1000 times for each function to achieve average results. For each run, the initial population was randomly created by means of using different seed numbers.

The results belonging to the DE algorithm given in Table 3 were achieved using parameter values that are 0,8 for CR and 0,5 for K. Parameter NP was chosen 10 for small dimensional and 20 for large dimensional functions.

Table 2. Average of number of generations for various F values.

F	F1(NP=10, K=0.5, CR=0.8)	F2(NP=10, K=0.5, CR=0.8)	F3(NP=20, K=0.5, CR=0.8)	F4(NP=20, K=0.5, CR=0.8)	F5(NP=10, K=0.5, CR=0.8)
0.2	240	10000^{+}	165	1100	10000^{+}
0.4	180	10000^{+}	145	1150	10000^{+}
0.6	200	10000^{+}	140	1750	10000^{+}
0.8	265	830	130	6700	10000^{+}
1	360	775	115	10000^{+}	775
1.5	700	1200	100	10000^{+}	1520
2	1190	1710	96	10000^{+}	3160

Table 3. Average of number of generations.

Algorithms	F1 Domain : [-5.12,5.12] Dim 3	F2 Domain: [-2.048,2.048] Dim 2	F3 Domain: [-5.12,5.12] Dim 5	F4 Domain: [-1.28,1.28] Dim 30	F5 Domain: [-65.536,65.536] Dim 2
PGA($\lambda = 4$)	1170	1235	3481	3194	1256
PGA($\lambda = 8$)	1526	1671	3634	5243	2076
Grefensstette	2210	14229	2259	3070	4334
Eshelman	1538	9477	1740	4137	3004
DE(F:Random Value)	260	670	125	2300	1200

When the results produced by the DE algorithm for all functions were evaluated together, it was observed that the best of the average of number of generations is obtained when the value of F is randomly generated between -2 and 2. It is seen in Table 3 that the convergence speed of the DE is significantly better than GAs although the scaling factor value was randomly produced for each generation without tuning. From the results it can be concluded that the DE algorithm is usually able to find optimum solutions for the functions with a less number of generations than other algorithms. The most significant improvement is with F3, resulting about 14 times smaller average of number of generations than other algorithm producing the best result for the same function.

5. Conclusion

The DE algorithm is a new heuristic approach mainly having three advantages; finding the true global minimum regardless of the initial parameter values, fast convergence, and using a few control parameters. In this work, the performance of the DE algorithm has been compared to that of some other well known genetic algorithms. From the simulation studies, it was observed that the convergence speed of the DE is significantly better than the GAs presented in this work. Therefore, the DE algorithm seems to be a promising approach for engineering optimization problems.

References

[1] K. Price, New Ideas in Optimization, McGraw-Hill Publishing Company, pp. 77-106, 1999.

[2] R. Storn, "Differential Evolution, A Simple and Efficient Heuristic Strategy for Global Optimization over Continuous Spaces", Journal of Global Optimization, Vol. 11, Dordrecht, pp. 341-359, 1997.

[3] http://www.icsi.berkeley.edu/∼storn/code.html

[4] P. Thomas and D. Vernon, "Image registration by differential evolution", P. Thomas and D. Vernon, Image registration by differential evolution, Proceedings of the Irish Machine Vision and Image Processing Conference, pp. 221-225, 1997.

[5] R. Storn, "Differential evolution design of an IIR-filter with requirements of magnitude and group delay", Proceedings of the IEEE Conference on Evolutionary Computation, pp. 268-273, 1996.

[6] R. Storn, "System Design by Constraint Adaptation and Differential Evolution", IEEE Trans. on Evolutionary Computation, Vol. 3, No. 1, pp. 22-34, 1999.

[7] J. Liu, J. Lampinen, "A fuzzy adaptive differential evolution algorithm", Lappeenranta University of Technology, TENCON '02 Proceedings 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering, Vol. 1, pp. 606-611, 2002.

[8] M. Strens and A. Moore, "Policy Search using Paired Comparisons", Journal of Machine Learning Research, Vol. 3, pp. 921-950, 2002,

[9] H.A. Abbass, Ruhul Sarker, and Charles Newton. "PDE: A pareto-frontier differential evolution approach for multi-objective optimization problems". Proceedings, 2001.

[10] H. Mühlenbein, M. Schomisch, J.Born, "The Parallel Genetic Algorithm as Function Optimizer", Proceedings of The Fourth International Conference on Genetic Algorithms, University of California, San diego, pp. 270-278, 1991.

[11] K.A. De Jong. "An Analysis of the Behavior of a Class of Genetic Adaptive Systems". Phd thesis, University of Michigan, Dissertation Abstracts International 36(10), 5140B. (University Microfilms No. 76-9381), 1975.