Self-adaptive Differential Evolution

Mahamed G.H. Omran[1], Ayed Salman[2], and Andries P. Engelbrecht [3]

[1] Faculty of Computing & IT, Arab Open University, Kuwait
mjomran@engineer.com
[2] Computer Engineering Department, Kuwait University, Kuwait
ayed@eng.kuniv.edu.kw
[3] Department of Computer Science, University of Pretoria, Pretoria, South Africa
engel@cs.up.ac.za

Abstract. Differential Evolution (DE) is generally considered as a reliable, accurate, robust and fast optimization technique. DE has been successfully applied to solve a wide range of numerical optimization problems. However, the user is required to set the values of the control parameters of DE for each problem. Such parameter tuning is a time consuming task. In this paper, a self-adaptive DE (SDE) is proposed where parameter tuning is not required. The performance of SDE is investigated and compared with other versions of DE. The experiments conducted show that SDE outperformed the other DE versions in all the benchmark functions.

1 Introduction

Evolutionary algorithms (EAs) are general-purpose stochastic search methods simulating natural selection and evolution in the biological world. EAs differ from other optimization methods, such as Hill-Climbing [1] and Simulated Annealing [2], in the fact that EAs maintain a population of potential (or candidate) solutions to a problem, and not just one solution [3].

Due to its population-based nature, EAs can avoid being trapped in a local optimum and consequently can often find global optimal solutions. Thus, EAs can be viewed as global optimization algorithms. However, it should be noted that EAs may fail to converge to a global optimum [4].

Recently, Storn and Price [5] proposed a new EA called Differential Evolution (DE). DE is similar to Genetic Algorithms (GAs) [6]. The main difference between GAs and DE is that, in GAs, mutation is the result of small perturbations to the genes while in DE mutation is caused by arithmetic combinations of individuals [6]. At the beginning of the evolution, the mutation operator of DE favors exploration. Then as the evolution progresses the mutation operator favors exploitation. Hence, DE automatically adapts the mutation increments (i.e. search step) to the correct value during the evolution process. Another difference between GAs and DE is that simple GAs use a binary representation while DE uses a floating-point representation [7].

DE is easy to implement, requires little parameter tuning [8] and exhibits fast convergence [9]. However, according to Krink *et al.* [10], noise may adversely affect the performance of DE due to its greedy nature.

Y. Hao et al. (Eds.): CIS 2005, Part I, LNAI 3801, pp. 192 – 199, 2005.

DE has been successfully applied to solve a wide range of optimization problems. DE is now generally considered as a reliable, accurate, robust and fast optimization technique. However, the user needs to find the best values for the control parameters of DE for each problem. Finding the best values for the control parameters for each problem is a time consuming task. This paper proposes a new version of DE where the control parameters are self-adaptive. The new version is called *Self-adaptive Differential Evolution* (SDE). The results of the experiments conducted are shown and compared with the versions of DE proposed by Price and Storn [11].

The reminder of the paper is organized as follows: Section 2 provides an overview of DE. The proposed version of DE is given in Section 3. Benchmark functions to measure the performance of DE are provided in Section 4. Results of the experiments are presented in Section 5. Finally, Section 6 concludes the paper.

2 Differential Evolution

Differential evolution does not make use of a mutation operator that depends on some probability distribution function, but introduces a new arithmetic operator which depends on the differences between randomly selected pairs of individuals.

For each parent, $\boldsymbol{x}_i(t)$, of generation t, an offspring, $\boldsymbol{x}'_i(t)$, is created in the following way: Randomly select three individuals from the current population, namely $\boldsymbol{x}_{i1}(t)$, $\boldsymbol{x}_{i2}(t)$ and $\boldsymbol{x}_{i3}(t)$, with $i1 \neq i2 \neq i3 \neq i$ and $i1$, $i2$, $i3 \sim U(1,\ldots, s)$, where s is the population size. Select a random number $r \sim U(1,\ldots, N_d)$, where N_d is the number of genes (parameters) of a single chromosome. Then, for all parameters $j = 1,\ldots, N_d$, if $U(0, 1) < P_r$, or if $j = r$, let

$$x'_{i,j}(t) = x_{i3,j}(t) + F(x_{i1,j}(t) - x_{i2,j}(t)) \tag{1}$$

otherwise, let

$$x'_{i,j}(t) = x_{i,j}(t) \tag{2}$$

In the above, P_r is the probability of reproduction (with $P_r \in [0, 1]$), F is a scaling factor with $F \in (0, \infty)$, and $x'_{i,j}(t)$ and $x_{i,j}(t)$ indicate respectively the j-th parameter of the offspring and the parent.

Thus, each offspring consists of a linear combination of three randomly chosen individuals when $U(0, 1) < P_r$; otherwise the offspring inherits directly from the parent. Even when $P_r = 0$, at least one of the parameters of the offspring will differ from the parent (forced by the condition $j = r$).

The mutation process above requires that the population consists of more than three individuals.

After completion of the mutation process, the next step is to select the new generation. For each parent of the current population, the parent is replaced with its offspring if the fitness of the offspring is better, otherwise the parent is carried over to the next generation.

Price and Storn [11] proposed ten different strategies for DE based on the individual being perturbed (i.e. $\boldsymbol{x}_{i3,j}(t)$), number of individuals used in the mutation process and the type of crossover used. The strategy shown in this section is known as DE/rand/1. This strategy is considered to be the most widely used and the most successful strategy [7].

The performance of DE is sensitive to the choice of control parameters [5, 12, 13, 14]. Recently, there were several attempts to control the parameters of DE. Liu and Lampinen [13] proposed a fuzzy DE where the values of the control parameters (i.e. F and P_r) are adapted using fuzzy logic. Human knowledge and previous experience are used to establish the fuzzy rules and membership functions. This is not very objective and depends on how good the knowledge of the expert is.

Abbas [15] proposed the Self-adaptive Pareto DE (SPDE), a self-adaptive approach to DE for multi-objective optimization problems. In SPDE, the parameter F is generated for each variable from a normal distribution, $N(0,1)$. Each individual, i, has its own probability of reproduction, P_i. The parameter P_i is first initialized for each individual in the population from a uniform distribution between 0 and 1. Then, P_r is adapted according to the following equation:

$$P_i(t) = P_{i1}(t) + N(0,1) \times (P_{i2}(t) - P_{i3}(t)) \tag{3}$$

where $i1, i2, i3 \sim U(1,\ldots, s)$.

According to Abbas [15], the proposed approach performed well compared to other evolutionary multi-objective optimization approaches.

Zaharie [12] theoretically studied the behavior of DE and proposed an approach of adapting the control parameters of DE that is guided by the evolution of population diversity.

More recently, Bui *et al.* [16] proposed to use DE/rand/1 that generates F from a uniform distribution between 0 and 1. According to Bui *et al.* [16], this approach performed better than the conventional DE/rand/1 using a fixed value of F.

3 Self-adaptive Differential Evolution (SDE)

Similar to SPDE, SDE uses self-adaptation to adapt the control parameters of DE. However, SDE self-adapts F, contrary to SPDE which uses a normal distribution $N(0,1)$ for F. Furthermore, in SDE, P_r is generated for each individual from a normal distribution while SPDE self-adapts P_r according to equation (3). In addition, SPDE uses a normal distribution with mean zero and standard deviation one (i.e. $N(0,1)$). On the other hand, SDE uses normal distributions of different means and standard deviations.

SDE works as follows: For each parent, $\boldsymbol{x}_i(t)$, of generation t, an offspring, $\boldsymbol{x}'_i(t)$, is created in the following way: Randomly select three individuals from the current population, namely $\boldsymbol{x}_{i1}(t)$, $\boldsymbol{x}_{i2}(t)$ and $\boldsymbol{x}_{i3}(t)$, with $i1 \neq i2 \neq i3 \neq i$ and $i1, i2, i3 \sim U(1,\ldots, s)$, where s is the population size. Select a random number $r \sim U(1,\ldots, N_d)$, where N_d is the number of genes (parameters) of a single chromosome. Then, for all parameters $j = 1,\ldots, N_d$, if $U(0, 1) < N(0.5,0.15)$, or if $j = r$, let

$$x'_{i,j}(t) = x_{i3,j}(t) + F_i(t)(x_{i1,j}(t) - x_{i2,j}(t)) \tag{4}$$

otherwise, let

$$x'_{i,j}(t) = x_{i,j}(t)$$

where

$$F_i(t) = F_{i4}(t) + N(0,0.5) \times (F_{i5}(t) - F_{i6}(t)) \tag{5}$$

with $i4$, $i5$, $i6 \sim U(1,\ldots, s)$.

Thus, each individual i has its own scaling factor F_i which is a function for the scaling factor of randomly selected individuals. The parameter F_i is first initialized for each individual in the population from a normal distribution $N(0.5,0.15)$ generating values which fits well within the range (0,1].

The rationale behind using a normal distribution $N(0.5,0.15)$ for P_r is that $N(0.5,0.15)$ will generate values in the range of [0.5-3×0.15,0.5+3×0.15] which covers the P_r boundary giving more probability to values surrounding 0.5. The reason for preferring values surrounding 0.5 is that $P_r = 0.5$ represents a uniform crossover (i.e. there is an equal probability that the new offspring will be chosen from either the perturbed individual or from the original individual). Hence, using $N(0.5,0.15)$ will provide a relatively fair chance for both the perturbed individual and the original individual to be selected as the new offspring. According to our preliminary experiments (not shown because of space limit), using a normal distribution $N(0.5,0.15)$ for P_r generally performed better than using equation (3) on the tested functions.

The normal distribution $N(0,0.5)$ in equation (5) is used instead of $N(0,1)$ on the basis of the assumption that $N(0,0.5)$ will generate values in the range of [-1.5,1.5] which covers the F boundary (remember that F is usually $\in$ [0.5, 1]) better than using $N(0,1)$. $N(0,1)$ will generate values in the range [-3,3]. Hence, $N(0,1)$ may increase the probability of generating values of F outside the range (0,1]. If the value of F_i is not within (0,1], F_i will be repaired according to the repair rule used by Abbas [15]. The repair rule will truncate the constant part of the value (e.g. the value 1.4 will be adjusted to 0.4).

4 Benchmark Functions

The benchmark functions in this section provide a balance of unimodal and multimodal functions. These standard test functions, taken from the literature of evolutionary computation, have been used in various DE studies [6, 10].

For each of these functions, the goal is to find the global minimizer. Formally speaking:

Given f: $\Re^{N_d} \rightarrow \Re$

find $\boldsymbol{x}^* \in \Re^{N_d}$ such that $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in \Re^{N_d}$

Spherical: $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_d} x_i^2 \tag{6}$$

with $-100 \leq x_i \leq 100$.

Rosenbrock: $\boldsymbol{x}^* = (1,1,\ldots,1)$, with $f(\boldsymbol{x}^*) = 0$.

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_d/2} \left(100\left(x_{2i} - x_{2i-1}^2\right)^2 + \left(1 - x_{2i-1}\right)^2\right) \tag{7}$$

with $-2.048 \leq x_i \leq 2.048$.

Rastrigin: $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_d} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{8}$$

with $-5.12 \leq x_i \leq 5.12$.

Griewank: $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

$$f(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{N_d} x_i^2 - \prod_{i=1}^{N_d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{9}$$

with $-600 \leq x_i \leq 600$.

Spherical and Rosenbrock are unimodal while Rastrigin and Griewank are multimodal functions.

5 Experimental Results

In this section, SDE is compared with the other strategies of DE proposed by Price and Storn [11]. The control parameters for these strategies were set as follows: F = 0.5 and P_r = 0.9. No attempt was made to tune the DE parameters to each problem. The rationale behind this decision is the fact that in real-world applications the evaluation time is significant and as such parameter tuning is usually a time consuming process [10]. For all the algorithms used in this section s = 40. Furthermore, all functions were implemented in 50 dimensions.

The results reported in this section are averages and standard deviations over 30 simulations. Each simulation was allowed to run for 20 000 evaluations of the objective function.

Table 1 shows the results of the experiments. Examining the results, SDE significantly outperformed the other strategies in all the test functions. This shows the

efficiency of the SDE scheme in the unimodal and multimodal functions. The improvement is even more significant for the Rastrigin function. The Rastrigin function has many good local optima. Hence, it is known to be a difficult function to optimize. SDE performed very well in this function suggesting that SDE works well with difficult multimodal functions. Furthermore, examining the standard deviations of all the algorithms, it can be seen that SDE has the smallest standard deviation. Thus, one can conclude that SDE is more stable and thus more robust than the other versions of DE.

Figures 1, 2, 3 and 4 show the DE/best/1 and DE/rand-to-best/1 getting trapped in a local optimum very early during the search. This behavior is expected because of the low diversity in both strategies. Figures 1, 2 and 4 show that SDE has a faster convergence rate than DE/rand/2. On the other hand, the three figures show that DE/rand/1 and DE/best/2 converge faster than SDE. However, SDE converges to a better solution than both DE/rand/1 and DE/best/2. For the Rastrigin function shown in Figure 3, it can be observed that SDE converges to a better solution faster than DE/rand/1, DE/rand/2 and DE/best/2.

Table 1. Mean and standard deviation (±SD) of the function optimization results (N_d =50)

	Sphere	Rosenbrock	Rastrigin	Griewank
DE/rand/1	0.004225± 0.008481	135.352596± 41.405378	367.286133± 28.252943	0.810391± 0.491085
DE/best/1	9.10373± 2.299977	20704.1313± 10067.8147	291.457902± 38.724004	213.568685± 53.005959
DE/best/2	0.000767± 0.001640	165.565743± 58.467422	363.211698± 125.392344	0.33766± 0.228069
DE/rand/2	0.038096± 0.022325	191.239558± 55.30455	405.489141± 18.75283	1.540165± 0.184424
DE/rand-to-best/1	4.294976± 0.797361	6016.779462± 2551.570461	162.186077± 28.675567	94.336715± 20.46955
SDE	**0.000034± 0.000059**	**108.599572± 37.453761**	**36.808762± 8.520613**	**0.070461± 0.067611**

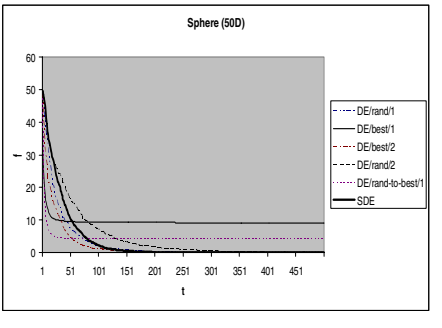

Fig. 1. DE strategies performance for the Sphere (50D)

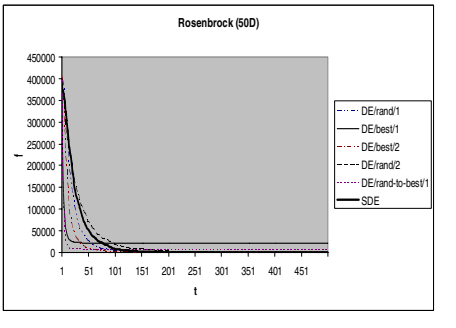

Fig. 2. DE strategies performance for the Rosenbrock (50D)

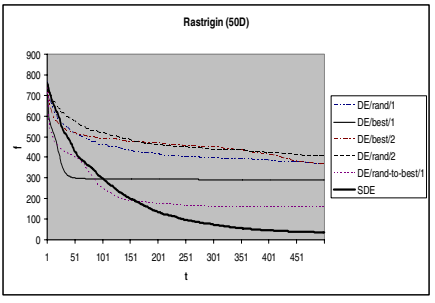

Fig. 3. DE strategies performance for the Rastrigin (50D)

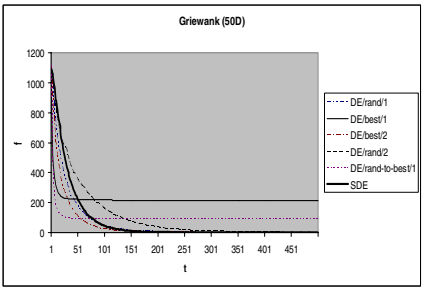

Fig. 4. DE strategies performance for the Griewank (50D)

6 Conclusion

The performance of DE is sensitive to the choice of control parameters. Finding the best values for these parameters for each problem is a time consuming task. This paper proposes a self-adaptive version of DE, called SDE. The approach was tested on four benchmark functions where it outperformed other well-known versions of DE.

Future research will investigate the contribution of each parameter in SDE. Furthermore, SDE will be tested on more functions and compared with other optimization techniques (e.g. SPDE, ES and EP). In addition, the effect of noise on the performance of SDE will be explored. The effect of the population size and the problem dimension will also be investigated.

References

1. Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics. Springer-Verlag, Berlin (2000)
2. Van Laarhoven, P., Aarts, E.: Simulated Annealing: Theory and Applications. Kluwer Academic Publishers (1987)
3. Engelbrecht, A.: Computational Intelligence: An Introduction. John Wiley and Sons (2002)
4. Gray, P., Hart, W., Painton, L., Phillips, C., Trahan, M., Wagner, J.: A Survey of Global Optimization Methods. Sandia National Laboratories, 1997, http:// www.cs.sandia.gov/opt/survey (visited 2 July 2004) (1997)
5. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA (1995)
6. Feoktistov, V., Janaqi, S.: Generalization of the Strategies in Differential Evolution. In: the Proceedings of the 18th International Parallel and Distributed Processing Symposium (2004) 165-170
7. Babu, B., Jehan, M.: Differential Evolution for Multi-Objective Optimization. In: the 2003 Congress on Evolutionary Computation, Vol. 4 (2003) 2696-2703

8. Paterlini, S., Krink, T.: High Performance Clustering with Differential Evolution. In: the 2004 Congress on Evolutionary Computation, Vol. 2 (2004) 2004-2011
9. Karaboga, D., Okdem, S.: A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm. Turk Journal of Electrical Engineering, Vol. 12, No. 1 (2004) 53-60
10. Krink, T., Filipic, B., Fogel, G.: Noisy Optimization Problems – A Particular Challenge for Differential Evolution. In: the 2004 Congress on Evolutionary Computation (2004) 332-339
11. Price, K., Storn, R.: DE Web site, http://www.ICSI.Berkeley.edu/~storn/code.html (visited 9 July 2005) (2005)
12. Zaharie, D.: Parameter Adaptation in Differential Evolution by Controlling the Population Diversity. In: the 4th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, Analele Univ. Timisoara, Vol. XXXX, Issue 2 (2002)
13. Liu, J., Lampinen, J.: A Fuzzy Adaptive Differential Revolution Algorithm. In: Proceedings of the IEEE TENCON'02 (2002) 606-611
14. Wei, C., He, Z., Zhang, Y., Pei, W.: Swarm Directions Embedded in Fast Evolutionary Programming. In: Proceedings of the 2002 Congress on Evolutionary Computation (2002) 1278-1283
15. Abbas, H.: The Self-Adaptive Pareto Differential Evolution Algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation (2002) 831-836
16. Bui, L., Shan, Y., Qi, F., Abbas, H.: Comparing Two Versions of Differential Evolution in Real Parameter Optimization. Technical Report TR-ALAR-200504009, The Artificial Life and Adaptive Robotics Laboratory, School of IT and Electrical Engineering, University of New South Wales, Canberra, Australia (2005)