Key Engineering Materials
ISSN: 1662-9795, Vols. 439-440, pp 315-320
doi:10.4028/www.scientific.net/KEM.439-440.315
© 2010 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.
Online: 2010-06-07

A Hybrid Differential Evolution Algorithm for Solving Function Optimization

Zhigang Zhou

Department of Computer Science and Technology, Dezhou University, China

zzg@dzu.edu.cn

Keywords: Differential evolution (DE); evolutionary algorithm; optimization.

Abstract. One of the key points resulting in the success of differential evolution (DE) is its mechanism of different mutation strategies for generating mutant vectors. In this paper, we also present a novel mutation strategy inspired by the velocity updating scheme of particle swarm optimization (PSO). The proposed approach is called HDE, which conducts the mutation strategy on the global best vector for each generation. Experimental studies on eight well-known benchmark functions show that HDE outperforms other three compared DE algorithms in most test cases.

RETRACTED

Introduction

Differential Evolution (DE) [1] is a recently proposed evolutionary technique, which has been shown to be a simple yet efficient evolutionary algorithm for many optimization problems [2]. Since the DE algorithm is simple, easy implementation and robustness, it has been applied to many real-world problems, such as image processing, signal processing, pattern recognition, and optimal design.

However, the performance of DE is still quite dependent on the setting of control parameters such as the mutation factor and the crossover probability according to both experimental studies and theoretical analyses. Although there are some suggestions for parameter settings, the interaction between the parameter setting and the optimization performance is still complicated and not completely understood. This is mainly because there is no fixed parameter setting that is suitable for different kinds of problems.

To tackle this problem, many researchers proposed different parameter setting mechanisms. The control parameter is altered by some deterministic rules without taking into account any feedback from the evolutionary search. One example is the time dependent change of the mutation rates proposed by Holland [3]. Gamperle *et al.* [4] evaluated different parameter settings for DE on the Sphere, Rosenbrock's and Rastrigin's functions. Their experimental results revealed that the global optimum searching capability and the convergence speed are very sensitive to the choice of control parameter F, and CR. Furthermore, a plausible choice of the population size ps is between $3D$ and $8D$, with the scaling factor $F = 0.6$ and the crossover rate CR in [0.3, 0.9]. Recently, the authors in [5] claim that $0.4 < F < 0.95$ with $F = 0.9$ is a good first choice. CR typically lies in (0, 0.2) when the function is separable, while in (0.9, 1) when the function's parameters are dependent. Liu and Lampinen [6] proposed a fuzzy adaptive DE (FADE), which uses a fuzzy logic controller to set the probability of mutation and crossover. Qin and Suganthan [7] presented a self-adaptive DE (SaDE) for numerical optimization, which focused on adaptation for parameter CR and mutation strategies of DE. Brest *et al.* [8] introduced self-adapting control parameter settings in DE (SADE). Yang *et al.* [9] introduced a neighborhood search strategy to DE (NSDE), which generates F from Gaussian and Cauchy distributed random numbers instead of predefining a constant F. Ali and Torn [10] introduced auxiliary population and automatic calculating of the amplification factor F.

In this paper, we propose a hybrid DE algorithm to improve the performance of DE. The proposed approach is called HDE, which employs a novel mutation strategy inspired by the velocity updating model of PSO. Experimental studies on eight well-known benchmark problems show that HDE obtains better performance when comparing with other three DE variants.

This article is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0)

Differential Evolution

The main idea of DE is not to mutate vector components by simply replacing their values by random values. Instead, two population mates are randomly selected whose weighted difference is added to a third randomly selected population member creating this way a mutant vector. Then, either a two-point crossover [1] or a multi-point crossover [1] is performed between the mutant vector and the current population member being considered. At the end, the new created offspring vector (trial vector) replaces the current considered vector in the next generation, if its fitness is better. Otherwise, the trial vector is discarded.

Let D and ps be the dimension of the problem and the population size, respectively. A vector member in the population can be defined by

$$X_{i,G} = \left(X_{1i,G}, X_{2i,G}, \ldots, X_{Di,G},\right) \tag{1}$$

where $i = 1, 2, \ldots ps$, $G = 1, 2, \ldots, MAX_G$ and MAX_G is the maximum number of generations.

Like genetic algorithms, DE also has three operators, mutation, crossover and selection. First, we create a mutant vector $V_{i,G}$. And then we recombine the mutant vector and the current vector to create a new trail vector $U_{i,G}$. At last, we compare the fitness value of $V_{i,G}$ with $U_{i,G}$, and select a better one as the new current vector in the next generation. The three operators are described as follows.

DE uses the difference between randomly selected individuals as the source of random variations for a third individual, referred to as the target individual. Trail solutions are generated by adding weighted differences vector. This process is referred as a mutation operator described as follows. For each vector $X_{i,G}$ in generation G, a mutant vector $V_{i,G}$ is defined by

$$V_{i,G} = X_{r1,G} + F(X_{r2,G} - X_{r3,G}) \tag{2}$$

where $X_{i,G}(i = 1, 2, \ldots ps)$ are solution vectors in generation G, ps is the population size, $i = 1, 2, \ldots ps$ and r_1, r_2, and r_3 are mutually different random integer indices selected from $\{1, 2, \ldots ps\}$.

After the mutation step, a new trail vector $U_{i,G} = \left(U_{1i,G}, U_{2i,G}, \ldots, U_{Di,G},\right)$ is generated by recombination of the mutant vector and the current vector

$$U_{ji,G} = \begin{cases} V_{ji,G}, & if\ rand_j(0,1) \le CR \vee j = k \\ X_{ji,G}, & otherwise \end{cases} \tag{3}$$

where CR is the predefined crossover probability, and $rand_j(0,1)$ is a random number within (0, 1) for the ith dimension and $k \in \{1, 2, \ldots, D\}$ is a random parameter index.

After the two steps a selection mechanism is used to choose a better vector between $U_{i,G}$ and $X_{i,G}$ to update the current vector in the next generation. For a minimization problem, the vector with smaller fitness value is better.

The Proposed Approach

Generally, the termination condition of an evolutionary algorithm is that the global best individual in the population finds the global optimum or the number of function evaluations reaches to the predefined maximum value. It can be found that the global best individual is important toward searching the global optimum. If the global best individual is trapped, the whole population may not find good solutions. To tackle this problem, we propose a novel mutation strategy on the global best individual. The motivation of the approach is inspired by the velocity updating model of PSO which utilizes the searching experiences of the previous best individuals and the global best individual.

In PSO, an individual in the population is called “particle”. Each particle has two vectors: position and velocity, which are updated as follows [11].

RETRACTED

$$V_i(t+1) = w * V_i(t) + c_1 * rand1() * (pbest_i - X_i(t)) + c_2 * rand2() * (Best - X_i(t)) \quad (4)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (5)$$

where X_i and V_i are the position and velocity of the ith particle, $pbest_i$ *and Best* are previous best particle of the ith particle and the global best particle found by all particles so far respectively, w is an inertia factor, *rand1*() and *rand2*() are two random numbers independently generated within the range of [0,1], and c_1 and c_2 are two learning factors.

Based on the velocity updating equation, we propose a novel mutation strategy as follows.

$$Best^* = a_1 * Best + a_2 * (Best - X_{i1}) + a_3 * (X_{i2} - X_{i1}) \quad (6)$$

where Best is the global best individual in the population, X_{i1} and X_{i2} are two different individuals, $i1$ and $i2$ are two different random number within [1, ps], $i1 \neq i2$, ps is the population size, a_1, a_2 and a_3 are three random numbers within [0,1], and $a_1 + a_2 + a_3 = 1$. The random numbers can be generated by

$$a_1 = rand(0,1), a_2 = rand(0,1), a_3 = rand(0,1) \quad (7)$$

$$sum = a_1 + a_2 + a_3 \quad (8)$$

$$a_1 = a_1 / sum, a_2 = a_2 / sum, a_3 = a_3 / sum \quad (9)$$

where *rand*(0,1) is a random number within [0,1].

The Framework of HDE Algorithm

```
Begin
  while NE < MAXNE do
    for i = 1 to ps do
      Generate trial individual according to Eq. 2 and Eq. 3;
      Calculate the fitness of the trial individual;
      Select the fitter one between Xi and the trail;
    end for
    Generate random numbers according to Eq. 7, Eq. 8 and Eq. 9;
    Conduct the mutation according to Eq. 6;
    If Best* is better than Best
      Replace Best with Best*;
    end if
  end while
End
```

Every iteration, we conduct the mutation strategy. If the individual *Best** after the mutation is better than the old *Best*, then replace the *Best* with *Best* *; otherwise keep the *Best* unchangeable. The application of the mutation strategy used in DE is described in the framework of the HDE algorithm, where ps is the population size, *Best* is the global best individual in the population, NE is the number of function evaluations, and MAX_{NE} is the maximum number of function evaluations.

Simulation Studies

Test Functions. In this paper, we test the proposed approach HDE on eight well-known benchmark problems, which were used in early studies [12]. All the functions used in this paper are to be minimized. The description of the benchmark functions and their global optima are listed as follows.

RETRACTED

$$f_1 = \sum_{i=1}^{D} x_i^2$$

where $x_i \in [-100, 100]$, D=30, and the global optimum is 0.

$$f_2 = \sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} x_i$$

where $x_i \in [-10, 10]$, D=30, and the global optimum is 0.

$$f_3 = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$$

where $x_i \in [-100, 100]$, D=30, and the global optimum is 0.

$$f_4 = \max_i \left(|x_i|, 1 \le i \le D \right)$$

where $x_i \in [-100, 100]$, D=30, and the global optimum is 0.

$$f_5 = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$$

where $x_i \in [-30, 30]$, D=30, and the global optimum is 0.

$$f_6 = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$$

where $x_i \in [-100, 100]$, D=30, and the global optimum is 0.

$$f_7 = \sum_{i=1}^{D} i x_i^4 + rand[0, 1)$$

where $x_i \in [-1.28, 1.28]$, D=30, and the global optimum is 0.

$$f_8 = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{| x_i |} \right)$$

where $x_i \in [-500, 500]$, D=30, and the global optimum is [illegible]569.5.

Comparison of HDE with DE. In this section, we compare the proposed approach HDE with classical DE on the ten test problems. For the sake of fair competition, we use the same parameter settings for HDE and DE. The population size *ps*, *CR* and *F* are set to 50, 0.9 and 0.5, respectively. Both DE and HDE use the same mutation strategy *DE*/*rand*/1. All the experiments in this paper are conducted 30 times with different random seeds, and the average results throughout the optimization runs are recorded.

Table 1. The results achieved by DE and HDE.

Functions	DE		HDE	
	Mean	Std Dev	Mean	Std Dev
f_1	1.58e-57	3.48e-57	5.57e-61	4.59e-61
f_2	4.98e-36	7.26e-36	6.49e-38	2.90e-38
f_3	7.63e-07	5.21e-07	5.14e-08	4.62e-08
f_4	7.24	4.82	0.902	0.83
f_5	22.68	6.36	3.75	4.29
f_6	0	0	0	0
f_7	6.61e-03	2.13e-03	1.48e-03	3.21e-03
f_8	-12095.7	332.5	-12451.2	85.6

The comparison results between HDE and DE are presented in Table 1, where "Mean" indicates the mean best function values, and "Std Dev" stands for the standard deviation. From the results, it can be seen that HDE outperforms DE in all test cases except for functions f_6. On this function, both HDE and DE achieve the same results.

Comparison of HDE with SaDE and NSDE. This section presents another comparative study to further verify the performance of HDE. The involved algorithm includes Self-adaptive DE (SaDE)

RETRACTED

and DE with neighborhood search (NSDE). The parameter settings of SaDE and NSDE are described in [13]. For IDE, the population size *ps*, *CR* and *F* are set to 50, 0.9 and 0.5, respectively. The initial population is generated uniformly at random in the search domain of the functions. The maximum number function evaluations (MAX_{NE}) are listed in Table 2.

Table 2. The results achieved by SaDE, NSDE and HDE.

Functions	MAX_{NE}	SaDE	NSDE	HDE
		Mean	Mean	Mean
f_1	150000	7.49e-20	7.76e-16	5.57e-61
f_2	150000	6.22e-11	4.51e-10	6.49e-38
f_3	150000	1.12e-18	1.06e-14	5.14e-08
f_4	150000	2.96e-02	2.54e-02	0.902
f_5	500000	2.10e+01	1.24e+01	3.75
f_6	150000	0	0	0
f_7	150000	7.58e-03	1.20e-02	1.48e-03
f_8	150000	-12569.5	-12569.5	-12551.2

The average results over 30 trails of HDE, SaDE and NSDE are presented in Table 2, where "Mean" indicates the mean best function values, and "Std Dev" stands for the standard deviation. The results of SaDE and NSDE are taken form Table 2 and 3 in [13]. From the results, it cane be seen that HDE outperforms SaDE and NSDE on functions f_1, f_2, f_5 and f_7, and significantly improve the results on functions f_2 and f_5. For function f_6, all the three algorithms obtain the same performance. SaDE achieves better results than NSDE and HDE on function f_3, while NSDE outperforms the other two DE algorithms on function f_4. Both SaDE and NSDE achieve better results than IDE on function f_8.

Conclusion

This paper presents a hybrid DE algorithm to improve the performance of DE. The proposed approach is called HDE, which employs a novel mutation strategy inspired by the velocity updating model of PSO. Every generation, we conduct the mutation on the global best individual, and select a better one between the mutant and the best as the current best individual. Experimental studies on eight well-known benchmark problems show that HDE obtains better performance than classical DE, SaDE and NSDE in most test cases.

However, HDE and other compared DE algorithms falls into local minima on function f_5. It suggests that the proposed mutation strategy is not suitable for all kinds of problems. It is worth to introduce more efficient strategies into DE to improve its performance in the future work.

References

[1] R. Storn and K. Price, Differential evolution--A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization, (1997), pp. 341

[2] J. Vesterstrøm and R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Proceedings of the 2004 Congress on Evolutionary Computation, vol. 2 (2004), pp. 1980.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press, (1975)

[4] R. Gamperle, S. D. Muller and A. Koumoutsakos, Parameter study for differential evolution, in WSEAS NNA-FSFS-EC 2002, Interlaken, Switzerland, (2002)

[5] J. Ronkkonen, S. Kukkonen and K. V. Price, Real parameter optimization with differential evolution, in Proc. Congr. Evol. Comput. (CEC), vol. 1 (2005), pp. 506

RETRACTED

[6] J. Liu, and J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Computing-A Fusion of Foundations, Methodologies and Applications, (2005), pp. 448

[7] A. K. Qin, and P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization. Proc. Congr. Evol. Comput., (2005), pp. 1785

[8] J. Brest, S. Greiner, B. Boskovic, M. Mernik and V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput., (2006), pp. 646

[9] Z. Yang, J. He, and X. Yao, Making a difference to differential evolution, Advance in Metaheuristics for Hard Optimization, (2008), pp. 397

[10] M. M. Ali and A. Torn, Population set-based global optimization algorithms: some modifications and numerical studies, Comput. Oper. Res., (2004), pp. 1703

[11] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of the Conference on Evolutionary Computation, IEEE Press, Piscataway, (1998), pp. 69

[12] X. Yao, Y. Liu and G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput., vol. 3(1999), pp. 82

[13] Z. Yang, K. Tang and X. Yao, Self-adaptive differential evolution with neighborhoodsearch, In Proceedings of Congress on Evolutionary Computation, (2008), pp. 1110

RETRACTED