A Comparative Study of Various Strategies in Differential Evolution

Urszula Boryczka,[1] Przemysław Juszczuk, [2] Leszek Kłosowicz [3]

[1] University of Silesia, Institute of Computer Science, Sosnowiec, Poland,
e-mail: urszula.boryczka@us.edu.pl
[2] e-mail: przemyslaw.juszczuk@gmail.com
[3] e-mail: leszekklosowicz@gmail.com

Abstract. This paper presents a comparison of various strategies of differential evolution. Differential evolution (DE) is a simple and powerful optimization method, which is mainly applied to numerical optimization and many other problems (for example: neural network train, filter design or image analysis). The comparison of various modifications (named strategies) of DE algorithm allows to choose the algorithm version which is best adjusted to desirable requirements. Three parameters are tested: speed, accuracy and completeness. The first section of this article presents general optimization problem and says a little about methods used to function optimization. The next section describes differential evolution — basic algorithm is presented. Two different crossover methods, process of initial population creation and basic mutation schema are described. The third section describes the most popular DE strategies. In the fourth section a new modification (called λ–modification) of DE algorithm is presented. Next section provides basic information about four test functions and differential evolution parameters used in research. The paper presents then summary and final conclusions.

1 Introduction

In the last decades many deterministic and stochastic methods of an optimization problem have been developed. However, no universal technique which could give the good results for all optimization problems has been found yet. One of these problems is numerical optimization which is used to test many new methods. Its aim is to find solution:

$$x^* = \arg \min_{x \in X} F(x)$$

for the quality function $F(x)$, where X denotes a set of feasible solution and $x \in X$ is a vector $x = [x^1, x^2, \ldots, x^{n_F}]$. n_F is a number of function dimensions.

A major impediment for numerical optimization is multimodality of function $F(x)$, since the found solution may not be the global optimum. Different approaches are used to find the global optimum for multimodal function. Since deterministic methods are often too weak or too slow, stochastic methods are used. They often take inspiration from biological or social behaviour. Most of stochastic techniques base on the population of individuals which is improved in subsequent iterations of algorithm. At the beginning

of algorithm the individuals are distributed randomly in search space. In each algorithm iteration the population is improved and individuals converge to one or more optima. Differential evolution is one of such stochastic methods and it is described in point 2.

2 Differential Evolution

Differential evolution is a stochastic technique which was developed by K. Price and R. Storn in 1995 [10]. It is a population–based optimization method which can be used for example to numerical optimization [9], neural network train [6], filter design [8] or image analysis [12]. DE is conceptually similar to the simple evolutionary algorithm, but there are also quite big differences. First of all, the mutation is the major genetic operator. It is not a trivial process and it also provides the algorithm convergence. Moreover, the mutation is performed before the crossover process. Although the basic DE algorithm is very powerful, many valuable modifications have been introduced (for example [11], [5] and [13]).

The pseudocode of the general DE algorithm [2] is presented in Algorithm 1.

Algorithm 1: General Differential Evolution Algorithm

```
1  t = 0
2  Initialize the population P(0)
3  while stop condition is false do
4      foreach individual x_i ∈ P(t) do
5          Evaluate the fitness F(x_i(t))
6          Create the trial vector u_i(t) by applying the mutation operator
7          Create an offspring x'_i(t) by applying the crossover operator
8          if f(x'_i(t)) is better then f(x_i(t)) then
9              Add x'_i(t) to P(t + 1)
           else
10             Add x_i(t) to P(t + 1)
11     t = t + 1
12 Return the solution (the individual with the best fitness)
```

The DE algorithm begins with the initialization of population $P(0)$ which consist of n_X individuals. The initialization consists in the random distribution of individuals. The population should be distributed uniformly, which provides a good sampling of search space. In the main loop of the algorithm some actions which should improve the population are performed. For the each individual (vector) $x_i(t)$, firstly, its fitness is evaluated. Then the mutation process follows. It consist in the creation of the trial vector $u_i(t)$:

$$u_i(t) = x_{i_1}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$$

Individual $x_i(t)$ is named the target vector. $(x_{i_2}(t) - x_{i_3}(t))$ is a differential vector created from the two random individuals $x_{i_2}(t)$ and $x_{i_3}(t)$. The differential vector gives information about the fitness landscape and in this way the search process is directed. $F \in (0, \infty)$ is a scaling factor which controls the weight of the differential vector when

the trial vector is created. Other methods of the creation of the differential vector are mentioned in point 3. The crossover process consists in the creation of a new individual (offspring) $x'_i(t)$. Some of the elements of vector $x'_i(t)$ come from individual $x_i(t)$ and the others from the trial vector $u_i(t)$. This process can be expressed as follows:

$$x'^j_i(t) = \begin{cases} u^j_i(t) & \text{if } j \in J \\ x^j_i(t) & \text{in other case} \end{cases}$$

The set J contains the indexes which indicate the elements of the trial vector $u_i(t)$. Usually, to determine the set J the binomial or exponential method is used (Algorithm 2 and 3 respectively).

Algorithm 2: Binomial method of the set J determination

```
j* ~ U(1, n_F)
J = J ∪ {j*}
foreach j ∈ {1, 2, ..., n_F} do
    if U(0,1) < CR and j ≠ j* then
        J = J ∪ {j}
```

Algorithm 3: Exponential method of the set J determination

```
J = {}
j ~ U(0, n_F − 1)
repeat
    J = J ∪ {j + 1}
    j = (j + 1) mod n_F
until U(0,1) ≥ CR or |J| = n_F
```

These simple algorithms provide random choice of vector indexes to the set J. The choice of each index j is performed with some probability $CR \in [0, 1]$. The greater the value of CR, the bigger the chance that the index j will be added to the set J. After the crossover process the offspring is compared with its parent. Next, the better one of these individuals is added to the new population. The last step of the algorithm is the increment of the generation counter t. The best individual from the last generation is the result of the DE algorithm.

3 DE strategies

Like for many other evolutionary algorithms, also for differential evolution many modifications were developed. The most often modified elements of DE algorithm are:

- a method of target vector selection (denoted as x),
- a number of differential vectors used for trial vector creation (y),
- a crossover method (z).

Each of the DE algorithms can be described by strategy DE/x/y/z. Most often described DE strategies are [2]:

- Strategy I: DE/rand/1/z
  It is the DE strategy which characterises the basic DE algorithm. It uses random (rand) selection of target vector and only one differential vector for the trial vector creation. Like other strategies also this one can use binomial (bin) or exponential (exp) crossover. In the DE/rand/1/z strategy the trial vector is calculated form the equation:
  $$u_i(t) = x_{i_1}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$$
- Strategy II: DE/best/1/z
  In this strategy the best individual $\hat{x}(t)$ from the current population is the target vector. In this case the trial vector $u_i(t)$ is calculated as follows:
  $$u_i(t) = \hat{x}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$$
- Strategy III: DE/x/n_v/z
  n_v signifies the number of differential vectors used for trial vector $u_i(t)$ creation. Trial vector in this strategy is calculated from the following equation:
  $$u_i(t) = x_{i_1}(t) + F \cdot \sum_{k=1}^{n_s} (x_{i_2,k}(t) - x_{i_3,k}(t))$$
  $(x_{i_2,k}(t) - x_{i_3,k}(t))$ denotes the k^{th} differential vector ($k \in \{1, 2, \ldots, n_v\}$). The larger the value of n_v, the better the search space exploration. Unfortunately, the more differential vectors are used in the mutation, the greater the complexity of the algorithm.
- Strategy IV: DE/rand to best/n_v/z
  In DE/rand to best/n_v/z strategy the target vector is calculated partly from the best individual in the current population and partly from the random individual:
  $$u_i(t) = \gamma\hat{x}(t) + (1-\gamma)x_{i_1}(t) + F \cdot \sum_{k=1}^{n_s} (x_{i_2,k}(t) - x_{i_3,k}(t))$$
  Parameter $\gamma \in [0, 1]$ denotes which of these individuals is more importance. The closer to zero the value of γ, the more important the random individual $x_{i_1}(t)$ (exploration is favoured). In other case, the best individual $\hat{x}(t)$ in the current population has more weight (exploitation is favoured). If $\gamma \approx 0.5$, then both individuals have similar influence on the trial vector $u_i(t)$.
- Strategy V: DE/current to best/$1 + n_v$/z
  This strategy requires at least two differential vectors to create the trial vector $u_i(t)$. The first of them, $(\hat{x}(t) - x_i(t))$, is calculated from the best individual $\hat{x}(t)$ in the current population and from the target vector $x_i(t)$. The rest of differential vectors $(F \cdot (x_{i_1,k} - x_{i_2,k}))$ are created from two random individuals $x_{i_1}(t)$ and $x_{i_2}(t)$. The trial vector $u_i(t)$ is calculated as follows:
  $$u_i(t) = x_i + F \cdot (\hat{x}(t) - x_i(t)) + F \cdot \sum_{k=1}^{n_v} (x_{i_1,k}(t) - x_{i_2,k}(t))$$

4 λ–modification

λ–modification has been introduced to improve the accuracy and speed of DE algorithm. It is conceptually very simple and consists in multiplication of the target vector by parameter λ when the trial vector $u_i(t)$ is calculated. For DE strategies described in point 3 the trial vector is determined as follows:

- Strategy I′: $u_i(t) = \lambda \cdot x_{i_1}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$,
- Strategy II′: $u_i(t) = \lambda \cdot \hat{x}(t) + F \cdot (x_{i_2}(t) - x_{i_3}(t))$,
- Strategy III′: $u_i(t) = \lambda \cdot x_{i_1}(t) + F \cdot \sum_{k=1}^{n_s}(x_{i_2,k}(t) - x_{i_3,k}(t))$,
- Strategy IV′: $u_i(t) = \lambda \cdot (\gamma\hat{x}(t) + (1-\gamma)x_{i_1}(t)) + F \cdot \sum_{k=1}^{n_s}(x_{i_2,k}(t) - x_{i_3,k}(t))$,
- Strategy V′: $u_i(t) = \lambda \cdot x_i + F \cdot (\hat{x}(t) - x_i(t)) + F \cdot \sum_{k=1}^{n_v}(x_{i_1,k}(t) - x_{i_2,k}(t))$.

Parameter λ (arbitrarily set to 0.5) significantly reduces the distance which is covered by the individuals in the subsequent iterations. It results in convergence improvement, which is especially desired in the final stage of calculations. This simple modification yields surprisingly good results, which has been presented in point 5.

5 Experimental study

The aim of this research work is to determine how various DE strategies influence the results for numerical optimization tasks. Three parameters are tested: speed (SPD) – the average number of algorithm iterations which is needed to achieve the accuracy not worse then the assumed accuracy threshold; error (ERR) – the average calculation error for certain number of iterations; completeness (CMP) – the average number of DE algorithm runs in which the assumed accuracy is achieved (for certain number of iterations). DE algorithm has the following parameters: binomial crossover, crossover parameter $CR = 0.5$ [9], mutation parameter $F = 0.7$ [9], population size $n_X = 100$, number of differential vectors $n_v = 2$ (for strategy III), parameter $\gamma = 0.5$ (for strategy IV), $\lambda = 0.5$. During tests for ERR and CMP parameters, the number of algorithm iterations amounts to 2000. The following four functions (all 20 dimensional) are tested:

- De Jong: $F_1(x) = \sum_{i=1}^{20} x_i^2$ for $x_i \in [-5.12, 5.12]$
  $f(x^*) = 0$, accuracy threshold (for the parameter ERR): 0.5%
- Rosenbrock: $F_2(x) = \sum_{i=1}^{19}[(1-x_i)^2 + 100(x_{i+1} - x_i^2)^2]$ for $x_i \in [-2.048, 2.048]$
  $f(x^*) = 0$, accuracy threshold: 0.5%
- Rastrigin: $F_3(x) = 200 + \sum_{i=1}^{20}[x_i^2 - 10\cos(2\pi x_i)]$ for $x_i \in [-5.12, 5.12]$
  $f(x^*) = 0$, accuracy threshold: 0.5%
- Ackley: $F_4(x) = -20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{20}\sum_{i=1}^{20} x_i^2}\right) - \exp\left(\frac{1}{20}\sum_{i=1}^{20}\cos(2\pi x_i)\right) + 20 + \exp(1)$ for $x_i \in [-32.768, 32.768]$
  $f(x^*) = 0$, accuracy threshold: 0.5%

For each test function and each of the DE strategies mentioned in point 3, DE algorithm is run 1000 times and the obtained results are averaged. The test results (without modification) for parameters SPD, ERR and CMP are placed in Tables 1, 3 and 5 respectively. Tables 2, 4 and 6 show the obtained test results for DE algorithm with applied λ–modification.

Table 1. Test results for SPD parameter (without modification)

Function	Strategy I	Strategy II	Strategy III	Strategy IV	Strategy V
De Jong	582	82	1468	43	58
Rosenbrock	1812	136	6632	60	82
Rastrigin	>10000	654	>10000	253	656
Ackley	2821	147	6426	110	147

Table 2. Test results for SPD parameter (with λ–modification)

Function	Strategy I′	Strategy II′	Strategy III′	Strategy IV′	Strategy V′
De Jong	78	50	351	42	40
Rosenbrock	112	69	684	59	92
Rastrigin	1126	1035	>10000	254	8231
Ackley	209	127	941	109	161

Table 3. Test results for ERR parameter (without modification)

Function	Strategy I	Strategy II	Strategy III	Strategy IV	Strategy V
De Jong	0.01%	3.6%	2.8%	$\approx 0\%$	$\approx 0\%$
Rosenbrock	1.1%	0.2%	4.9%	0.3%	0.4%
Rastrigin	83.1%	0.2%	91%	$\approx 0\%$	$\approx 0\%$
Ackley	54.3%	0%	91.2%	$\approx 0\%$	$\approx 0\%$

Table 4. Test results for ERR parameter (with λ–modification)

Function	Strategy I′	Strategy II′	Strategy III′	Strategy IV′	Strategy V′
De Jong	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$
Rosenbrock	0.4%	0.4%	0.4%	0.4%	0.4%
Rastrigin	53.0%	0.1%	67.0 %	$\approx 0\%$	51.85%
Ackley	$\approx 0\%$	0%	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$

6 Conclusions

After the analysis of the test results of experiments with the use of the various DE strategies (without λ–modification) the following conclusions may be reached:

Table 5. Test results for CMP parameter (without modification)

Function	Strategy I	Strategy II	Strategy III	Strategy IV	Strategy V
De Jong	46%	48%	49%	39%	45%
Rosenbrock	35%	67%	49%	45%	44%
Rastrigin	—	54%	—	54%	58%
Ackley	52%	55%	36%	52%	48%

Table 6. Test results for CMP parameter (with λ–modification)

Function	Strategy I′	Strategy II′	Strategy III′	Strategy IV′	Strategy V′
De Jong	100%	100%	100%	100%	100%
Rosenbrock	100%	100%	100%	100%	100%
Rastrigin	—	41%	—	100%	—
Ackley	100%	100%	100%	100%	100%

- the strategies II, IV and V (all use the best individual to create the trial vector) significantly reduce the calculation error (Table 3) and also work much faster then others (Table 1),
- the strategy IV gives the best result for the parameter SPD (Table 1),
- for the all functions good results for the parameter CMP gives the strategy II,
- the strategies I and III are weak and give poor results.

λ–modification introduces the following changes:

- it increases the speed of the calculations for almost all test function (Table 2) — only Rastrigin Function and Strategy V′ behave otherwise,
- it significantly reduces the calculation error (Table 4),
- it strongly increases the research completeness (Table 6),
- only for Rastrigin Function λ–modification sometimes lowers the optimization quality,
- λ–modification increases the optimization quality regardless of the accepted strategy.

This research shows that neither of DE strategies (without λ–modification) provides best results for all test functions simultaneously. However, the DE/rand to best/n_v/z strategy seems to be the best. When λ–modification is not applied, this strategy gives the best result for the parameter SPD and also in many cases for the parameter ERR. DE algorithm which uses this strategy and λ–modification gives the best results for all test function. This strategy uses the best (assurance of exploitation) and a random (assurance of exploration) individual to calculate the trial vector. The use of n_v differential vectors gives a lot of information about the fitness landscape and properly directs the search process.

λ–modification presented in this paper requires further research (up to the present it has been tested only for one value $\lambda = 0.5$). Firstly, the best value (or values range) for parameter λ should be determined and then the method for dynamical changes of its value or its self-adaptation should be worked out.

Bibliography

[1] S. Das, A. Konar, and U. K. Chakraborty. Two improved differential evolution schemes for faster global search. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 991–998, New York, NY, USA, 2005. ACM.

[2] A. Engelbrecht. *Computational Intelligence: An Introduction.* Halsted Press, New York, NY, USA, 2002.

[3] R. Joshi and A. C. Sanderson. Minimal representation multisensor fusion using differential evolution. In *CIRA '97: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, page 266, Washington, DC, USA, 1997. IEEE Computer Society.

[4] J. Lampinen and I. Zelinka. Mixed variable non-linear optimization by differential evolution. In *Zlin, Czech Republic. Technical University of Brno, Faculty of Technology Zlin, Department of Automatic Control*, pages 45–55, 1999.

[5] J. Liu and J. Lampinen. A differential evolution based incremental training method for rbf networks. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 881–888, New York, NY, USA, 2005. ACM.

[6] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis. Effective backpropagation training with variable stepsize. *Neural Netw.*, 10(1):69–82, 1997.

[7] K. Price, R. Storn, and J. Lampinen. *Differential evolution – A practical Approach to Global Optimization.* Springer, 2005.

[8] R. Storn. Differential evolution design of an iir-filter. In *in IEEE International Conference on Evolutionary Computation ICEC96*, pages 268–273. IEEE Press, 1996.

[9] R. Storn. On the usage of differential evolution for function optimization. In *NAFIPS'96*, pages 519–523. IEEE, 1996.

[10] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, 1997.

[11] J. Tvrdík. Differential evolution: Competitive setting of control parameters. In *Proceedings of the international multiconference on computer science and information technology*, pages 207–213, 2006.

[12] Kasemir K. U. and K. Betzler. Detecting ellipses of limited eccentricity in images with high noise levels. *Image and Vision Computing*, 21:221–227(7), 10 February 2003.

[13] D. Zaharie. A multipopulation differential evolution algorithm for multimodal optimization. In *Proceedings of Mendel 2004, 10th International Conference on Soft Computing*, pages 17–22, Brno, Czech Republic, 2004.