ON SEARCH CAPABILITIES OF THE DIFFERENTIAL EVOLUTION ALGORITHM
PAR DIFERENCIĀLĀS EVOLŪCIJAS ALGORITMA MEKLĒŠANAS IESPĒJĀM

Anatoly Sukov, Decision Support Systems Group, Institute of Information Technology,
Riga Technical University,1 Kalku Street, Riga LV–1658, Riga, Latvia

Abstract. *This paper examines the algorithm of differential evolution that has appeared rather recently. This algorithm ascribed by its developers to a class of evolutionary algorithms is a comparatively non–complicated technique of solution search as applied to multiparameter optimisation tasks. Nevertheless, there are two essential factors preventing from wide application of the considered solution search technique. One of them lies in the principle of coding vectors (variables) that constitute a population the algorithm works with. The second problem is of pure technical character: in the process of search, stagnation occurs, or impossibility to find new solutions, when there is no optimal solution in the population and the vectors available are not heterogeneous. Besides studying search possibilities (limitations) of the differential evolution, some ways to cope with the problem of stagnation so as to raise the performance of the algorithm are also suggested.*

1. Introduction

Along with widely known genetic algorithms, other evolutionary algorithms exist (Bentley, 1999) that have been developed to solve optimisation tasks. Among them the differential evolution (DE) can be mentioned. The main criterion as to why this algorithm can be ascribed to the class of evolutionary ones is the presence of the respective concepts and solution search principles in it. The DE algorithm comprises individuals (vectors), population, crossover and mutation operators.

DE greatly differs from the standard genetic algorithm (Goldberg, 1989) in that it solely uses integer or real numbers as evolution objects. This means that a fenotype (solution) and genotype (solution representation in the algorithm) are identical. Respectively, the space of solutions and space of search are also identical. This approach to coding has both positive and negative features. DE is adapted to manipulating really integer solutions but at the same time is restricted by this application area (in turn, GA is known to be universal regarding this aspect). If we turn to the role distribution of evolutionary operators of mutation and crossover, certain differences can also be found. In the GA, crossover is primary, whereas in DE mutation is primary. The above differences in the algorithms demonstrate the similarity of DE and evolutionary strategies. DE differs from evolutionary strategies in that the mutation of the directed vector of standard deviations (Bäck and Schwefel, 1995) that are responsible for mutations of the vector of values can be observed in the latter. The task of this study is to examine the behaviour of DE with an applied optimisation task. With this, major attention will be paid to stagnation (the lack of progress in search under the absence of complete convergence) that was recognised as the main shortcoming of the algorithm (Lampinen and Zelinka, 2000).

2. Parameters and principles of the DE algorithm

The interpretation of DE described below is a *DE/rand/1/bin* scheme (Lampinen and Zelinka, 2000). In this algorithm the processing of real integer or mixed integer/real integer variables constituting a vector of variables of the optimised objective function takes place

ISBN 9984 - 585 - 36 - 0

$$f(X): R^D \rightarrow R,$$
$$X=(x_1, ..., x_D), X \in R^D$$

where X determines a vector consisting of D parameters of the objective function. Normally each of parameters has its limits, *lower* and *upper,* respectively, $x^{(L)}$ and $x^{(U)}$:
$x_j^{(L)} \leq x_j \leq x_j^{(U)}, \quad j=1, ..., D.$

Like other evolutionary algorithms, DE manipulates a population, P_G, consisting of NP real integer vectors $X_{i,G}$, where i defines an individual belonging to the population but G defines a generation (epoch) to which the population belongs:

$$P_G = X_{i,G}, \quad i=1, ..., NP, \quad G=1, ..., G_{max},$$
$$X_{i,G} = x_{j,i,G}, \quad i=1, ..., NP, \quad j=1, ..., D.$$

On the basis of the existing variable limits, the initialisation of the initial (zero) population occurs:

$$P_0 = x_{j,i,0} = rand_{j,i}[0;\ 1]^*(x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}, \quad i=1, ..., NP, \quad j=1, ..., D,$$

where $rand_{j,i}[0;\ 1]$ is an uniformly distributed random variable within the limits [0.0; 1.0], that is selected anew for each j and i. Starting from the first generation, a set of vectors (chromosomes) of the current population, P_G, participates in the selection and formation (on the basis of random selection) of trial vectors for next population, P_{G+1}. A population of trial vectors, $P'_{G+1}=U_{i,G+1}=u_{j,i,G+1}$, is created as follows:

$$u_{j,i,G+1}= \begin{cases} v_{j,i,G+1}= x_{j,r3,G} + F^*(x_{j,r1,G} - x_{j,r2,G}) \text{ IF } rand_{j,i}[0;\ 1) \leq CR \vee j=k, \\ x_{j,i,G}, \end{cases}$$

where $i=1, ..., NP, \quad j=1, ..., D$;
$k \in \{1, ..., D\}$ is a parameter randomly selected one time for each i;
$r1, r2, r3 \in \{1, ..., NP\}$ are selected at random, but $r1 \neq r2 \neq r3 \neq i$;
$CR \in [0;\ 1], F \in (0;\ 1+]$.

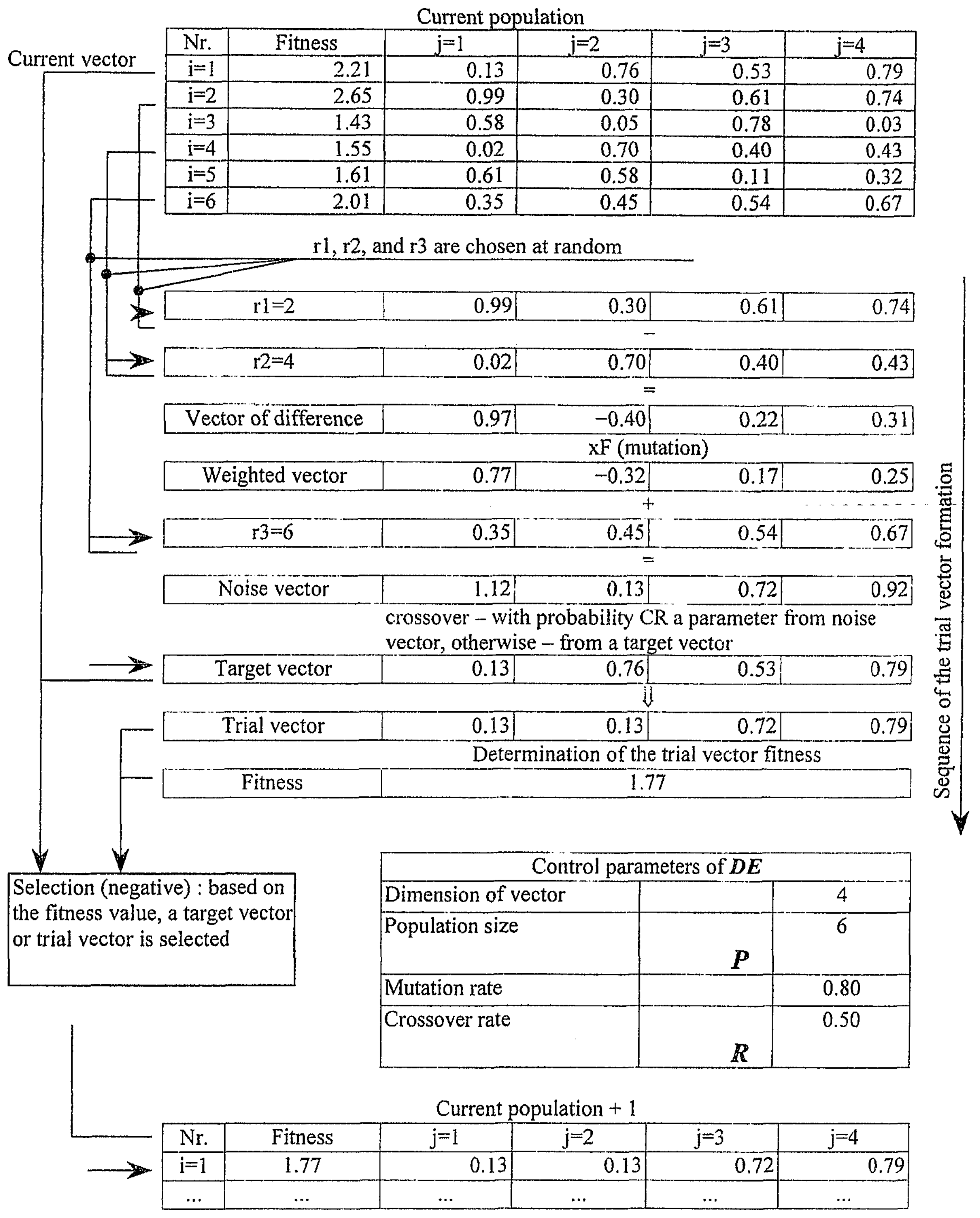
Random indexes: $r1$, $r2$, and $r3$ are determined for each i (that is for each chromosome anew). The task of index k is to prevent from the coincidence of $v_{i,G+1}$ and $x_{i,G}$. An example of the trial vector generation is shown in Fig. 1. CR and F are control parameters of DE.

Control parameters of the algorithm: D, NP, F, and CR are assigned before the algorithm starts working and remain constant until it stops (the standard procedure). The main stopping criteria are the following: either achieving the time limit for the evolutionary solution search or the situation when $G=G_{max}$.

A population for the next generation, , P_{G+1}, is created of the current population, P_G, and of a population of trial vectors, $U_{i,G+1}$, according to the following deterministic rule (see also Fig.1):

$$X_{i,G+1} = \begin{cases} U_{i,G+1}= u_{j,i,G+1} \quad \text{IF } f(U_{i,G+1}) \leq f(X_{i,G}), \\ X_{i,G},. \end{cases}$$

ISBN 9984 - 585 - 36 - 0


Current vector
r1, r2, and r3 are chosen at random
−
=
xF (mutation)
+
=
crossover – with probability CR a parameter from noise vector, otherwise – from a target vector
⇓
Determination of the trial vector fitness
Sequence of the trial vector formation
Selection (negative) : based on the fitness value, a target vector or trial vector is selected

Current population

Nr.	Fitness	j=1	j=2	j=3	j=4
i=1	2.21	0.13	0.76	0.53	0.79
i=2	2.65	0.99	0.30	0.61	0.74
i=3	1.43	0.58	0.05	0.78	0.03
i=4	1.55	0.02	0.70	0.40	0.43
i=5	1.61	0.61	0.58	0.11	0.32
i=6	2.01	0.35	0.45	0.54	0.67

r1=2	0.99	0.30	0.61	0.74
r2=4	0.02	0.70	0.40	0.43
Vector of difference	0.97	−0.40	0.22	0.31
Weighted vector	0.77	−0.32	0.17	0.25
r3=6	0.35	0.45	0.54	0.67
Noise vector	1.12	0.13	0.72	0.92
Target vector	0.13	0.76	0.53	0.79
Trial vector	0.13	0.13	0.72	0.79
Fitness	1.77			

Control parameters of ***DE***		
Dimension of vector		4
Population size	***P***	6
Mutation rate		0.80
Crossover rate	***R***	0.50

Current population + 1

Nr.	Fitness	j=1	j=2	j=3	j=4
i=1	1.77	0.13	0.13	0.72	0.79
...	...	...	...	...	...

Fig. 1. Formation of individuals for the next population. The optimized function is $f(X)=x_1+x_2+x_3+x_4$

From this rule it follows that selection in DE is of negative character as the worst solutions are in essence removed and replaced by the best ones under compulsion. As opposite

ISBN 9984 - 585 - 36 - 0

to it, the positive or standard selection foresees selecting pairs for crossover in the same way as it happens, say, in simple genetic algorithms.

3. Case study

To make the experiments, a function of two variables was chosen (see Fig.2) that looks like:

$$f(x_1, x_2) = 0.5 - \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001 \cdot \left(x_1^2 + x_2^2\right)\right)^2}.$$

It is evident that the global maximum of this function is at point (0, 0), that is f(0, 0)=1. Since the task of DE will be to find just the maximum of the function, this solution (at the beginning of co–ordinates) does not seem to be effective from the viewpoint of the task complexity. Due to this the function was changed as follows:

$$f(x_1, x_2) = 0.5 - \frac{\sin^2\left(\sqrt{(x_1 - 18.171)^2 + (x_2 + 40.225)^2}\right) - 0.5}{\left(1 + 0.001 \cdot \left((x_1 - 18.171)^2 + (x_2 + 40.225)^2\right)\right)^2}.$$

It means that the solution is a point (18.171, − 40.225).

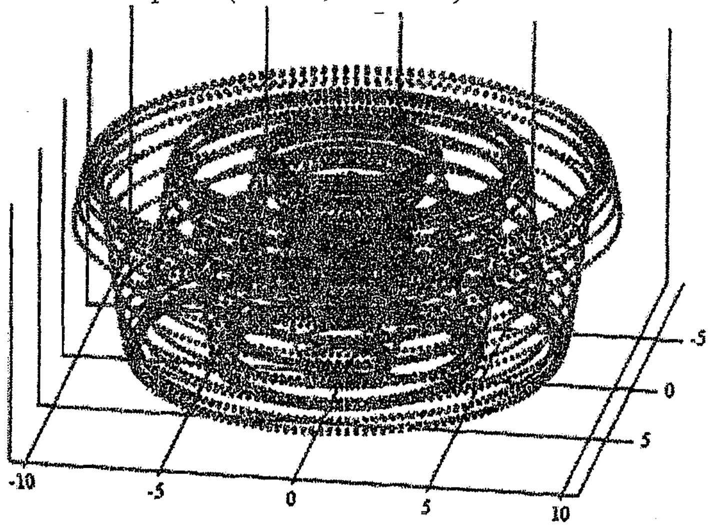


Fig.2. Graph of function f(x₁, x₂)

4. Experiments

The results of the experiments performed are given in Table 1. The space of search (initialisation limits) for all the experiments and each variable are similar: $x_j^{(L)}=-63.0$, $x_j^{(U)}=63.0$. The aim of each experiment was to increase the effectiveness of the algorithm based on the results of the previous experiment. For this, the effectiveness of each (G+1)th iteration was calculated by formula

ISBN 9984 - 585 - 36 - 0

$$\frac{\left[\sum_{i=1}^{NP} f(X_{i,G+1}) - \sum_{i=1}^{NP} f(X_{i,G})\right]}{\sum_{i=1}^{NP} f(X_{i,G})}.$$

By this expression one can determine an increment of the average fitness value, that is the measure of effectiveness. Table 1, in its turn, shows average efficiency values for each of algorithms, i.e. the mean result after 200 generations or after occurrence of the complete convergence when the average fitness value equals the maximum value.

Table 1.

Control parameters and results of different DE

Algorithm		Results after 200 iterations	
Experiment 1 (standard DE)			
Size of population, NP	150	Maximum fitness	0.99891
Mutation, F	0.9	Average fitness	0.99026
Crossover rate, CR	0.9	1'st best result (generation)	no
		Complete convergence (generation)	no
		Average effectiveness values	0.34617
Experiment 2 (standard DE)			
Size of population, NP	150	Maximum fitness	1.00000
Mutation, F	0.5	Average fitness	1.00000
Crossover rate, CR	0.5	1'st best result (generation)	176
		Complete convergence (generation)	185
		Average effectiveness value	0.3713
Experiment 3 (standard DE)			
Size of population, NP	50	Maximum fitness	1.00000
Mutation, F	0.5	Average fitness	1.00000
Crossover rate, CR	0.5	1'st best result (generation)	157
		Complete convergence (generation)	165
		Average effectiveness value	0.43744
Experiment 4 (modified DE)			
Size of population, NP	50	Maximum fitness	1.00000
Mutation, F	?	Average fitness	1.00000
Crossover rate, CR	0.5	1'st best result (generation)	125
		Complete convergence (generation)	138
		Average effectiveness value	0.53731

As a result of the first three experiments, the adjustment of control parameters of the standard DE was made. More particularly, in the first and in the second experiment optimal values of mutation and crossover rate have been derived empirically. Respectively, the mean effectiveness in the second experiment was higher than in the first one. The third experiment has shown a possibility of using a less population size (three times less) that also yielded an increase in the algorithm's effectiveness. The computational costs were also three times less. In the fourth experiment the standard DE was varied. The variations produced a positive effect. The operator modified was mutation, speaking more precisely, the size of mutation. In the ordinary algorithm this is normally a constant value that is assigned for the whole time of evolution process. In the last experiment the size of mutation was varied (a new value was generated) every time when the value of effectiveness was dropping as compared to that at the previous generation.

ISBN 9984 - 585 - 36 - 0

5. Conclusions

The task of the DE algorithm effectiveness raise, set in this study, was solved both by the standard and a new approach.

- Standard approach: adjustment of control parameters, CR, F, and NP.
- New approach: controlled variation of the mutation size, i.e. this is not a mutation constant any more but the value adjusted by the algorithm itself.

It is natural that at this moment the "adjustability" of the algorithm is of dual character. From the one side the algorithm knows for sure when to change the size of mutation (if the effectiveness drops), from the other side, however, the variation made is of random character. Practical results, however, show that the introduction of such additional randomness has the advantage over the standard algorithm.

As a whole, based on the results of the study it is possible to conclude about the possibility of using the adjustable mutation in DE instead of the existing standard (static) operator. Further research in this area will be oriented towards the development of less randomised mechanism of mutation size variation.

References

1. Bäck T. and Schwefel H.–P. (1995). Evolution Strategies I: Variants and their computational implementation. Genetic Algorithms in Engineering and Computer Science, editors Periaux J. and Winter G. John Wiley & Sons Ltd.
2. Bentley P.J. (1999). An Introduction to Evolutionary Design by Computers. In: Evolutionary Design by Computers (Bentley P. J.,Ed.) Morgan Kaufmann, p. 1–73.
3. Goldberg, D. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Reading: Addison–Wesley.
4. Lampinen J. and Zelinka I. (2000). On Stagnation of the Differential Evolution Algorithm. Proceedings of MENDEL 2000, 6th International Conference on Soft Computing. – Brno, June 7–9, 2000, p. 76–83.

JĒDZIENU VISPĀRINĀŠANAS ALGORITMA CORA SAVIENOŠANA AR LĒMUMU KOKU ĢENERĒŠANAS ALGORITMU C4.5 INTEGRATION OF CONCEPTS GENERALISATION ALGORITHM CORA WITH DECISION TREE INDUCTION ALGORITHM C4.5

Ēriks Tipāns, Rīgas Tehniskā universitāte, ASTF, Informācijas tehnoloģijas institūts
Lēmumu atbalsta sistēmu profesora grupa, e–pasts: eriks@ibm.cs.ru.lv

Abstract. *There are considered possibilities to create the new concepts generalization algorithm in this paper, which would combine methods used in decision trees induction algorithm C4.5 and concepts generalization by features algorithm CORA. The newly created algorithm will be named CORA 4.5.*

1. Ievads – algoritms CORA

Bibliogrāfijā, kas veltīta mākslīgā intelekta problēmām, ir plaši apskatīti dažādi populāri jēdzienu vispārināšanas algoritmi (Гладун, 1987), tomēr jāatzīst, ka pietiekoši plašu popularitāti nav guvis M. Bongarda 60. gados izstrādātais algoritms CORA (Bongard, 1970), kas veic jēdzienu vispārināšanu, balstoties uz klasificējamo objektu pazīmju kompleksu

ISBN 9984 - 585 - 36 - 0