Soft Comput (2005) 9: 448–462
DOI 10.1007/s00500-004-0363-x

ORIGINAL PAPER

J. Liu · J. Lampinen

A Fuzzy Adaptive Differential Evolution Algorithm

Published online: 2 June 2004

Abstract The differential evolution algorithm is a floating-point encoded evolutionary algorithm for global optimization over continuous spaces. The algorithm has so far used empirically chosen values for its search parameters that are kept fixed through an optimization process. The objective of this paper is to introduce a new version of the Differential Evolution algorithm with adaptive control parameters – the fuzzy adaptive differential evolution algorithm, which uses fuzzy logic controllers to adapt the search parameters for the mutation operation and crossover operation. The control inputs incorporate the relative objective function values and individuals of the successive generations. The emphasis of this paper is analysis of the dynamics and behavior of the algorithm. Experimental results, provided by the proposed algorithm for a set of standard test functions, outperformed those of the standard differential evolution algorithm for optimization problems with higher dimensionality.

Keywords Differential evolution · Evolutionary algorithms · Fuzzy logic control

1 Introduction

Setting the search parameter values in evolutionary algorithms (EA) can be divided into two classes: *parameter tuning* and *parameter control*. Parameter tuning is the commonly practiced approach that amounts to finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run. Parameter control is an alternative, as it amounts to starting a run with initial parameter values, which are changed during the run. Methods for changing the value of a parameter can be classified into one of three categories [3]:

– *Deterministic parameter control*: This takes place when the value of a strategy parameter is altered by some deterministic rule. This rule modifies the strategy parameter deterministically without using any feedback from the search. Usually, a time-varying schedule is used, i.e., the rule will be used when a set number of generations have elapsed since the last time the rule was activated.
– *Adaptive Parameter Control*: This takes place when there is some form of feedback from the search and this feedback is used to determine the direction and/or magnitude of the changes to the strategy parameter. Setting the value of the strategy parameter may involve credit assignment, and the action of the EA may determine whether or not the new value persists or propagates throughout the population.
– *Self-Adaptive Parameter Control*: The idea of an evolutionary search can be used to implement the self-adaptation of two search parameters. Here the parameters to be adapted are coded into the chromosomes that undergo mutation and recombination. Better values for these encoded parameters are supposed to result in better individuals that in turn are more likely to survive and produce offspring and hence propagate better parameter values.

In this paper, an approach for "adaptive parameter control" is under investigation.

Previously, fuzzy concepts and genetic algorithms (GAs) have been integrated into the so-called fuzzy genetic algorithms (FGAs), which can be presented as being based on two approaches [4]:

– Modeling different GA components by using fuzzy logic based techniques.
– Managing problems in an imprecise environment, where the imprecision is modeled by fuzzy sets, GAs then represent and run these problems.

J. Liu (✉) · J. Lampinen
Laboratory of Information Processing,
Lappeenranta University of Technology,
P.O. Box 20, FIN-53851 Lappeenranta, Finland
E-mail: junhong.liu@lut.fi
Tel.: +358-5-6212813
Fax: +358–5–6212899

Several fuzzy logic based techniques have been used previously for improving GA behavior, i.e., conducting the choice of search parameter settings. The knowledge of GA experts was used for dynamically computing appropriate settings, such as population size, crossover operator and mutation operator throughout the GA's execution to avoid the problem of premature convergence [6].

The differential evolution (DE) algorithm, an evolutionary algorithm recently introduced by Price and Storn [16], is designed for global optimization problems over continuous domains. It has three control search parameters:

(1) The mutation control parameter, *F*, is a real and constant factor that controls the amplification of the differential variation.
(2) The crossover control parameter, *CR*, is a real and constant factor that controls which parameter contributes to which trial vector parameter in the crossover operation.
(3) The population size, *NP*, is the number of population members.

There are many papers about DE and its applications [1, 5, 7–10, 14, 16–21, 23], where recommended value ranges for the mutation control parameter, *F*, the crossover control parameter, *CR*, and the population size, *NP*, are given [17]. The values of these parameters determine whether the algorithm will effectively find a near-optimum solution and whether it will find such a solution efficiently. Choosing the right parameter values by a trial-and error approach is often a time-consuming task. Several papers have discussed the influence of parameters on the performance of DE; they will be briefly overviewed in the following paragraphs.

The probability of stagnation depends on how many different potential trial solutions are available and also on their ability to enter into the population of the following generation, i.e., it depends on population size, mutation control parameter, crossover control parameter, current population and objective function, as presented in [5].

By introducing two new parameters (tolerances) and taking into account the diversity in the population, values of the mutation control parameter and the crossover control parameter can be modified to improve the efficiency of the algorithm and the quality of the solution [10].

A self-adaptive approach to DE parameters, the mutation amplification and the crossover parameter, was presented in [1].

The influence of three control parameters on the performance of DE was demonstrated individually by conducting experiments on test functions in [7], which revealed that ways of improving the effectiveness, efficiency and robustness of DE could be found by finding a better approach for setting the DE's search control parameter values. It concluded that: (1) The compound effect of the search parameters of DE should be considered in order to make the DE algorithm perform well for the problem at hand; (2) The best settings were rather heavily problem dependent, so that there were difficulties in setting the values for parameters and in understanding their influence, both individually and in combination, on the DE performance.

In [8], an algorithm based on the fuzzy logic control (FLC) was presented in which the mutation amplification, *F,* was controlled using a single FLC. Its two inputs were: linearly depressed parameter vector change and function value change over the whole population members between the current generation and the last generation. The use of FLC to control the DE's parameter was considered for solving two problems to which a standard DE could be subjected: a low convergence velocity and failure risk (stagnation and premature convergence). These problems were due to: (1) Control parameters not being well chosen initially for a given task; (2) Parameters being kept fixed through the whole process and having no response to the population's information (function parameter vectors, function values, and their changes) even though the environment in which the DE operated may be variable; (3) Lack of knowledge from the search space. The method combined the features of DE and fuzzy logic. It possessed robustness and fuzziness. FLC was proposed for controlling parameters of DE in order to be able to choose the initial control parameters freely and adjust the control parameters on-line to dynamically adapt to changing situations. The Ackley's path function was used for demonstration. It was found that DE with a fuzzy search parameter control could perform better than DE using all fixed parameters.

In [9], two FLCs were used to control the mutation control parameter and the crossover parameter. These two parameters were adapted individually for each generation. Parameter vector change and function value change over the whole population members between the last two generations were nonlinearly depressed and then used as the inputs for both FLCs. A set of benchmark functions were tested to explore the performance of this fuzzy adaptive differential evolution algorithm (FADE) which showed that FADE resulted in a fast convergence with a higher problem dimension.

The relationship between the control parameters of DE and the evolution of population variance has been analyzed both from a theoretical and an empirical viewpoint. This could be used to obtain control parameters' values in order to prevent premature convergence as in [20].

Introducing a population refreshment mechanism into DE, which counted the number of 'copies' among solution vectors in the population according to a predefined tolerance, prevented stagnation and improved the convergence speed [23].

Parameters of DE were adapted by controlling the population diversity in [21].

There still exists however a lack of knowledge on how to find reasonably the best values for the control parameters of DE for a given function. Since the interaction of control parameters with the DE's performance is complex in practice, a DE user should select the initial parameter settings for a problem at hand by an educated guess. Then, the trial-and-error method has to be used for fine-tuning the control parameters further. In practice, the optimization run has to be performed multiple times with different settings. In some cases, the time for finding these parameters is unacceptably long.

In the original DE, the control variables are kept fixed during the optimization. In this paper, a new version of DE, where the mutation control parameter and the crossover control parameter are adaptive, is introduced. It is called the fuzzy adaptive differential evolution algorithm. It uses a fuzzy knowledge-based system [11] to dynamically control DE parameters, such as F and CR.

This paper aims to introduce FADE, giving an initial analysis of its behaviour and some comparisons against several adaptive methods for a set of test functions. The paper is organized as follows: the DE algorithm is presented briefly in Sect. 2, where the possibility of implementing fuzzy adapting for determining parameters of DE is given. In Sect. 3 the proposed algorithm follows: the fuzzy adapting idea is first included in the determination of the parameters for DE; the inputs and outputs of the fuzzy controllers are defined based on the information obtained during the last two generations; each input and output have an associated set of linguistic labels, whose meaning is specified through membership functions of fuzzy sets taking into consideration the uncertainty and non-linearity that appear in the stochastic optimisation of a function; the bounded ranges of values for every input and outputs are determined in order to define these membership functions over them; then the fuzzy rules describing the relationship between the inputs and output are defined by using the experience and knowledge of the DE experts. Experiments are then presented in Sect. 4 and conclusions are drawn in Sect. 5. The test functions applied for experimentation are listed in the Appendix in Sect. 6.

2 The Differential Evolution Algorithm

2.1 The $(G_{\max}, NP, \lambda, \rho)$ Differential Evolution

The DE algorithm has been demonstrated to be an efficient, effective and robust optimization method in guiding evolutionary principles of "survival of the fittest", especially in the case of problems containing continuous problem variables as in [16].

DE is defined for continuous variables by the following 10-tuple adopted from [15]:

$$(G_{\max}, NP, \lambda, \rho)DE := (P(0), G_{\max}, \lambda, \rho, \omega, NP, F, CR, t, e) \ , \tag{1}$$

with:

$P_{(0)} := (\mathbf{x}_{1,0}, \mathbf{x}_{2,0}, \ldots, \mathbf{x}_{NP,0}) \in \boldsymbol{I}$ Initial population.
$NP \in \aleph$ Population size.
$\lambda = NP \in \aleph$ Number of offspring.
$\boldsymbol{I} : = \Re^D$ Space of individuals.
$G_{\max} \in \aleph, G_{\max} \geq 1$ Maximum generation.
$\rho \in \{1, 2, 3, 4, 5\}$ Number of ancestors for a descendant.
$\omega \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ Type of DE strategy.
mut: $\boldsymbol{I} \to \boldsymbol{I}$ Mutation operator.
$F \in [0, 2]$ Mutation control parameter.
cro: $\boldsymbol{I}^{NP+\lambda} \to \boldsymbol{I}^{NP}$ Crossover operator.
$CR \in [0, 1]$ Crossover control parameter.
sel: $\boldsymbol{I}^{NP+\lambda} \to \boldsymbol{I}^{NP}$ Selection operator.
$t : \boldsymbol{I} \to \Re^D_+$ Termination criterion.
$e \in \Re_+$ Accuracies required.

The $(D, f, m, \boldsymbol{G})$ optimization problem for continuous variables at hand may be defined as follows:

$$\text{Minimize } \{f(\mathbf{x}) | \mathbf{x} \in M \subseteq \Re^D\} \ , \tag{2}$$

with:

$D \in \aleph$ Dimension of the problem.
$f : M \to \Re$ Objective function.
$\mathrm{M} = \{\mathbf{x} \in \Re^D | g_j(\mathbf{x}) \geq 0 \ \forall j \in \{1, \ldots, m\}\}$ Feasible region.
$m \in \aleph_0$ Number of constraints.
$\boldsymbol{G} = \{g_j : \Re^D \to \Re \ \forall \ j \in \{1, \ldots, m\}\}$ Set of constraints.

$P_{(0)}$ denotes the initial population (iteration counter $G = 0$) of parents and consists of arbitrary vectors $\mathbf{x} \in \boldsymbol{I}$. A vector represents each element of the population at reproduction cycle G:

$$\mathbf{x}_{i,G} \in P_{(G),} \ i \in \aleph \ . \tag{3}$$

One iteration, that is a step from a population $P_{(G)}$ towards the next reproduction cycle with $P_{(G+1)}$ is modeled as follows:

$$P_{(G+1)} := opt_{DE}\left(P_{(G)}\right) \ , \tag{4}$$

where $opt_{DE} : \boldsymbol{I}^{NP} \to \boldsymbol{I}^{NP}$ is defined by

$$opt_{DE} := \mathbf{sel} \circ (\mathbf{cro} \circ \mathbf{mut}) \ , \tag{5}$$

operating on an input population $P_{(G)}$ according to

$$opt_{DE}(P_{(G)}) := \mathbf{sel}(P_{(G)} \cup (\mathbf{cro}(\mathbf{mut}(P_{(G)})))) \ , \tag{6}$$

where $\cup$ denotes the union operation on multiples. Equation (6) clarifies that the population at generation $G + 1$ is obtained from $P_{(G)}$ by first applying mutation and crossover, which results in an intermediate population $P'_{(G)}$ of size $NP,$ and then applying the selection operator to the union of $P_{(G)}$ and $P'_{(G)}$. Recall that the mutation operator generates only one trial individual per application, which can then be subject to crossover with an existing population member.

2.1.1 The mutation operator mut(ρ,F,ω)

The mutation operator $\mathbf{mut} : \boldsymbol{I} \to \boldsymbol{I}$ is defined as follows:

$$\mathbf{mut} := \mathbf{mu} \circ \mathbf{co} \ , \tag{7}$$

where

$\mathbf{co} : \boldsymbol{I} \rightarrow \boldsymbol{I}^{\rho}$ choose $1 \leq \rho \leq 5$ different parents vectors from $\boldsymbol{I}$ with uniform probability.

$\mathbf{mu} : \boldsymbol{I}^{\rho} \rightarrow \boldsymbol{I}$ creates one offspring vector by generating one mutated vector from $(\rho - 1)$ parents and then adding to the perturbed parent.

There are several ways to conduct the mutation in order to get an offspring [17] as shown in Table 1.

2.1.2 *The crossover operator* $cro(CR, \omega)$

The crossover operator $\mathbf{cro} : \boldsymbol{I}^{NP+\lambda} \rightarrow \boldsymbol{I}^{NP}$ is defined as follows:

$$\mathbf{cro} := \mathbf{cr_x} \ , \tag{8}$$

with $\mathbf{cr}_x$ denoting the component-wise crossover operators acting on object variables.

$\mathbf{cr}_x \ (CR, z_1, z_2, \ldots, z_n) := \boldsymbol{I}^{NP+\lambda} \rightarrow \boldsymbol{I}^{NP}$ cross the noisy object variables obtained from the mutation operation and target object variables based on the comparison result between CR and z_i, where z_i s are random variables, $i \in \{1, 2, \ldots, n\}, n \in \aleph$. The offspring variables come from the noisy object variable with a probability of CR for every object.

Methods of conducting the crossover operation are the exponential crossover [17] and the binomial crossover [18].

2.1.3 *The selection operator sel*

The traditional deterministic DE selection operator can be defined as:

$$\mathbf{sel} := \boldsymbol{I}^{NP+\lambda} \rightarrow \boldsymbol{I}^{NP} \ . \tag{9}$$

Let $P_{(G)}$ denote the parent population in the reproduction cycle $G, P'_{(G)}$ their offspring produced by mutation and crossover, and $Q_{(G)} = P_{(G)} \cup P'_{(G)} \in \boldsymbol{I}^{NP+\lambda}$, where the operator $\cup$ denotes the union operation on multiples. Then

$$P_{(G+1)} := \mathbf{sel}(Q_{(G)}) \ . \tag{10}$$

The next reproduction cycle contains the NP best individual according to the following relation:

$$\mathbf{X}_{G+1} \in P_{(G+1)} :$$

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{x}_{i,G}^{(P'_{(G)})} & \text{if } f(\mathbf{x}_{i,G}^{(P'_{(G)})}) \leq f(x_{i,G}^{(P_{(G)})}), \\ \mathbf{x}_{i,G}^{(P_{(G)})} & \text{otherwise}, \end{cases}$$
$$i = \{1, 2, ..., NP\} \ . \tag{11}$$

Table 1 DE schemes

Scheme	Explanation
DE/rand/1	The perturbed parent is a random member and added with a difference vector.
DE/best/1	The perturbed parent is the current best member and added with a difference vector.
DE/rand-to-best/1	The perturbed parent is the target member and added with a difference vector.
DE/best/2	The perturbed parent is the current best member and added with two difference vectors.
DE/rand/2	The perturbed parent is a random member and added with two difference vectors.

Note: the mutation operation can be a binomial crossover or an exponential crossover

2.2 The influence of control variables on the performance of DE

Following previous reports [5], [14], [16], [17], [18], [19], F, CR, and NP are kept fixed during the optimization process, and can be found by trial-and-error, usually after performing a few experiments with different values. The suggested choices are in [17]:

- $F \in [0.5,1]$;
- $CR \in [0.8,1]$;
- $NP = 10{\cdot}D$ where D is the dimensionality of the problem.

The experiments on the search parameters' influence on the performance of DE were carried out with a set of standard test functions in [7]. The results from the experiments showed that with different functions (problems) $NP{=}10{\cdot}D$ was really helpful advice, but for F and CR, the situation was complicated. Figures 1 and 2 show the results obtained using DE with different values of F and CR. The stopping criteria adopted for DE is to terminate the search process when one of the following conditions is satisfied: (1) the maximum number of generations ($G_{\max}$) is reached; (2) $|f(\mathbf{x}) - f(\mathbf{x}^*)| < e$ where $f(\mathbf{x})$ is the current best objective function value and $f(\mathbf{x}^*)$ is the optimum of the objective function. Figure 1 shows the curves of failures and function evaluations versus F for the 7th function of the Testbed as an example. The results were obtained with parameters set as $F \in [0, 2]$, $CR = 0.9$, $D = 5$, $NP = 50$, $f(\mathbf{x}^*) = 0$, $e = 10^{-5}, G_{\max} = 500$. Figure 2 shows the curves of failures and function evaluations versus CR for the 7th function of the Testbed as an example. The results were obtained with parameters set as $F = 0.9$, $CR \in [0, 1]$, $D = 2$ or 5, $NP = 10{\cdot}D, f(\mathbf{x}^*) = 0$, $e = 10^{-4}$, $G_{\max} = 800$. In Figs.1 and 2, NFE and NRF represent respectively the number of function evaluations and the number of run failures that do not converge to the global optimum in all 100 executions (with different seed values).

It can be seen that in Fig. 1 there is a range of F without failures and a valley in the curve of function evaluations versus F when F changes from 0 to 2. For different functions, the two curves can be different, but experience the same trend. The lowest point on the curve in Fig. 1b may not fix at one point in the range of [0.5, 1], it can even exist outside this range.

Figure 2 shows how failures and function evaluations change with CR. When $D>2$, the curve of function evaluations versus CR experience a similar trend to those versus F. When $D = 2$, the higher the value of CR in the range, the better the convergence speed of DE.

At this point, a new method using fuzzy logic to adapt control parameters of DE is introduced, which combines the features of DE and fuzzy logic, and possesses robustness and fuzziness.

3 The fuzzy adaptive differential evolution

The FADE algorithm for optimization problems is an adaptation of the DE algorithm described above. FADE works as follows. Assuming that all variables are bounded between $[x_j^{(L)}, x_j^{(U)}]$, an initial population is generated at random using uniformly distributed random numbers chosen from a uniform distribution on the interval [0.0, 1.0]. DE generates a new noisy vector by adding the fixed weighted difference between two population vectors to a third vector. A trial vector is generated in a way that the source of each trial vector parameter can come from the target vector parameter or from the trial vector parameter based on the comparison result between the crossover control parameter and a uniformly distributed random number on the interval [0, 1] that is generated new for each parameter. If the resulting trial vector yields a lower or equal objective function value than a predetermined population member (the target vector), the generated trial vector will replace the target vector that it is compared with; otherwise, the old target vector is retained. This process continues until the population is completed. In contrast to DE, FADE adapts the mutation control parameter and crossover control parameter. The generic version of this algorithm is presented in Fig. 3.

The basic idea of control parameter adaptation can be expressed using the control diagram in Fig. 4 adapted from [11]. Two FLC systems are used as the basis of a fuzzy control mechanism. The following steps are used.

3.1 System parameters and fuzzy sets membership functions

The mean square roots of differences concerning two successive generations over the whole population during the optimization process for function values (FC) and population members (PC) are depressed and used as the inputs of FLCs, and the control parameters (F and CR) of DE are the outputs. The inputs are defined as:

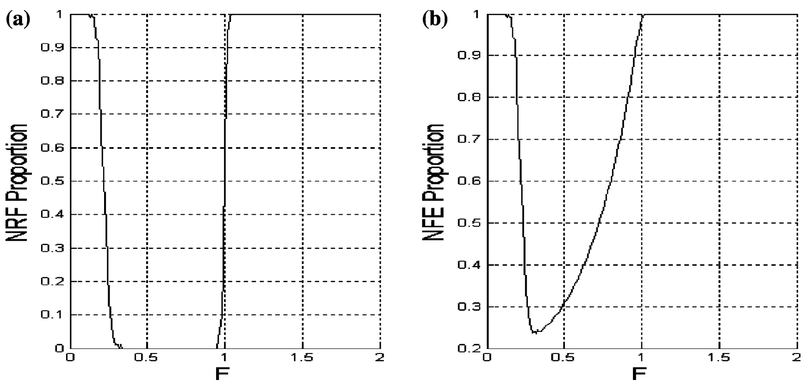


Fig. 1a, b Proportions. **a** proportion of failed optimization runs of DE among all runs performed, **b** proportion of function evaluations to the maximum number of function evaluations. $F \in [0, 2]$

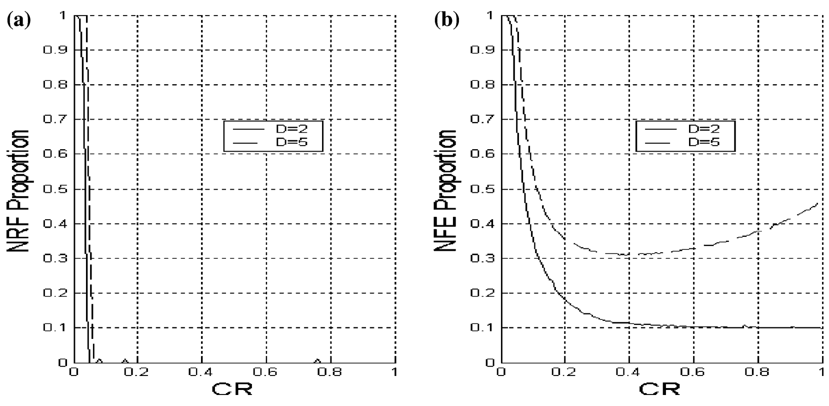
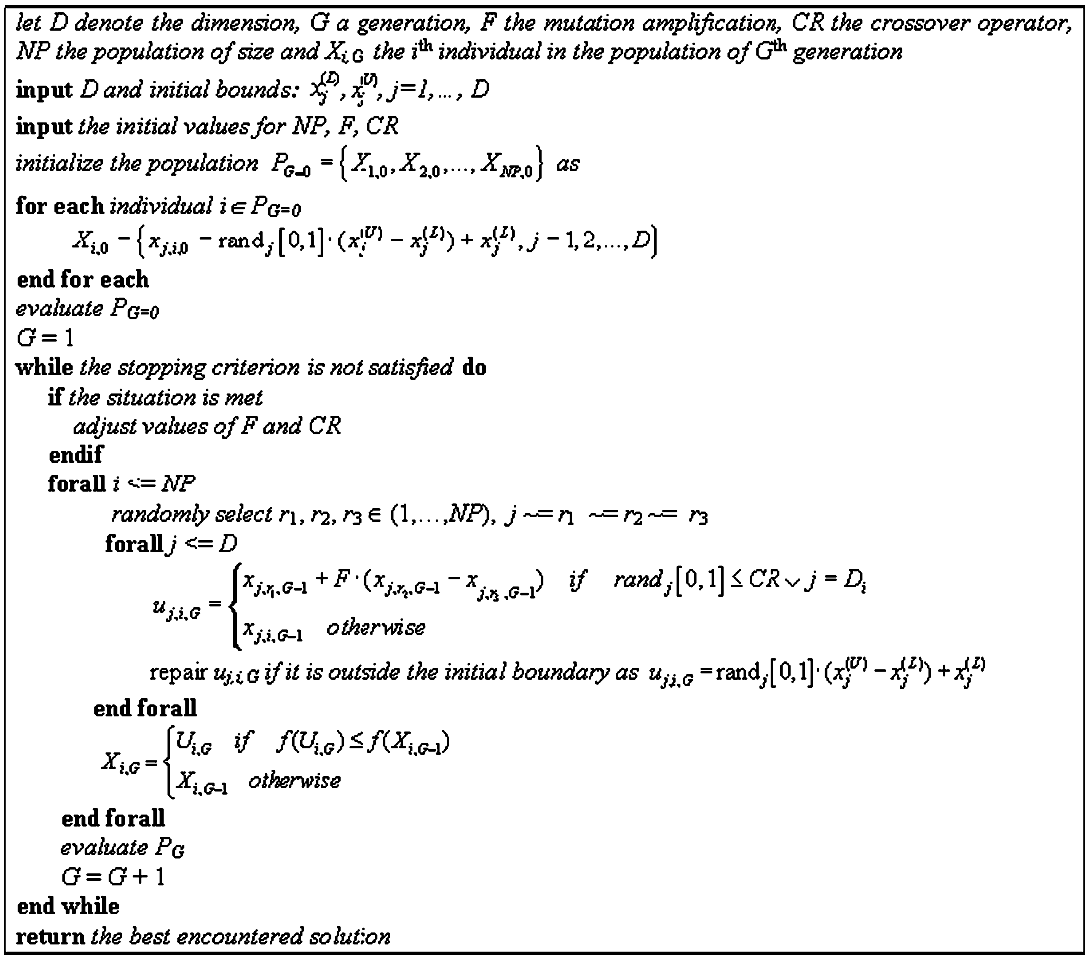


Fig. 2a, b Proportions. **a** proportion of failed optimization runs of DE to all runs performed, **b** proportion of function evaluations to the maximum number of function evaluations. $CR \in [0, 1]$



```
let $D$ denote the dimension, $G$ a generation, $F$ the mutation amplification, $CR$ the crossover operator,
$NP$ the population of size and $X_{i,G}$ the i^th individual in the population of $G$^th generation
input $D$ and initial bounds: $x_j^{(L)}, x_j^{(U)}, j=1,\ldots,D$
input the initial values for $NP$, $F$, $CR$
initialize the population $P_{G=0} = \{X_{1,0}, X_{2,0}, \ldots, X_{NP,0}\}$ as
for each individual $i \in P_{G=0}$
    $X_{i,0} = \{x_{j,i,0} = \mathrm{rand}_j[0,1]\cdot(x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}, j = 1,2,\ldots,D\}$
end for each
evaluate $P_{G=0}$
$G = 1$
while the stopping criterion is not satisfied do
  if the situation is met
    adjust values of $F$ and $CR$
  endif
  forall $i <= NP$
      randomly select $r_1, r_2, r_3 \in (1,\ldots,NP)$, $j \sim= r_1 \sim= r_2 \sim= r_3$
    forall $j <= D$
        $u_{j,i,G} = \begin{cases} x_{j,r_1,G-1} + F\cdot(x_{j,r_2,G-1} - x_{j,r_3,G-1}) & \text{if } rand_j[0,1] \le CR \vee j = D_i \\ x_{j,i,G-1} & \text{otherwise} \end{cases}$
      repair $u_{j,i,G}$ if it is outside the initial boundary as $u_{j,i,G} = \mathrm{rand}_j[0,1]\cdot(x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$
    end forall
    $X_{i,G} = \begin{cases} U_{i,G} & \text{if } f(U_{i,G}) \le f(X_{i,G-1}) \\ X_{i,G-1} & \text{otherwise} \end{cases}$
  end forall
  evaluate $P_G$
  $G = G + 1$
end while
return the best encountered solution
```

Fig. 3 The adaptive differential evolution algorithm

$$\left.\begin{aligned} PC &= \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\sum_{j=1}^{D}\left(x_{j,i}^{(n)} - x_{j,i}^{(n-1)}\right)^2} \\ FC &= \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\left(f_i^{(n)} - f_i^{(n-1)}\right)^2} \end{aligned}\right\}, \tag{12}$$

and

$$\left.\begin{aligned} d_{11} &= 1 - (1 + PC)\cdot e^{-PC} \\ d_{12} &= 1 - (1 + FC)\cdot e^{-FC} \\ d_{21} &= 2\cdot\left(1 - (1 + PC)\cdot e^{-PC}\right) \\ d_{22} &= 2\cdot\left(1 - (1 + FC)\cdot e^{-FC}\right) \end{aligned}\right\}, \tag{13}$$

where $f_i^{(n)}$ represents the ith component of the function value vector $f \in \Re^{NP}$ for the nth generation, $i = 1, 2, \ldots, NP$; $x_{j,i}^{(n)}$ represents the component in the jth row and ith column of the parameter matrix $\mathbf{X}_{D\times NP}$ for the nth generation, $i = 1, 2, \ldots, NP, j = 1, 2, \ldots, D$; n represents the generation index; PC is the parameter vector change in magnitude during the last two successive generations and depressed into the range of [0, 1] as d_{11} and the range of [0, 2] as d_{21}; FC is the function value change during the last two successive generations and depressed into [0, 1] as d_{12} and [0, 2] as d_{22}.

d_{i1} and d_{i2} are treated as fuzzy variables of the ith FLC system with 6 elements, where $i = 1, 2$; 3 elements are for F and CR, individually. The values are then assigned as membership grades in 3 fuzzy subsets as follows: *small*, *medium* and *big*.

More precisely, we denote them by fuzzy sets $a_1^{(i)}$, $a_2^{(i)}$, $a_3^{(i)}$, $c_1^{(i)}$, $c_2^{(i)}$, $c_3^{(i)}$, $ou_1^{(i)}$, $ou_2^{(i)}$, $ou_3^{(i)}$, where $i = \{1, 2\}$ for the 1st FLC and the 2nd FLC, i.e., $a_1^{(i)}$, $a_2^{(i)}$, and $a_3^{(i)}$are the fuzzy subsets *small*, *medium*, and *big* (short as S, M, and B) of the fuzzy variable d_{i1} of the ith FLC, $c_1^{(i)}$, $c_2^{(i)}$, $c_3^{(i)}$are the fuzzy subsets S, M, and B of the fuzzy variable d_{i2} of the i th FLC, $ou_1^{(1)}$, $ou_2^{(1)}$, $ou_3^{(1)}$ are the fuzzy subsets S, M, and B of the fuzzy variable F of the 1st FLC, $ou_1^{(2)}$, $ou_2^{(2)}$, $ou_3^{(2)}$ are the fuzzy subsets S, M, and B of the fuzzy variable CR of the 2nd FLC. They are defined in the universe of discourse $A^{(1)}$, $C^{(1)}$, $OU^{(1)}$, $A^{(2)}$, $C^{(2)}$ and $OU^{(2)}$ respectively [13], i.e., $A^{(i)} = \{a_1^{(i)}, a_2^{(i)}, a_3^{(i)}\}$, $C^{(i)} = \{c_1^{(i)}, c_2^{(i)}, c_3^{(i)}\}$, and $OU^{(i)} = \{ou_1^{(i)}, ou_2^{(i)}, ou_3^{(i)}\}$, $i = \{1, 2\}$ are 6 spaces.

There are many alternative membership functions in MATLAB, such as the Gaussian curve membership function, the Generalized bell curve membership function, the Triangular membership function, the Trape-

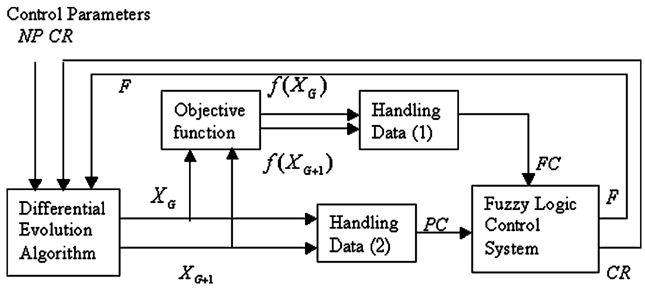


Fig. 4 Control diagram of DE with fuzzy adapting: *PC* – the parameter vector change in magnitude and *FC* – the objective function value change

zoidal membership function, etc. Since the choice of the type and the precise values for the parameters have, in general, only little influence on the results, decisions, and conclusions obtained, as long as a local monotonicity is preserved [2], the Gaussian curve membership function, f_g, is chosen for every input and output in Fig. 5.

Based on experimental results in [7], d_{1j} and CR range in [0, 1], while d_{2j} and F range in [0, 2]. The centre of a membership function of a fuzzy set is $c_{z,zr}$, where z is the name of the fuzzy set, i.e., $z = \{d_{11}, d_{12}, F, d_{21}, d_{22}, CR\}$, and zr is the name of a fuzzy subset, whose linguistic value is *small*, *medium* or *big* (the so-called reference fuzzy sets). The centres of membership functions of $a_1^{(i)}$, $c_1^{(i)}$ and $ou_1^{(i)}$, $c_{z,S}$, i.e., $c_{d_{11},S}$, $c_{d_{12},S}$, $c_{F,S}$, $c_{d_{21},S}$, $c_{d_{22},S}$, and $c_{CR,S}$, are close to the lower limit of that input; the centres of membership functions of $a_3^{(i)}$, $c_3^{(i)}$ and $ou_3^{(i)}$, $c_{z,B}$, i.e., $c_{d_{11},B}$, $c_{d_{12},B}$, $c_{F,B}$, $c_{d_{21},B}$, $c_{d_{22},B}$, and $c_{CR,B}$, are close to the upper limit of the input; the centres of membership functions of $a_2^{(i)}$, $c_2^{(i)}$ and $ou_2^{(i)}$, $c_{z,M}$, i.e., $c_{d_{11},M}$, $c_{d_{12},M}$, $c_{F,M}$, $c_{d_{21},M}$, $c_{d_{22},M}$, and $c_{CR,M}$, are about the average of $c_{z,S}$ and $c_{z,B}$, where $i = \{1, 2\}$. Based on this principle and the obtained curves of NRC and NFE proportion versus F and CR in [7], the parameters that Gaussian curve membership functions depend on are chosen as following: (1) The centres of the membership functions of fuzzy subsets, i.e., *small*, *medium*, and *big*, are: 0.05, 0.5 and 0.9 individually for d_{11}, 0.01, 0.5 and 0.9 for d_{12}, 0.1, 0.8 and 1.5 for d_{21}, 0.1, 0.8 and 1.5 for d_{22}, 0.3, 0.6 and 0.9 for F, 0.4, 0.7 and 1.0 for CR. (2) The shape of each membership function ($\sigma_{z,zr}$) is determined in the way that the bigger the slope of the curve, the bigger the value of $\sigma_{z,zr}$. The shapes of the membership functions for one fuzzy set are the same, i.e., $\sigma_{d_{11},zr} = 0.25$, $\sigma_{d_{12},zr} = 0.35$, $\sigma_{F,zr} = 0.5$, $\sigma_{d_{21},zr} = 0.5$, and $\sigma_{d_{22},zr} = 0.5$, $\sigma_{CR,zr} = 0.35$, where $zr = \{S, M, B\}$. The definitions of the membership functions are given in the form of

$$\mu_{zr}(z) = f_g(z, c_{z,zr}, \sigma_{z,zr}) = e^{-\frac{(z-c_{z,zr})^2}{2\cdot\sigma_{z,zr}^2}} , \tag{14}$$

as shown in Table 2.

3.2 Fuzzy Rules

Once the fuzzy subsets like *small, medium* and so on are defined, the so-called IF-THEN rule statements are used to formulate the conditional statements that comprise fuzzy logic. There are "9×2" rules as shown in Table 3. Links existing between the linguistic values are given in the form of conditional statements (rules) [13]:

$$\text{if}(A_n \text{ and } C_n), \text{then } OU_n , \tag{15}$$

where $n = 1, 2, \ldots, N, N = 9$ denotes the number of the rules for each FLC. A_n, C_n and OU_n are fuzzy sets taken from the collection of the reference fuzzy sets introduced in Section 3.1. Writing down all the control rules explicitly indicating these reference fuzzy sets, we have:

$$if(A_{n_1(n)}^{(i)} \text{ and } C_{n_2(n)}^{(i)}), \text{ then } OU_{n_3(n)}^{(i)} , \tag{16}$$

where

$n_1 : \{1, 2, \ldots, N\} \rightarrow \{1, 2, 3\}$,
$n_2 : \{1, 2, \ldots, N\} \rightarrow \{1, 2, 3\}$,

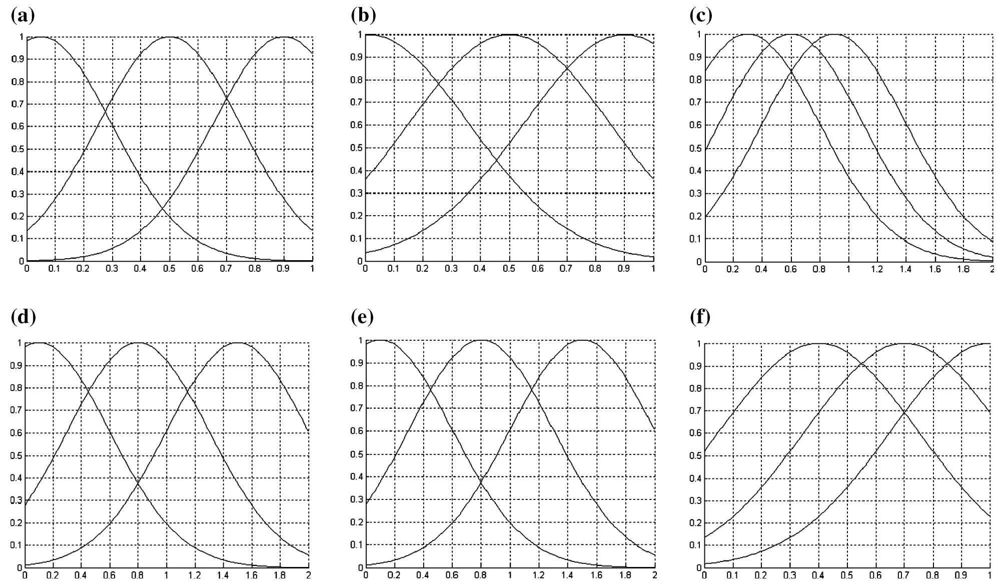


Fig. 5a–f Membership functions for inputs and outputs. **a** d_{11}, **b** d_{12}, **c** F, **d** d_{21}, **e** d_{22}, **f** CR

Table 2 Membership functions

Inputs, outputs	Membership functions	
	The first FLC	The second FLC
d_{i1}	$\mu_S(d_{11}) = f_g(d_{11}, 0.25, 0.05)$ $\mu_M(d_{11}) = f_g(d_{11}, 0.25, 0.5)$ $\mu_B(d_{11}) = f_g(d_{11}, 0.25, 0.9)$	$\mu_S(d_{21}) = f_g(d_{21}, 0.5, 0.1)$ $\mu_M(d_{21}) = f_g(d_{21}, 0.5, 0.8)$ $\mu_B(d_{21}) = f_g(d_{21}, 0.5, 1.5)$
d_{i2}	$\mu_S(d_{12}) = f_g(d_{12}, 0.35, 0.01)$ $\mu_M(d_{12}) = f_g(d_{12}, 0.35, 0.5)$ $\mu_B(d_{12}) = f_g(d_{12}, 0.35, 0.9)$	$\mu_S(d_{22}) = f_g(d_{22}, 0.5, 0.1)$ $\mu_M(d_{22}) = f_g(d_{22}, 0.5, 0.8)$ $\mu_B(d_{22}) = f_g(d_{22}, 0.5, 1.5)$
F	$\mu_S(F) = f_g(F, 0.5, 0.3)$ $\mu_M(F) = f_g(F, 0.5, 0.6)$ $\mu_B(F) = f_g(F, 0.5, 0.9)$	
CR		$\mu_S(CR) = f_g(CR, 0.35, 0.4)$ $\mu_M(CR) = f_g(CR, 0.35, 0.7)$ $\mu_B(CR) = f_g(CR, 0.35, 1.0)$

Note: S = *small*; M = *medium*; B = *big*. d_{ij} = the j^{th} input of the i^{th} FLC system

$n_3 : \{1, 2, \ldots, N\} \to \{1, 2, 3\}$,
$i := 1, 2.$

In Table 3, each one of these rules has two input variables and one output variable, as a rule is: "IF d_{i1} is A and d_{i2} is C then u is OU", where d_{i1}, d_{i2} and u are linguistic variables while A, C and OU are fuzzy subsets defined by membership functions. It describes the partial behavior or characteristic of a system and represents the mapping or fuzzy relation $R_{((d_{i1},d_{i2}),u)}$ from the input space $D_{i1} \times D_{i2}$ to the output space U. Numerically, such a fuzzy relation may be expressed as

$$R_{((d_{i1},d_{i2}),u)} = \int_{(D_{i1}D_{i2})} \int_{U} \mu_R((d_{i1},d_{i2}),u)$$
$$= \int_{(D_{i1}D_{i2})} \int_{U} \min\left((\mu_A(d_{i1}), \mu_C(d_{i2})), \mu_{OU}(u)\right) \quad , \tag{17}$$

where $\mu_A(d_{i1})$ is called the membership function or the grade of membership of d_{i1} in A that maps D_{i1} to the membership space, $\mu_C(d_{i2})$ is called the membership function or grade of membership of d_{i2} in C that maps D_{i2} to the membership space, $\mu_{OU}(u)$ is called the membership function or the grade of membership of u in OU that maps U to the membership space, $\mu_R((d_{i1}, d_{i2}),u)$ is called the membership function of the fuzzy relation that is a fuzzy set on $(D_{i2} \times D_{i2}) \times U$ [22].

Table 3 The rules of function

Rules	Fuzzy Sets			
	d_{i1}	d_{i2}	F	CR
1	*S*	*S*	*S*	*S*
2	*S*	*M*	*M*	*M*
3	*S*	*B*	*B*	*B*
4	*M*	*S*	*M*	*M*
5	*M*	*M*	*M*	*M*
6	*M*	*B*	*B*	*B*
7	*B*	*S*	*B*	*B*
8	*B*	*M*	*B*	*B*
9	*B*	*B*	*B*	*B*

Note: S = *small*; M = *medium*; B = *big*.

In the form of a fuzzy relation that combines all the rules [13], we have:

$$R = R_1 \cup R_2 \cup \cdots \cup R_9 = \bigcup_{n=1}^{9} \left(A_n^{(i)} \times C_n^{(i)} \times OU_n^{(i)}\right),$$
$$R(a_l^{(i)}, c_j^{(i)}, ou_k^{(i)}) = \max_{1 \le n \le 9} \left[A_n^{(i)}(a_l) \wedge C_n^{(i)}(c_j) \wedge OU_n^{(i)}(ou_k)\right] \quad , \tag{18}$$

where $i = \{1, 2\}$, $l = \{1, 2, 3\}$, $j = \{1, 2, 3\}$, and $k = \{1, 2, 3\}$. Note that the union of R_ns reflects the modelling of 'or else', the conjunction standing in the control protocol. The cartesian product is considered to be represented by the minimum operator. The fuzzy relation summarises associations between fuzzy sets A_i, C_i and OU_i. Written down in a more explicit manner, it results in a superposition of the minimization with a maximization as [2]:

$$\mu_R(d_{i1}, d_{i2}, u) = \max_{1 \le n \le 9} \min\left(\mu_{A_n}(d_{i1}), \mu_{C_n}(d_{i2}), \mu_{ou_n}(u)\right) \quad . \tag{19}$$

F and CR should be adapted based on PC and FC (i.e., d_{ij}). When PC, the parameter vector change in magnitude, is larger, it indicates that population members of later generation are spreading in a region of the optimal solution, which is moving farther away aggregately from that of the former generation. So the mutation control parameter, F, should be increased to produce noisy vectors with bigger difference for the crossover operation and purposely a finite number of potential trial solutions that should spread over a certain area containing the acceptable result. The opposite situation occurs when PC is smaller. What FC, the function value change, indicates is similar to PC. So, the bigger the values of PC, the higher the values of F. CR should be adjusted similarly as F: the crossover control parameter, CR, should be increased so that in the crossover operation the possibility that a trial vector parameter comes from a noisy vector is higher, which provides more different potential trial solutions that will be used to replace a certain number of current solutions during the

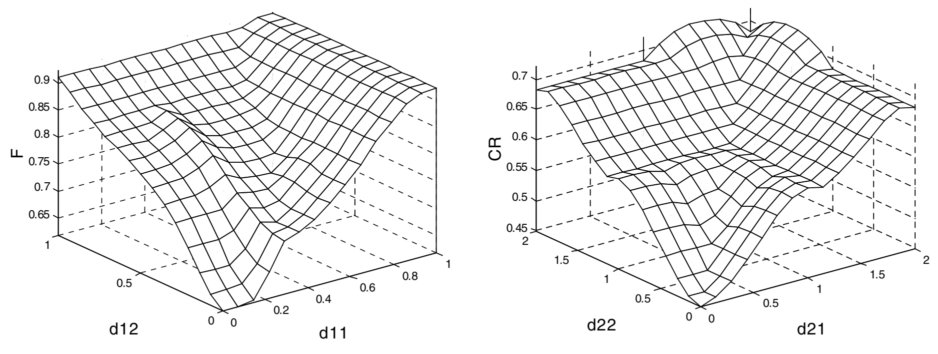


Fig. 6a, b The control surface of Differential Evolution algorithm. **a** The adaptive *F*, **b** the adaptive *CR*

selection when *FC* or *PC* is bigger. The opposite situation occurs when *FC* or *PC* is smaller.

3.3 Control strategy

The fuzzy control strategy is used to map from the given inputs to an output. Mamdani's fuzzy inference method is used. The output of each rule is a fuzzy set. These output fuzzy sets are then aggregated into a single output fuzzy set.

For any input of the controller, i.e, the output of the optimization process transformed to the changes of function values and parameter vectors, A' and C', the fuzzy control OU' (fuzzy inputs $d_{i1} := A'$ and $d_{i2} := C'$ and the fuzzy output $u := OU'$) is obtained via a compositional rule of inference [13]:

$$OU' = (A' \times C') * R \ , \tag{20}$$

where, R is a fuzzy relation specified by the above rules, "$*$" is a fuzzy operator to be defined, i.e., if there are multiple parts to the antecedent, fuzzy logic operators are applied and the resolving of the antecedent to a single number between 0 and 1 occurs. Inserting the values of the membership functions of A' and C' as well as R:

$$OU'(ou_k) = \bigvee_{\substack{a_l \in A^{(i)} \\ c_j \in C^{(i)}}} \left[A'(a_l) \wedge C'(c_j) \wedge R(a_l, c_j, ou_k)\right] \ , \tag{21}$$

i.e.,

$$\begin{aligned} \mu_{OU'}(u) &= \sup_{d_{i1} \in D_{i1}, d_{i2} \in D_{i2}} *(\mu_{A'}(d_{i1}), \mu_{C'}(d_{i2}), \mu_R((d_{i1}, d_{i2}), u)) \\ &= \sup_{\substack{d_{i1} \in D_{i1}, d_{i2} \in D_{i2}, u \in U \\ n=1,2,\ldots,9}} \min\{\mu_{A'}(d_{i1}), \mu_{C'}(d_{i2}), \mu_{A_n}(d_{i1}), \\ &\quad \mu_{C_n}(d_{i2}), \mu_{OU_n}(u)\} \\ &= \max_{n=1,2,\ldots,9} \min\{hgt(A' \cap A_n), hgt(C' \cap C_n), \mu_{OU_n}(u)\}, \\ &\quad for\ all\ u \in U \end{aligned} \tag{22}$$

where $i = \{1, 2\}$ in accordance with the notation in Sect. 3.2.

Choosing for R the matrix representation $R \triangleq (((r_{ljk})))$, for A', C', OU' the corresponding vector representations $A' \triangleq (a'_l)$, $C' \triangleq (c'_j)$, $OU' \triangleq (ou'_k)$, yields with supremum instead of the sum, and with minimum instead of the arithmetic product [2]:

$$ou'_k = \sup_{\substack{l \\ j}} \min\left\{a'_l, c'_j, r_{ljk}\right\} \ . \tag{23}$$

The output fuzzy sets OU_i for each rule are then aggregated into a single output fuzzy set OU.

3.4 Defuzzification strategy

Defuzzification is mapping from a space of fuzzy output U into a space of real output. The input is a fuzzy set μ (the aggregate output fuzzy set) and the output is a

Fig. 7a–c The behaviour of the Adaptive Differential Evolution algorithm for the sphere function (D = 50, NP = 500). **a** F - generation (F starting from 0.9), **b** CR-generation (CR starting from 0.9), **c** Function Objective function value – generation. Figure legend: (●) DE; (-) FADE (DE adapting F and CR)

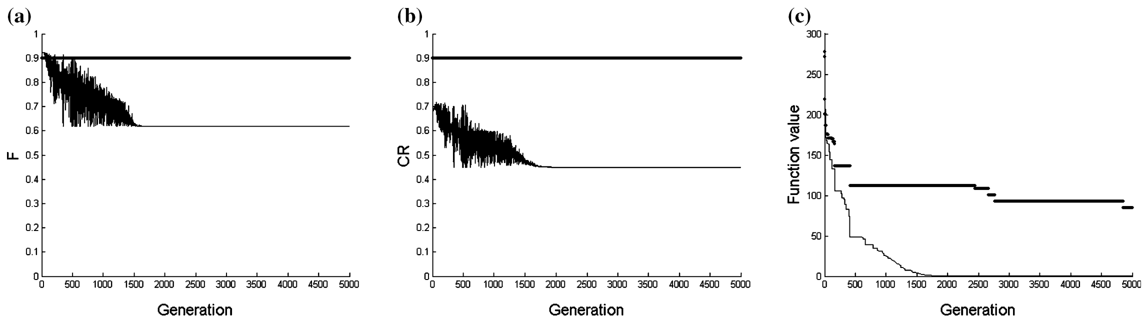


Table 4 Settings for algorithms

Method, Parameters	Settings for Tests No. 1, No. 2, No. 3 and No. 4			
Strategy	DE/rand/1/bin			
Test problems	min $f(\mathbf{X})$			
Tests	1	2	3	4
Number of individuals	$10{\cdot}D$	$10{\cdot}D$	$10{\cdot}D$	$10{\cdot}D$
Crossover operator	0.9	0.9	FLC	FLC
Mutation amplification	0.9	FLC	0.9	FLC

Test No. 1: approximate value according to the DE literature. Each setting is constant per run of DE
Tests No. 2, No. 3 and No. 4: the base for each setting is FLC, which is given by an expert. For each test of DE with fuzzy adaptation the setting was according to the values of DE parameters described in Section III, the initial setting is identical to test No. 1

single number u_o. For this process, different defuzzification strategies can be chosen, like the Centroid of area method, Bisector of area method, Mean of maximum method, Smallest of maximum method and the Largest of maximum method, etc [13]. The centroid defuzzification technique (CDT) was chosen, i.e., the centre of gravity of the fuzzy control OU treated as its numerical representative is calculated as [13]:

$$u_{z,o} = \frac{\int\limits_U u \cdot OU(u)\mathrm{d}u}{\int\limits_U OU(u)\mathrm{d}u} = \frac{\mu_z(w_{z,S}) \cdot w_{z,S} + \mu_z(w_{z,M}) \cdot w_{z,M} + \mu_z(w_{z,B}) \cdot w_{z,B}}{\mu_z(w_{z,S}) + \mu_z(w_{z,M}) + \mu_z(w_{z,B})}, \tag{24}$$

where $z = \{F, CR\}$; μ_z is the activation level obtained within the matching phase; $w_{z,s}, w_{z,M}, w_{z,B}$ are numerical representatives (e.g. the centers of gravity of fuzzy sets); $u_{z,o}$ is the pointwise control value, a defuzzified value that is directly a value of a DE parameter, like: $u_{F,o} = 0.7$ represents the value of the mutation amplification, F, for the 1st FLC.

4 Implementation and experimental study

The setting of FLC presents a subjective view of an expert with respect to the objective characteristics of DE. The setting is viewable as the control surface of DE (See Figure 6). The change in behavior (convergence) of FADE (test No. 4) in accordance with the fuzzy control action can be seen from the characteristics in Fig. 7 with the first test function in the Appendix as an example.

Figure 7 also shows the different behaviors of control parameters F and CR in FADE and DE in a high dimensional problem case ($D = 50$). In Fig. 7a and 7b, while the values of F and CR of FADE change during the first 2000 generations of a total of 5000 generations and then do not change significantly, the values of F and CR of DE are kept constant. These behaviors correspond with the population information of the relevant optimization processes as shown in Fig. 7c. In Fig. 7c, the curves are the best found function values during a single run of 5000 generations of DE and FADE, respectively. The curve of FADE (the dashed line) descents gradually and reaches the optimum during a run while the curve of DE (the dotted line) descents slowly and has a very big difference from that of FADE at the end, which is consistent with that demonstrated in [8].

In the tests, the following settings (in Table 4) were used.

For testing the generalization performance of FADE, a set of standard test functions given in the Appendix was used, which contains the De Jong test functions (slightly modified) as presented in [18] plus some additional functions that present further distinctive features [12] for a global minimizer. A test function denoted as Ackley's path function is used as an example for demonstration. Figure 8 shows the average of 100 independent tests using traditional DE and FADEs with different dimensionalities, like 3 and 50. In Fig. 8b, the curve of DE lies above the curves of FADE, which means that for the given objective function value, the FADE algorithm needs fewer generations than DE does. Experiments with the test functions in the Appendix, the results of which are given in Tables 5 and 6, also show how the FADE algorithms perform compared to the static one. When the

Fig. 8a, b Comparisons of DE with and without adaptation. **a** $D = 3$, **b** $D = 50$. Figure legend: (●) DE; (-) DE adapting CR; (—) DE adapting F; (— -) DE adapting F and CR

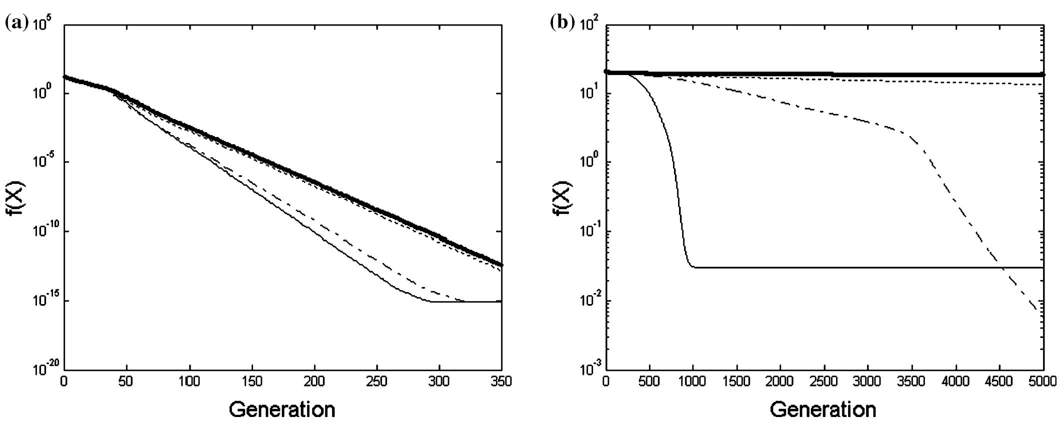
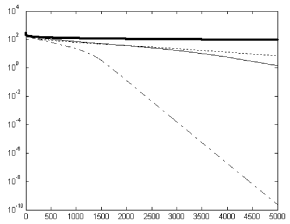
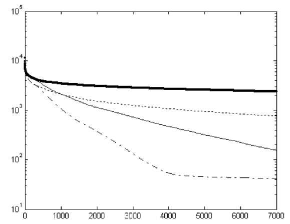
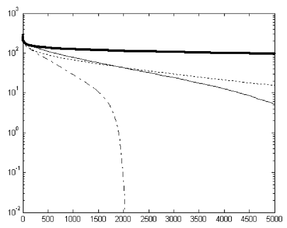
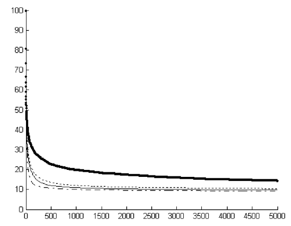
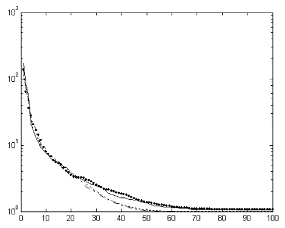
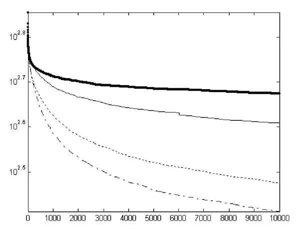
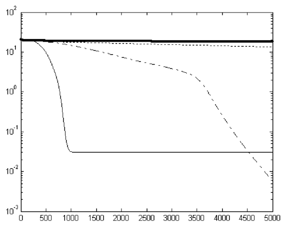
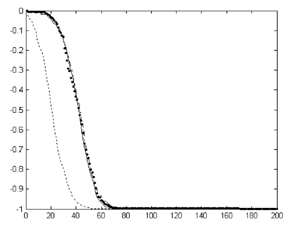
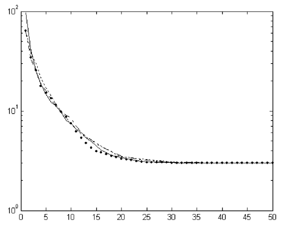
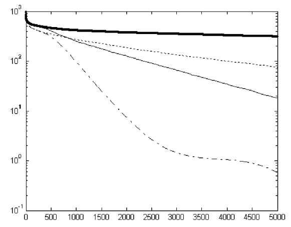


Table 5 Results of experiments

Comparison of DE and FADE			
Values	Curves of best solutions	Values	Curves of best solutions
Test function 1: $f(\mathbf{0}) = 0$ $D = 50$ $NP = 500$ $G = 5000$		Test function 2: $f(\mathbf{1}) = 0$ $D = 50$ $NP = 500$ $G = 5000$	
Test function 3: $f(\mathbf{x}) = 0$ $D = 50$ $NP = 500$ $G = 5000$		Test function 4: $f(\mathbf{0}) <= 15$ $D = 30$ $NP = 300$ $G = 5000$	
Test function 5: $f(\mathbf{-32}) \cong$ 0.998004 $D = 2$ $NP = 20$ $G = 100$		Test function 6: $f(\mathbf{0}) = 0$ $D = 50$ $NP = 500$ $G = 5000$	
Test function 7: $f(\mathbf{0}) = 0$ $D = 50$ $NP = 500$ $G = 5000$		Test function 8: $f(\pi) = 0$ $D = 2$ $NP = 20$ $G = 200$	
Test function 9: $f(0,-1) = 3$ $D = 2$ $NP = 20$ $G = 50$		Test function 10: $f(\mathbf{0}) = 0$ $D = 50$ $NP = 500$ $G = 5000$	

Note: $f(\mathbf{x}^*)$ = the optimum of the specified function. Figure legend: (•) DE; (–) DE adapting *CR*; (—) DE adapting *F*; (— –) DE adapting *F* and *CR*

dimensionality of the problem is high (for example, D = 50 in Table 5), the adaptive ones perform better, specially the one adapting *F* & *CR,* but they do not reveal clear advantages in the case of a lower dimensionality (see also Fig. 8) as shown by functions with a dimensionality of 2 in Tables 5 and 6, and the 4th test function as well.

In particular, for Test functions 1, 2, 3, 6, and 10 with a dimensionality of 50, the FADE algorithms perform better than static ones, while among those three FADE algorithms, the one adapting *F* & *CR* finds the optimum the quickest. For Test function 4 with a dimensionality of 30, the FADE algorithms, which do not differ from each other significantly, perform better than the static one. When the dimensionality is 50, the FADE algorithms perform much faster than the static one. For Test function 5 with a dimensionality of 2, the performance gets better in the order of the static DE, DE adapting *CR,* DE adapting *F,* and DE adapting *F* & *CR*. For Test

Table 6 Numerical results for test functions

Test functions	Dimension	Generation	Test	No. 1	Test	No. 2	Test	No. 3	Test	No. 4
			E	var	E	*var*	*E*	*var*	*E*	*var*
1	3	50	2.1543e-4	6.2882e-8	7.1284e-6	9.0721e-11	1.6216e-4	2.6282e-8	1.1753e-5	1.8808e-10
1	50	5000	9.3767e + 1	5.18246e + 1	1.4380	1.032e-1	6.9915	3.334e-1	2.3508e-10	2.9675e-21
2	2	50	1.1609e-4	8.2268e-8	8.1815e-4	5.0920e-5	6.7e-3	7.1633e-4	2.2e-3	7.7313e-5
2	50	7000	2.3777e + 3	6.7682e + 4	1.5538e + 2	4.0851e + 2	7.6576e + 2	4.3188e + 3	4.1566e + 1	1.82e-2
3	5	30	3.000e-1	2.323e-1	7.00e-2	6.58e-2	6.00e-2	5.70e-2	0	0
3	50	5000	9.5540e + 1	6.1867e + 1	5.2400	2.0226	1.5500e + 1	1.6465	0	0
4	30	5000	1.4312e + 1	6.845e-1	9.7325	1.779e-1	1.0488e + 1	1.689e-1	9.1640	1.502e-1
4	50	5000	6.0845e + 1	2.9003e + 1	1.9857e + 1	3.699e-1	2.4363e + 1	6.317e-1	1.8988e + 1	2.925e-1
5	2	100	1.0675	1.046e-1	1.0573	2.520e-2	1.0079	9.9e-3	0.9980	2.5135e-26
6	2	3000	0.0099	0.0099	0.0298	0.0291	0	0	0	0
		100	0.0100	9.9e-3	0.0299	2.91e-2	9.5408e-9	6.1908e-16	1.1310e-7	4.7674e-13
6	50	10000	4.7268e + 2	2.2229e + 2	4.0644e + 2	1.8956e + 3	2.9900e + 2	9.1675e + 1	2.5849e + 2	9.1682e + 1
7	3	50	0.5090	0.0984	0.2126	0.0360	0.2866	0.0231	0.1556	0.0120
7	50	5000	1.8173e + 1	6.85e-2	1.3358e + 1	8.99e-2	3.02e-2	1.18e-2	5.9e-2	1.2307e-6
8	2	80	−0.9951	1.5334e-40	–0.9971	3.1286e-4	–1.0000	2.0525e-90	–0.9896	1.3e-3
9	2	50	3.0001	3.5365e-8	3.0000	2.4750e-1	3.0007	1.7087e-5	3.0001	3.3510e-7
10	2	180	0.0030	3.7404e-5	0.0031	1.3867e-5	0.0015	9.2406e-6	0.0024	1.1967e-5
10	50	5000	3.1278e+2	7.1679e+2	1.8182e+1	6.8813	7.5031e+1	3.3235e+1	5.776e-1	3.5e-3

Note: "*E*" stands for expectation and "*var*" for variance based on final results of the best objective function values of 100 test runs function 7 with a dimentionality of 50, the adaptive algorithms perform better. Specially, the one adapting F outperforms all the others at the beginning, while the one adapting F & CR exceeds it after a certain amount of generations. In the case of Test function 8 with a dimensionality of 2, the FADE algorithms perform better. While DE adapting CR performs the best amongst them all, the difference between the static DE, DE adapting F, and DE adapting F &CR is small. For Test function 9 with a dimensionality of 2, the FADE algorithms do not have any advantages.

5 Conclusion

This paper presents a fuzzy adaptive differential evolution algorithm for global optimization. This approach adapts the mutation control parameter and/or the crossover control parameter utilizing a fuzzy logic control approach. The parameters of FADE respond to the population's information, i.e., parameter vectors, function values, and their changes. The approach is tested with a set of standard test functions, where it outperforms the original DE when the dimensionality of the problem is high. It is found that a decrease in dimensionality has an effect on the performance in the test cases. In addition, the approach does work better when both adaptive control parameters are used. Based on human knowledge and expertise, FADE is designated to expedite the convergence velocity of DE by the use of adaptive parameters.

The experimental results suggest that the proposed algorithm performs better than those using all fixed parameters. The FADE algorithm, specially when adapting F & CR, converges much faster than the traditional DE particularly when the dimensionality of the problem is high or the problem concerned is complicated. Current research demonstrates that the novel FADE method is a useful approach towards hybridizing DE and FLC. This work presents a new point of view to DE parameter setting and can be understood as a new possible way for fuzzy parameter setting of DE to minimize the user load with regard to parameter choice. However, further work is considered essential in order to collect further evidence of the performance of FADE and any possible shortcomings.

Acknowledgements The work is partially supported by the Fortum Foundation of the Fortum Company, Finland.

6 Appendix: Test functions

Test function 1

Modified First De Jong Function (sphere) is the simplest test function. It is continuous, convex and uni-modal.

$$f(\mathbf{x}) = \sum_{j=1}^{D} x_j^2 \ , \tag{A1}$$

$x_j \in [-5.12,\ 5.12],\ j = 1{:}D$. The minimum is $f(\mathbf{0}) = 0$.

Test function 2

Modified second De Jong function (Rosenbrock's valley) is also known as the Banana function. The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, repeatedly used in assessing the performance of optimization algorithms.

$$f(\mathbf{x}) = \sum_{j=1}^{D-1} \left(100 * (x_{j+1} - x_j^2)^2 + (1 - x_j)^2\right) \ , \tag{A2}$$

$x_j \in [-2.048,\ 2.047],\ j = 1{:}D$. The minimum is $f(\mathbf{1}) = 0$.

Test function 3

Modified Third De Jong Function (step):

$$f(\mathbf{x}) = \sum_{j=1}^{D} \left(floor(x_j + 0.5)\right)^2 \ , \tag{A3}$$

$x_j \in [-5.12,\ 5.12],\ j = 1{:}D$. The minimum $f = 0$ is located at the plateau having $x_j \in [-0.5,\ 0.5]$.

Test function 4

Modified fourth De Jong Function (quartic):

$$f(\mathbf{x}) = \sum_{j=1}^{D} (j \cdot x_j^4 + \eta) \ , \tag{A4}$$

$x_j \in [-1.28,\ 1.28],\ j = 1{:}D,\ D = 30$. The minimum is $f(\mathbf{0}) <= 30 * E(\eta) = 15$, where $E(\eta)$ is the expectation of the random variable η with uniform distribution in the interval [0, 1].

Test function 5

Fifth De Jong function (Shekel's Foxholes):

$$f(\mathbf{x}) = \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{(i+1) + \sum_{j=0}^{1} (x_j - a_{ij})^6}} \ , \tag{A5}$$

$x_j \in [-65.536,\ 65.536],\ j = 1{:}D,\ D = 2$ with $a_{i0} = \{-32, -16, 0, 16, 32\}$ for $i = 0,\ 1,\ 2,\ 3,\ 4$ and $a_{i0} = a_{i mod 5,0}$ as well as, $a_{i1} = \{-32,\ -16,\ 0,\ 16,\ 32\}$ for $i = 0{,}5{,}10{,}15{,}20$ and $a_{i1} = a_{i+k,1},\ k = 1,\ 2,\ 3,\ 4$. The global minimum is $f(\mathbf{-32}) \cong 0.998004$.

Test function 6

Rastrigin's Function is based on function 1 with the addition of cosine modulation to produce many local minima. This test function is highly multi-modal, and the locations of the local minima are regularly distributed.

$$f(\mathbf{x}) = \sum_{j=1}^{D}\left(x_j^2 - 10\cos\left(2\pi \cdot x_j\right)\right) + 10 \cdot D \ , \tag{A6}$$

$x_j \in [-5.12,\ 5.12]$, $j = 1{:}D$. The global minimum is $f(\mathbf{0}) = 0$.

Test function 7

Ackley's Path Function is a widely used multi-modal function

$$f(\mathbf{x}) = -c_1 \cdot \exp\left(-c_2 \cdot \sqrt{\frac{1}{D} \cdot \sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D} \cdot \sum_{j=1}^{D} \cos(c_3 \cdot x_j)\right) + c_1 + \exp(1), \tag{A7}$$

$x_j \in [-32.768, 32.768]$, $j = 1{:}D$ with $c_1 = 20$, $c_2 = 0.2$, $c_3 = 2{\cdot}\pi$. The global minimum is $f(\mathbf{0}) = 0$.

Test function 8

Easom's function is a unimodal test function, where the global minimum has a small area relative to the search space. The function was inverted for minimization.

$$f(\mathbf{x}) = -\cos x_1 \cdot \cos x_2 \cdot \exp\left(-\left((x_1 - \pi)^2 + (x_2 - \pi)^2\right)\right) \ , \tag{A8}$$

$x_j \in [-100,\ 100]$, $j = 1{:}2$. The global minimum is $f(\boldsymbol{\pi}) = -1$.

Test function 9

Goldstein-Price's function is a global optimization test function.

$$f(\mathbf{x}) = \left(1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right) \cdot \left(30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right), \tag{A9}$$

$x_j \in [-2,\ 2]$, $j = 1{:}2$. The global minimum is $f(0,\ -1) = 3$.

Test function 10

Griewangk's Function: A widely used multimodal test function. Griewangk's function is similar to Rastrigin's function. It has many widespread local minima. However, the location of the minima is regularly distributed.

$$f(\mathbf{x}) = \frac{1}{4000} \cdot \sum_{j=1}^{D} x_j^2 - prod\left(\cos\left(\frac{x_j}{\sqrt{j}}\right)\right) + 1 \ , \tag{A10}$$

$x_j \in$ [-600,600], $j = 1{:}D$. The global minimum $f(\mathbf{0}) = 0$.

References

1. Abbass HA (2002) The self-adaptive pareto differential evolution algorithm. In: Proceedings of the congress on evolutionary computation, pp 831–836
2. Bandemer H (1995) Fuzzy sets, fuzzy logic, fuzzy methods with applications. Wiley, Chichester, pp 27–28, 80–81, and 96–98
3. Eiben AE, Hinterding R, Michalewicz Z (1999) Parameter control in evolutionary algorithms, IEEE Trans. Evolutionary Computation, vol 3, No. 2, pp 124–141
4. Herrera F, Lozano M, Verdegay JL (1995) Tackling fuzzy genetic algorithms. In: Winter G, Periaux J, Galan M, Cuesta P (Eds) Genetic algorithms in engineering and computer science. John Wiley and Sons, pp 167–189
5. Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: Proceedings of the 6th international Mendel conference on soft computing, pp 76–83
6. Lee MA, Takagi H (1993) Dynamic control of genetic algorithms using fuzzy logic techniques. In: Proceedings of the 5th international conference on genetic algorithms, pp 76–83
7. Liu J, Lampinen J (2002) On setting the control parameter of the differential evolution algorithm. In: Proceedings of the 8th international Mendel conference on soft computing, pp 11–18
8. Liu J, Lampinen J (2002) Adaptive parameter control of differential evolution. In: Proceedings of the 8th international Mendel conference on soft computing, pp 19–26
9. Liu J, Lampinen J (2002) A fuzzy adaptive differential evolution algorithm. In: Proceedings of the 17th IEEE region 10 international conference on computer, communications, control and power engineering, Vol I of III, pp 606–611
10. Lopez Cruz IL, Van Willigenburg LG, Van Straten G (2001) Parameter control strategy in differential evolution algorithm for optimal control. In: Proceedings of the international conference on artificial intelligence and soft computing, pp 211–216
11. Matousek R, Osmera P, Roupec J (2000) GA with fuzzy inference system. In: Proceedings of the congress on evolutionary computation, vol 1, pp 646–651
12. Michalewicz Z (1992) Genetic algorithms + data structures = evolution programs. Springer Berlin Heidelberg New York, pp 349–352
13. Pedrycz W (1993) Fuzzy control and fuzzy systems – second, extended, edition. Research Studies Press LTD. Taunton, Somerset, pp 94–110
14. Price KV (1999) An introduction to differential evolution. In: Corne D, Dorigo M, Glover F (Eds) New ideas in optimization. McGraw-Hill, London, pp 79–108
15. Schwefel HP, Bäck T (1998) Artificial evolution: how and why? In: Quagliarella D, Périaux J, Poloni C, Winter G (Eds) Genetic algorithms and evolution strategies in engineering and computer science. John Wiley, New York, pp 1–18
16. Storn R, Price K (1995) Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report TR-95–012, ICSI, March
17. Storn R (1996) On the usage of differential evolution for function optimization. In: Biennial conference of the North American fuzzy information processing society, pp 519–523

18. Storn R, Price K (1997) Differential evolution – a simple evolution strategy for fast optimization. Dr. Dobb's journal 22(4): 18–24 and 78, April
19. Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Global Optim 11(4): 341–359
20. Zaharie D (2002) Critical values for the control parameters of differential evolution algorithms. In: Proceedings of the 8th international Mendel conference on soft computing, pp 62–67
21. Zaharie D (2002) Parameter adaptation in differential evolution by controlling the population diversity. In: Proceedings. of 4th international workshop on symbolic and numeric algorithms for scientific computing, pp 385–397
22. Zimmermann HJ (1986) Fuzzy set theory – and its applications. Kluwer-Nijhoff, Boston, pp 11–12 and 61
23. Šmuc T (2002) Improving convergence properties of the differential evolution algorithms. In: Proceedings of the 8th international Mendel conference on soft computing, pp 80–86