

Self-adaptive and Deterministic Parameter Control in Differential Evolution for Constrained Optimization

Efrén Mezura-Montes and Ana Gabriela Palomeque-Ortiz

Abstract In this Chapter we present the modification of a Differential Evolution algorithm to solve constrained optimization problems. The changes include a deterministic and a self-adaptive parameter control in two of the Differential Evolution parameters and also in two parameters related with the constraint-handling mechanism. The proposed approach is extensively tested by using a set of well-known test problems and performance measures found in the specialized literature. Besides analyzing the final results obtained by the algorithm with respect to its original version, some interesting findings regarding the behavior found in the approach and in the values observed on each of the parameters controlled are also discussed.

1 Introduction

Evolutionary computing (EC) comprises a set of algorithms based on simulating the natural evolution and the survival of the fittest. These algorithms are known as Evolutionary Algorithms (EAs).

Three original EAs were proposed in the 1960's: (1) Genetic Algorithms (GAs) [10], Evolution Strategies (ES) [28] and Evolutionary Programming (EP) [9]. Despite the fact that they arose from different motivations, all of them have been used to solve complex search tasks [12] providing competitive results [1, 7, 23].

In the 1990's Storn and Price proposed a novel EA called Differential Evolution (DE) [27]. DE shares similarities with original EAs e.g. DE uses a population of solutions called vectors to sample the search space; DE also uses a recombination and mutation operators to generate new vectors from the current population and, finally, DE has a replacement process to discard the less fit vectors. Like ES, DE

Efrén Mezura-Montes & Ana Gabriela Palomeque-Ortiz
Laboratorio Nacional de Informática Avanzada (LANIA A.C.), Rébsamen 80, Centro, Xalapa,
Veracruz, 91000, MEXICO, e-mail: emezura@lania.mx, apalomeque@lania.edu.mx

uses real-value vectors to represent solutions (no decoding process is necessary as in traditional GAs with binary encoding). Unlike Gaussian distribution used in ES, DE does not use a pre-defined probability distribution for its mutation operator. Instead, DE uses the current distribution of vectors in the population to define the behavior of the mutation operator, and this seems to be one of its main advantages. Furthermore DE, in its original version, does not perform self-adaptation process to its parameters as ES does with its mutation operator.

The optimization problem in discrete, continuous or even mixed search spaces has been solved by using EAs. However, two shortcomings can be identified in this process: (1) A set of parameter values must be defined by the user and the behavior of the algorithm in the search process depends of these values and (2) in presence of constraints, a constraint-handling mechanism must be added to the EA in order to incorporate feasibility information in the selection and replacement processes, and this mechanism may involve additional parameters to be fine-tuned by the user.

Eiben and Schut [8] proposed a classification of parameter setting techniques: (1) Parameter tuning and (2) parameter control. Besides, parameter control is divided into deterministic, adaptive and self-adaptive. Parameter tuning consists on defining good values for the parameters before the run of an algorithm and then running it with these values. On the other hand, deterministic parameter control aims to modify the parameter values by a deterministic rule e.g. a fixed schedule. Adaptive parameter control aims to modify the parameter values based on some feedback from the search behavior e.g. diversity measure to update the mutation rate. Finally, self-adaptive parameter control encodes the parameter values into the chromosome of solutions and they are subject to variation operators. The expected behavior is that the search process will be able to evolve the solutions of the problems as well as to find the optimal values for the parameters of the algorithm. Eiben and Schut [8] mention that most of the work related to parameter setting is focused on variation operators (mostly on mutation) and population size.

DE, as the remaining EAs, lacks a mechanism to incorporate feasibility information into the fitness value of a given solution. Hence, the selection of an adequate constraint-handling technique for a given EA is an open problem. Coello [4] proposed a taxonomy of mechanisms: (1) Penalty functions, (2) special representations and operators, (3) repair algorithms, (4) separation of objectives and constraints and (5) hybrid methods. Penalty functions [25] decrease the fitness of infeasible solutions as to prefer feasible solution in the selection process. Special representations and operators are designed to represent only feasible solutions and the operators are able to preserve the feasibility of the offspring generated. Repair algorithms aim to transform an infeasible solution into a feasible one. The separation of objectives and constraints consists on using these values as separated criteria in the selection process of an EA [19]; this is opposed to penalty functions, where the values of the objective function and the constraints are joined into one single value. Finally, hybrid methods are a combination of different algorithms and/or mechanisms e.g. fuzzy-logic with EAs, cultural algorithms [15] and immune systems [5].

Research in parameter control for constrained optimization is scarce compared to unconstrained optimization. Furthermore, the research efforts do not usually

consider, with the exception of penalty-function-based approaches, the parameters added with the constraint-handling mechanism.

Based on the aforementioned, three main motivations originated this work: (1) to propose parameter control mechanisms in a competitive EA for constrained optimization by considering parameters of the constraint-handling mechanism (2) to analyze the behavior of these controlled parameter values when solving constrained optimization problems and (3) to know the on-line behavior of the proposed approach by measuring the evaluations required to reach the feasible region and the improvement within it.

We use a very competitive approach for constrained optimization known as Diversity Differential Evolution (DDE) [21] where some of the DE parameter values and also those parameter values of its constraint-handling mechanism are controlled by deterministic and self-adaptive mechanisms. Furthermore, we analyze the behavior of each parameter during the evolutionary process in order to provide some insights about the values they use. A set of 24 test functions [16, 22] and two performance measures [18] found in the specialized literature are used in the experimental design, where the aims are: (1) to compare the performance of the proposed DDE algorithm with respect to its original version and with respect to state-of-the-art approaches (2) to analyze the behavior of each parameter during the process and (3) to know the on-line behavior of the proposed approach compared with the original DDE version.

The Chapter is organized as follows: In Section 2 we formally present the problem of our interest. Section 3 offers a brief introduction to DE. After that, Section 4 presents a review of DE and parameter control in constrained optimization. In Section 5 we detail DDE, the approach which is the base of our study. Then, Section 6 introduces our parameter control proposal. The experimental design, the obtained results and their corresponding discussions are given in Section 7. Finally, Section 8 provides some conclusions and the future work.

2 Statement of the problem

We are interested in the general nonlinear programming problem in which, without loss of generality, we want to:

$$\text{Find } \mathbf{x} \text{ which minimizes } f(\mathbf{x}) \quad (1)$$

subject to:

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m \quad (2)$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, \dots, p \quad (3)$$

where \mathbf{x} is the vector of solutions $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$, and each $x_i \in \mathbf{R}$, $i = 1, \dots, n$ is bounded by lower and upper limits $L_i \leq x_i \leq U_i$. These limits define the search

space of the problem; m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear). If we denote with \mathcal{F} to the feasible region and with \mathcal{S} to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$. For an inequality constraint that satisfies $g_i(\mathbf{x}) = 0$, then we will say that it is active at \mathbf{x} . All equality constraints h_j (regardless of the value of \mathbf{x} used) are considered active at all points of \mathcal{F} . Most constraint-handling approaches used with EAs tend to deal only with inequality constraints. However, in those cases, equality constraints are transformed into inequality constraints of the form:

$$|h_j(\mathbf{x})| - \varepsilon \leq 0 \quad (4)$$

where ε is the tolerance allowed (a very small value).

3 Differential Evolution

DE is a simple, but powerful search engine that simulates natural evolution combined with a mechanism to generate multiple search directions based on the distribution of solutions in the current population. Each vector i in the population at generation G , $\mathbf{x}_{i,G}$, called at this moment of reproduction as the target vector will be able to generate one offspring, called trial vector ($\mathbf{u}_{i,G}$). This trial vector is generated as follows: First of all, a search direction is defined by calculating the difference between a pair of vectors r_1 and r_2 , called “*differential vectors*”, both of them chosen at random from the population. This difference vector is also scaled by using a user-defined parameter called “ $F \geq 0$ ” [27]. This scaled difference vector is then added to a third vector r_3 , called “*base vector*”. As a result, a new vector is obtained, known as the mutation vector. After that, this mutation vector is recombined with the target vector (also called parent vector) by using discrete recombination (usually binomial crossover) controlled by a crossover parameter $0 \leq CR \leq 1$ whose value determines how similar the trial vector will be with respect to the target vector. There are several DE variants [27]. However, the most known and used is DE/rand/1/bin, where the base vector is chosen at random, there is only a pair of differential vectors and a binomial crossover is used. The detailed pseudocode of this variant is presented in Figure 1.

4 Related Work

There are previous works on DE for constrained optimization. Lampinen used DE/rand/1/bin variant to tackle constrained problems [14] by using Pareto dominance in the constraints space, Mezura et al. [20] proposed to add Deb’s feasibility rules [6] into DE to deal with constraints. Kukkonen & Lampinen [13] improved

```

Begin
  G=0
  Create a random initial population  $\mathbf{x}_{i,G} \forall i, i = 1, \dots, NP$ 
  Evaluate  $f(\mathbf{x}_{i,G}) \forall i, i = 1, \dots, NP$ 
  For G=1 to MAX_GEN Do
    For i=1 to NP Do
      Select randomly  $r_1 \neq r_2 \neq r_3$  :
       $j_{rand} = \text{randint}(1, D)$ 
      For j=1 to n Do
        If ( $\text{rand}_j[0, 1) < CR$  or  $j = j_{rand}$ ) Then
           $u_{i,j,G+1} = x_{r_3,j,G} + F(x_{r_1,j,G} - x_{r_2,j,G})$ 
        Else
           $u_{i,j,G+1} = x_{i,j,G}$ 
        End If
      End For
      If ( $f(\mathbf{u}_{i,G+1}) \leq f(\mathbf{x}_{i,G})$ ) Then
         $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G+1}$ 
      Else
         $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$ 
      End If
    End For
    G = G + 1
  End For
End

```

Fig. 1 “DE/rand/1/bin” pseudocode. $\text{rand}[0, 1)$ is a function that returns a real number between 0 and 1. $\text{randint}(\text{min}, \text{max})$ is a function that returns an integer number between min and max. NP , MAX_GEN , CR and F are user-defined parameters. n is the dimensionality of the problem.

its DE-based approach now to solve constrained multiobjective optimization problems. Zielinsky & Laur also used Deb’s rules [6] in DE to solve some constrained optimization problems.

Other search techniques have been combined with DE. A gradient-based mutation with DE by Takahama & Sakai was recently proposed [32]. A combination of Particle Swarm Optimization and DE (called PESO+), where the DE mutation operator is considered as a turbulence operator, was proposed by Muñoz-Zavala et al. [26]. Other authors have proposed novel DE variants for constrained optimization [22] or multi-population DE approaches [33].

On the other hand, there are some studies regarding parameter control in DE for constrained optimization. Brest et al. [2] proposed an adaptive parameter control to two DE parameters (F and CR). Huang et al. [11] presented an adaptive approach to choose the most suitable DE variant to generate new trail vectors in constrained search spaces. In this proposal, DE parameters (F , K and CR) were also adapted. Liu & Lampinen [17] proposed to adapt DE parameters by means of Fuzzy Logic.

Besides controlling DE parameters, in this chapter two parameters related with the constraint-handling mechanism are controlled and analyzed. Furthermore, two performance measures help to understand the impact of the control process in the performance and behavior of the approach.

5 Diversity Differential Evolution

Based on the very competitive performance shown by DE in global optimization problems [3], an adapted version to solve numerical optimization problems in presence of constraints, called Diversity Differential Evolution (DDE) was proposed in [21]. Three simple modifications were made to the original DE/rand/1/bin (detailed in Figure 1):

1. The probability of a target vector to generate a better trial vector is increased by allowing it to generate NO offspring in the same generation.
2. A simple constraint-handling mechanism based on feasibility rules [6] is added to bias the search to the feasible region of the search space.
 - a. Between 2 feasible vectors, the one with the highest fitness value wins.
 - b. If one vector is feasible and the other one is infeasible, the feasible vector wins.
 - c. If both vectors are infeasible, the one with the lowest sum of constraint violation is preferred
 $(\sum_{i=1}^m \max(0, g_i(\mathbf{x})))$.
3. A selection ratio parameter $0 \leq S_r \leq 1$ is added to control the way vectors will be selected. Based on the S_r value the selection will be made based only in the value of the objective function $f(\mathbf{x})$ regardless of feasibility. Otherwise, the selection will be made based on the feasibility rules described before.

The detailed pseudocode of DDE is presented in Figure 2

6 Self Adaptive Diversity Differential Evolution

As it can be noted in the pseudocode presented in Figure 2, DDE adds two parameters (NO and S_r) to the original four parameters used in DE (NP , MAX_GEN , CR and F). Therefore, in this work, two parameter control mechanisms are proposed as to keep the user from defining the values of four (out of six) parameters. Three parameters are self-adapted and one of them uses a deterministic control. Furthermore, the behavior of these parameters and the online performance of the new approach are analyzed.

6.1 Self-adaptive parameter control

In order to get a self-adaptive parameter control, the parameters must be encoded within the solution of the problem. Motivated on the way Evolution Strategies work [30], three parameters are encoded in each solution: F , CR and NO as shown in Figure 3.

```

Begin
G=0
Create a random initial population  $\mathbf{x}_{i,G} \forall i, i = 1, \dots, NP$ 
Evaluate  $f(\mathbf{x}_{i,G}) \forall i, i = 1, \dots, NP$ 
For G=1 to MAX_GENERATIONS Do
  F=rand[0.3,0.9]
  For i=1 to NP Do
    For k=1 to NO Do
      Select randomly  $r_1 \neq r_2 \neq r_3 \neq i$ 
       $j_{rand} = \text{randint}(1, D)$ 
      For j=1 to n Do
        If ( $\text{rand}_j(0, 1) < CR$  or  $j = j_{rand}$ ) Then
          childj =  $x_{r_3,j,G} + F(x_{r_1,j,G} - x_{r_2,j,G})$ 
        Else
          childj =  $x_{i,j,G}$ 
        End If
      End For
      If k > 1 Then
        If (child is better than  $\mathbf{u}_{i,G+1}$ 
        based on the three selection criteria) Then
           $\mathbf{u}_{i,G+1} = \text{child}$ 
        End If
      Else
         $\mathbf{u}_{i,G+1} = \text{child}$ 
      End For
      If flip( $S_r$ ) Then
        If ( $f(\mathbf{u}_{i,G+1}) \leq f(\mathbf{x}_{i,G})$ ) Then
           $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G+1}$ 
        Else
           $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$ 
        End If
      Else
        If ( $\mathbf{u}_{i,G+1}$  is better than  $\mathbf{x}_{i,G}$ 
        based on the three selection criteria) Then
           $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G+1}$ 
        Else
           $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$ 
        End If
      End If
    End For
    G = G + 1
  End For
End

```

Fig. 2 DDE pseudocode. The steps modified with respect to the original DE algorithm are highlighted with an arrow. $\text{randint}(min, max)$ returns an integer value between min and max . $\text{rand}(0, 1)$ returns a real number between 0 and 1. Both functions adopt a uniform probability distribution. $\text{flip}(W)$ returns 1 with probability W . NP , MAX_GEN , CR , F , NO and S_r are user-defined parameters. n is the dimensionality of the problem.

$X_{1,1,G}$...	$X_{1,n,G}$	$F_{1,G}$	$CR_{1,G}$	$NO_{1,G}$
$X_{2,1,G}$...	$X_{2,n,G}$	$F_{2,G}$	$CR_{2,G}$	$NO_{2,G}$
⋮	⋮	⋮	⋮	⋮	⋮
$X_{NP,1,G}$...	$X_{NP,n,G}$	$F_{NP,G}$	$CR_{NP,G}$	$NO_{NP,G}$

Fig. 3 Encoded solutions including three parameters to be self-adapted.

Now, each solution has its own F , CR and NO values and these values are subject to differential mutation and crossover. The process is explained in Figure 4, where the trial vector in Diversity Differential Evolution will inherit the three parameter values from the target vector if the last decision variable was taken from it. On the other hand, the values for each parameter will be calculated by using the differential mutation operator i.e. they will be inherited from the mutation vector. The decision variables are handled as in traditional DE, however, the CR parameter value used in the process is that of the target vector.

<p>If (the last decision variable was inherited from the target vector) Then $child_j F = F_{i,G}$ $child_j CR = CR_{i,G}$ $child_j NO = NO_{i,G}$ Else $child_j F = F_{r_3,G} + F_{i,G}(F_{r_1,G} - F_{r_2,G})$ $child_j CR = CR_{r_3,G} + F_{i,G}(CR_{r_1,G} - CR_{r_2,G})$ $child_j NO = NO_{r_3,G} + F_{i,G}(NO_{r_1,G} - NO_{r_2,G})$ End If</p>

Fig. 4 Differential mutation applied to the self-adapted parameters. Note that the F value for the target vector $F_{i,G}$ is used in all cases where differential mutation is used.

6.2 Deterministic parameter control

Recalling the SR parameter explanation in Section 5, this parameter controls the percentage of comparisons made between pairs of vectors by only considering the objective function value, regardless of feasibility information. Therefore, it affects the bias in the search [29]. Higher SR values keep infeasible solutions located in promising areas of the search space, whereas lower SR values help to reach the feasible region by using Deb's rules [6].

Based on this behavior, the SR parameter is controlled by a fixed schedule. A simple function is used to decrease the value for this parameter in such a way that initial higher values allow DDE to focus on searching promising regions of the search space, regardless of feasibility, with the aim to approach the feasible region from a more convenient area. Later in the process, the SR values will be lower, assuming the feasible region has been reached and that it is more important to keep good feasible solutions. The range within SR values will be considered is the following: $[0.45, 0.65]$. At each generation, the SR value will be decreased based on the expression in Equation 5:

$$SR^{(t+1)} = SR^t - \Delta_{SR} \quad (5)$$

where $SR^{(t+1)}$ is the new value for this parameter, SR^t is the current SR value, Δ_{SR} is the amount decreased from this value at each generation and calculated as indicated in Equation 6:

$$\Delta_{SR} = \left(\frac{SR^0 - SR^{G_{max}}}{G_{max}} \right) \quad (6)$$

where SR^0 represents the initial value for SR , and $SR^{G_{max}}$ its last value in a given run.

The detailed pseudocode of Diversity Differential Evolution with the parameter control techniques, called Adaptive-DDE (A-DDE) is shown in Figure 5.

7 Experiments and Results

In order to extensively test the A-DDE algorithm, six experiments are conducted. The first experiment compares the final results of A-DDE with respect to the original DDE (called Static DDE). In order to verify that the self-adaptive mechanism is not equivalent to just generating random values for them within suggested ranges, the second experiment compares A-DDE with respect a special DDE version, called Static2 DDE, where the values for the four controlled parameters in A-DDE are just generated at random (by using a uniform distribution) within the same ranges used in A-DDE. The third experiment analyzes the convergence graphs for Static DDE, Static2 DDE and A-DDE. The fourth experiment includes the graphs for the three self-adapted parameters (F , CR and NO) in order to know which values are they taking as to provide competitive results. The fifth experiment compares Static DDE, Static2 DDE and A-DDE by using two performance measures for constrained optimization in order to know (1) how fast the feasible region is reached and (2) the ability of each DE algorithm to improve inside the feasible region (difficult for most EAs as analyzed in [18]). The two measures are the following:

1. Evals [14]: The number of evaluations (objective function and constraints) required to generate the first feasible solution are counted. A lower value is preferred because it means a faster approach to the feasible region.
2. Progress Ratio [18]: Originally proposed by Bäck for unconstrained optimization [1]. It is a measure of improvement inside the feasible region by using the objective function values of the first and the best feasible solution reached so far

at the end of the process. The formula is as follows: $Pr = \left| \ln \sqrt{\frac{f_{\min}(G_{ff})}{f_{\min}(T)}} \right|$, where

$f_{\min}(G_{ff})$ is the objective function value of the first feasible solution found and $f_{\min}(T)$ is the objective function value of the best feasible solution found in all the search so far. A higher value means a better improvement inside the feasible region.

```

Begin
  G=0
  ⇒ Create a random initial population  $\mathbf{X}_{i,G} \forall i, i = 1, \dots, NP$ 
  Evaluate  $f(\mathbf{X}_{i,G}) \forall i, i = 1, \dots, NP$ 
  ⇒ Select randomly  $SR \in (0.45, 0.65)$ 
  ⇒ Select randomly  $SR_{Gmax} \in (0.0, 0.5)$ 
  For G=1 to  $G_{max}$  Do
    For i=1 to NP Do
      ⇒ For k=1 to  $NO_{i,G}$  Do
        Select randomly  $r_1 \neq r_2 \neq r_3$  :
         $j_{rand} = \text{randint}(1, D)$ 
        For j=1 to D Do
          ⇒ If ( $\text{rand}_j[0,1] < CR_{i,G}$  or  $j = j_{rand}$ ) Then
             $child_{j,G} = x_{r_3,j,G} + F_{i,G}(x_{r_1,j,G} - x_{r_2,j,G})$ 
            ban=0
          Else
             $child_{j,G} = x_{r_j,j,G}$ 
            ban=1
          End If
        End For
        ⇒ If (ban==1) Then
           $child_{F,G} = F_{i,G}$ 
           $child_{CR,G} = CR_{i,G}$ 
           $child_{n_0,G} = n_{0i,G}$ 
        Else
           $child_{F,G} = F_{r_3,G} + F_{i,G}(F_{r_1,G} - F_{r_2,G})$ 
           $child_{CR,G} = CR_{r_3,G} + F_{i,G}(CR_{r_1,G} - CR_{r_2,G})$ 
           $child_{NO,G} = NO_{r_3,G} + F_{i,G}(NO_{r_1,G} - NO_{r_2,G})$ 
        End If
        If  $k > 1$  Then
          If ( $child$  is better than  $u_{i,G+1}$ 
            (Based on three selection criteria)) Then
             $u_{i,G+1} = child$ 
          End If
          Else
             $u_{i,G+1} = child$ 
          End If
        End For
        If flip(SR)
          If ( $f(u_{i,G+1}) \leq f(x_{i,G})$ ) Then
             $x_{i,G+1} = u_{i,G+1}$ 
          Else
             $x_{i,G+1} = x_{i,G}$ 
          End If
        Else
          If ( $u_{i,G+1} \leq x_{i,G}$ 
            (Based on three selection criteria)) Then
             $x_{i,G+1} = u_{i,G+1}$ 
          Else
             $x_{i,G+1} = x_{i,G}$ 
          End If
        End If
      End For
       $G = G + 1$ 
      ⇒  $SR = SR - \Delta_{SR}$ 
    End For
  End

```

Fig. 5 A-DDE pseudocode. Arrows indicate steps where the parameter control mechanisms are involved.

Finally, the sixth and last experiment compares the final results obtained by A-DDE with those reported by some state-of-the-art algorithms. In the first five experiments 24 well-known minimization test problems were used. These problems are used to test EAs in constrained search spaces. Details of the problems can be found in [16]. A summary of their features can be found in Table 1.

Table 1 Details of the 24 test problems. “ n ” is the number of decision variables, $\rho = |\mathcal{F}|/|\mathcal{S}|$ is the estimated ratio between the feasible region and the search space [24], LI is the number of linear inequality constraints, NI the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints. a is the number of active constraints at the optimum.

Prob.	n	Type of function	ρ	LI	NI	LE	NE	a
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.9971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000%	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3
g14	10	nonlinear	0.0000%	0	0	3	0	3
g15	3	quadratic	0.0000%	0	0	1	1	2
g16	5	nonlinear	0.0204%	4	34	0	0	4
g17	6	nonlinear	0.0000%	0	0	0	4	4
g18	9	quadratic	0.0000%	0	12	0	0	6
g19	15	nonlinear	33.4761%	0	5	0	0	0
g20	24	linear	0.0000%	0	6	2	12	16
g21	7	linear	0.0000%	0	1	0	5	6
g22	22	linear	0.0000%	0	1	8	11	19
g23	9	linear	0.0000%	0	2	3	1	6
g24	2	linear	79.6556%	0	2	0	0	2

For all the experiments the results are based on 30 independent runs for each DE algorithm for each test problem. The number of evaluations performed by each DDE version is 180,000 in order to promote a fair comparison (except experiment 6, where the results by the state-of-the-art algorithms were taken directly from the specialized literature). A tolerance value for equality constraints $\varepsilon = 1E-4$ was used.

7.1 Experiment 1

In the first experiment Static DDE and A-DDE are compared. The parameters used for each algorithm were the following:

1. Static DDE
 - $NP = 60$ y $GMAX = 600$
 - $SR = 0.45$
 - $NO = 5$, $CR = 0.9$ and $F \in (0.3, 0.9)$ generated at random.
2. A-DDE.
 - $NP = 60$ y $GMAX = 600$
 - $SR \in (0.45, 0.65)$, $SR^{G_{max}} \in (0.0, 0.5)$, randomly generated on each independent run and this value is controlled by the deterministic control.
 - $NO \in (3, 7)$, $CR \in (0.9, 1.0)$ and $F \in (0.3, 0.9)$ initially generated at random per each vector in the population and then handled with the self-adaptive control.

The statistical results (best, mean and worst values from a set of 30 independent runs) are summarized in Table 2.

From those results in Table 2, A-DDE was able to maintain the good performance of the original DDE in fourteen test problems (g01, g03, g04, g05, g06, g07, g08, g09, g11, g12, g15, g16, g18, g24). Furthermore, the statistical values were improved in six problems (g10, g14, g17, g19, g21 and g23). Finally, in problems g02 and g13 the results, mostly in the average and worst values, were not better than those provided by Static DDE. Problems g20 and g22 remained unsolved by A-DDE.

7.2 Experiment 2

A-DDE is now compared with respect to Static2 DDE, where the four parameters to be controlled in A-DDE are just generated at random between the same intervals in Static2 DDE. The parameter values used in A-DDE were the same reported in experiment 1 in Section 7.1. The parameters used in Static2 DDE were the following:

- Static2 DDE
 - $NP = 60$ y $GMAX = 600$
 - $SR \in (0.45, 0.65)$ generated at random instead of using the deterministic parameter control.
 - $NO \in (3, 7)$, $CR \in (0.9, 1.0)$ and $F \in (0.3, 0.9)$ also generated at random instead of using the self-adaptive parameter control.

Table 2 Comparison of results obtained with the original DDE version with static parameter values (named Static DDE) and the proposed Adaptive-DDE (deterministic and self-adaptive parameter control). “-”, means no feasible solution found. Values in **boldface** mean that the global optimum or best know solution was reached, values in *italic* mean that the obtained result is better (but not the optimal or best known) with respect to the approach compared.

Test problem	Best known solution	Best		Mean		Worst	
		Adaptive DDE	Static DDE	Adaptive DDE	Static DDE	Adaptive DDE	Static DDE
g01	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000	-15.000
g02	-0.803619	-0.803605	<i>-0.803618</i>	-0.771090	<i>-0.789132</i>	-0.609853	<i>-0.747876</i>
g03	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539
g05	5126.497	5126.497	5126.497	5126.497	5126.497	5126.497	5126.497
g06	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814	-6961.814
g07	24.306	24.306	24.306	24.306	24.306	24.306	24.306
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.63	680.63	680.63	680.63	680.63	680.63	680.63
g10	7049.248	7049.248	7049.248	7049.248	7049.262	7049.248	7049.503
g11	0.75	0.75	0.75	0.75	0.75	0.75	0.75
g12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
g13	0.053942	0.053942	0.053942	0.079627	0.053942	0.438803	<i>0.053961</i>
g14	-47.765	-47.765	-	-47.765	-	-47.765	-
g15	961.715	961.715	961.715	961.715	961.715	961.715	961.715
g16	-1.905	-1.905	-1.905	-1.905	-1.905	-1.905	-1.905
g17	8853.540	8853.540	8853.540	8854.664	<i>8854.655</i>	<i>8858.874</i>	8859.413
g18	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025
g19	32.656	32.656	32.656	32.658	32.666	32.665	32.802
g20	0.096700	-	-	-	-	-	-
g21	193.725	193.725	193.725	193.725	193.733	<i>193.726</i>	193.782
g22	236.431	-	-	-	-	-	-
g23	-400.055	-400.055	-	-391.415	-	-367.452	-
g24	-5.508	-5.508	-5.508	-5.508	-5.508	-5.508	-5.508

The summary of statistical values from a set of 30 independent runs is shown in Table 3.

The results in Table 3 suggest that the effect of the self-adaptive mechanism is not equivalent to just generating random values within convenient limits. A-DDE provided better statistical results (mostly in the mean and worst values from a set of 30 independent runs) in thirteen test problems (g01, g02, g04, g06, g07, g10, g13, g14, g16, g17, g19, g21 and g23). In nine problems the performance was similar between A-DDE and Static2 DDE (g03, g05, g08, g09, g11, g12, g15, g18 and g24). Finally, Static2 DDE was unable to provide better results in any problem.

Table 3 Comparison of results obtained with the DDE version with randomly-generated parameter values (named Static2 DDE) and the proposed Adaptive-DDE (deterministic and self-adaptive parameter control). “-”, means no feasible solution found. Values in **boldface** mean that the global optimum or best know solution was reached, values in *italic* mean that the obtained result is better (but not the optimal or best known) with respect to the approach compared.

Test problem	Best known solution	Best		Mean		Worst	
		Adaptive DDE	Static2 DDE	Adaptive DDE	Static2 DDE	Adaptive DDE	Static2 DDE
g01	-15.000	-15.000	-15.000	-15.000	-14.937	-15.000	-13.917
g02	-0.803619	-0.803605	<i>-0.803610</i>	<i>-0.771090</i>	-0.706674	<i>-0.609853</i>	-0.483550
g03	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30660.237	-30665.539	-30591.889
g05	5126.497	5126.497	5126.497	5126.497	5126.497	5126.497	5126.497
g06	-6961.814	-6961.814	-6961.814	-6961.814	-6959.015	-6961.814	-6877.840
g07	24.306	24.306	24.306	24.306	24.945	24.306	38.903
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.63	680.63	680.63	680.63	680.63	680.63	680.63
g10	7049.248	7049.248	7049.248	7049.248	7073.779	7049.248	7308.826
g11	0.75	0.75	0.75	0.75	0.75	0.75	0.75
g12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
g13	0.053942	0.053942	0.053942	<i>0.079627</i>	0.131458	0.438803	0.438803
g14	-47.765	-47.765	-	-47.765	-	-47.765	-
g15	961.715	961.715	961.715	961.715	961.715	961.715	961.715
g16	-1.905	-1.905	-1.905	-1.905	-1.905	-1.905	-1.900
g17	8853.540	8853.540	8853.540	<i>8854.664</i>	8855.884	<i>8858.874</i>	8859.693
g18	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025	-0.866025
g19	32.656	32.656	32.656	32.658	37.785	32.665	65.993
g20	-	-	-	-	-	-	-
g21	193.725	193.725	193.725	193.725	198.009	<i>193.726</i>	263.444
g22	-	-	-	-	-	-	-
g23	-400.055	-400.055	-400.011	<i>-391.415</i>	-370.854	<i>-367.452</i>	-165.715
g24	-5.508	-5.508	-5.508	-5.508	-5.508	-5.508	-5.508

7.3 Experiment 3

In order to analyze the convergence behavior of each DDE algorithm compared, the convergence graph of the run located in the median value of the 30 independent runs was plotted for each test problem. The graph starts when the first feasible solution is generated. The x -axis represents the generation number and the y -axis is calculated as follows: $\log_{10}(f(\mathbf{x}) - f(\mathbf{x}^*))$, where $f(\mathbf{x})$ is the best feasible solution found in the current generation and $f(\mathbf{x}^*)$ is the optimal solution or best known solution for the problem being solved (see Table 1).

For sake of clarity representative graphs were grouped based on the behavior observed: (1) Test problems where the convergence was similar among Static DDE, Static2 DDE and A-DDE in Figure 6, (2) problems where A-DDE converged to better solutions faster than the other two approaches in Figure 7 and (3) problems where A-DDE got trapped in a local optimum solution in Figure 8.

Regarding Figure 6, besides problems g06, g15 and g16, the convergence behavior was similar in other nine problems (g01, g03, g04, g05, g08, g09, g11, g12 and g24). A-DDE was able to present a better convergence behavior, besides problems g07, g19 and g21 in Figure 7, in other five problems (g10, g13, g14, g18 and g23). Finally, the two problems presented in Figure 8 (g02 and g17) are those where A-DDE got trapped in local optima solutions compared to the other two DDE algorithms.

There is not a clear pattern between convergence behavior and features of a test problem. However, the results indeed show that A-DDE mostly maintained the original DDE competitive convergence behavior and even was able to skip local optimum solutions and providing better results in some problems, most of them in presence of equality constraints (g13, g14, g21 and g23).

7.4 Experiment 4

The results of the experiment to analyze the values taken for the controlled parameters are reported as follows: Two graphs for representative test problems are presented. One graph includes the average value for the F and CR parameters in each generation of the run located in the median value out of 30 independent runs. The other graph presents the average value of the NO parameter of the same run. As in the previous experiment, the graphs are grouped based on the behavior found: (1) Those where the parameter values converged to a specific value in Figure 9, (2) problems where the parameter values oscillated and the final results were better with respect to the compared approaches in Figure 10 and finally (3) test problems where the parameter values oscillated and the final results were not better than those obtained with the other two DDE versions in Figure 11.

It is clear from Figures 9, 10 and 11 that an oscillating behavior was obtained in all cases, This effect was more remarked in F and NO , whereas in CR it is barely noted. Based on these results, the self-adaptive mechanism is able to find that $CR \approx 0.9$, which means trial vectors more similar to the mutation vector and less similar to the target vector, is a suitable value on this set of test functions. On the other hand, $F \in [0.5, 0.7]$ and $NO \in [3, 5]$ are adequate boundaries for the set of constrained problems used in the experiments.

Figure 9 includes representative graphs for test problems where the parameter values reached a single value after converging to an optimum (See Figure 6). Other test problems where the behavior was similar were g04, g08, g09, g11, g12 and g24.

Figure 10 contains graphs where A-DDE provided a better final result (See Figure 7), but required more time to converge. In the same way, the parameter values kept oscillating, helping the search by varying the values, mostly for F and NO . Other test functions with the same type of results were g01, g03, g05, g10, g14, g17, g18 and g23.

Finally, Figure 11 shows those graphs where the parameter values kept varying while A-DDE got trapped in a local optimum solution (See Figure 8).

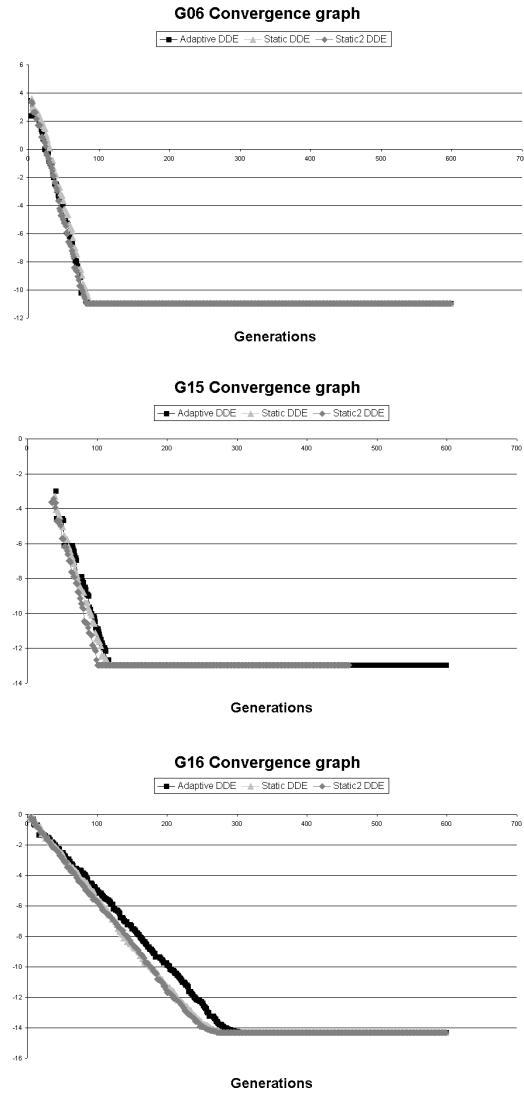


Fig. 6 Convergence graphs for problems g06, g15 and g16 where the behavior was similar in the three compared DE algorithms.

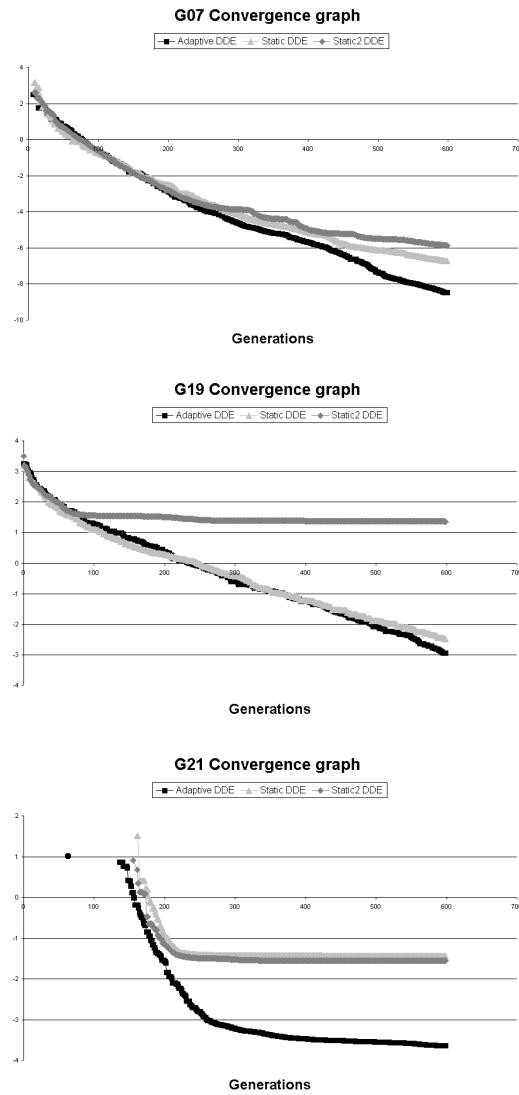


Fig. 7 Convergence graphs for problems g07, g19 and g21 where the behavior of A-DDE was better.

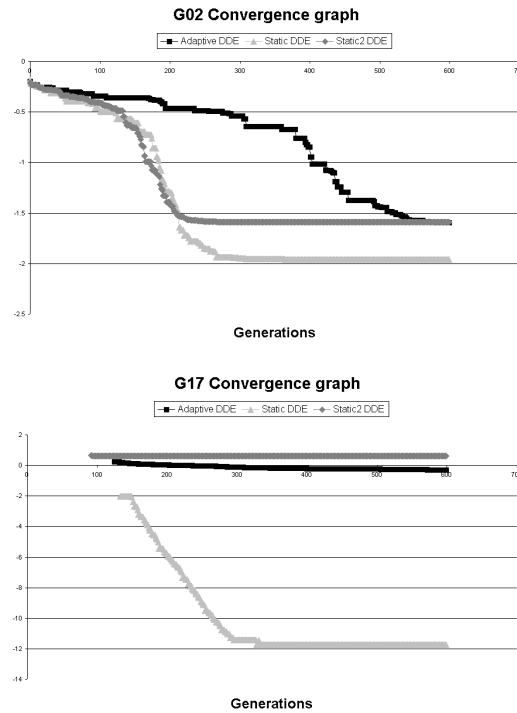


Fig. 8 Convergence graphs for problems g02 and g17 where the behavior of A-DDE was not better than those provided by the compared approaches.

7.5 Experiment 5

The 95%-confidence intervals for the mean value, out of 30 independent runs are presented for both performance measures: Evals in Table 4 and Progress Ratio in Table 5. The aim is to analyze the average performance of the three DDE versions as to establish the effect of the deterministic and self-adaptive control mechanisms.

Regarding the Evals results reported in Table 4, some test problems were not considered in the discussion because the feasible region was reached in the initial population or even in the first generation. These problems are g02, g04, g08, g09, g12, g19 and g24. Problems g20 and g22 are also excluded because no feasible solutions were found by any algorithm. The confidence intervals for Evals indicate that Static2 DDE reached the feasible region faster in eight problems: g03, g05, g06, g11, g13, g15, g17 and g21. Static DDE generated feasible solutions faster in four problems: g01, g07, g16 and g18. A-DDE only found feasible solutions faster in three problems: g10, g14 and g23. These results point out that the deterministic and self-adaptive mechanisms, despite maintaining or improving the quality and

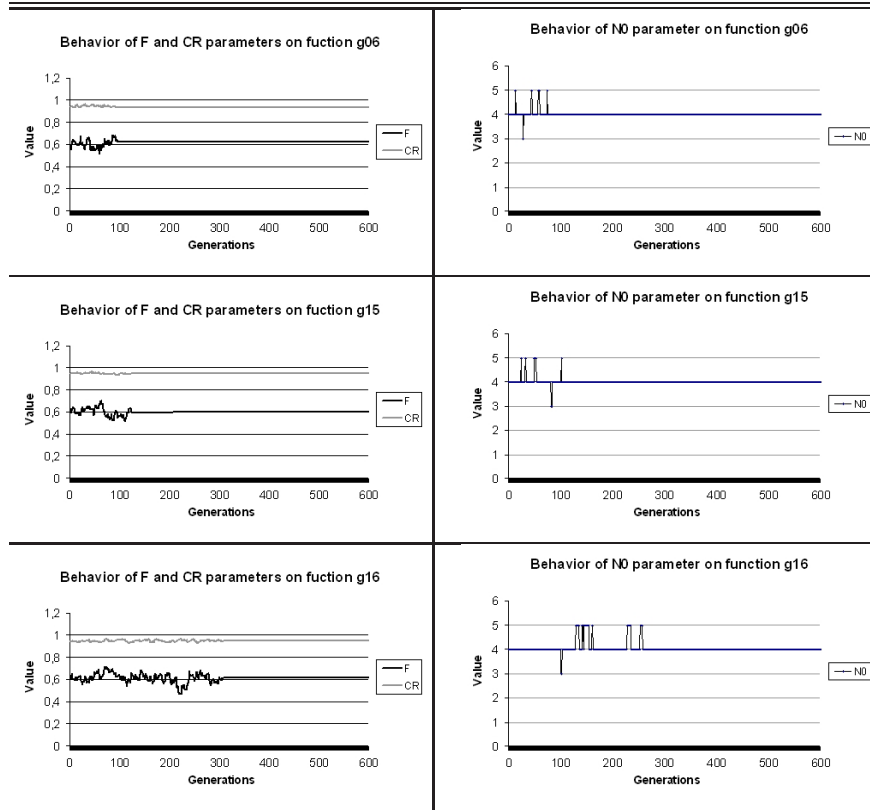


Fig. 9 Average values for F , CR y NO parameters in each generation on the run located in the median value out of 30 independent runs for problems $g06$, $g15$ and $g16$. The parameter values converged to a single value.

consistency of the final results, delayed the arrival to the feasible region with respect to the other two DDE variants.

The results for the Progress Ratio in Table 5, where again problems $g20$ and $g22$ are excluded from discussion because no feasible solutions were found, show that A-DDE obtained a better improvement inside the feasible region in ten problems: $g02$, $g04$, $g09$, $g10$, $g14$, $g15$, $g16$, $g17$, $g21$ and $g23$. Static DDE obtained a better Progress Ratio interval in eight problems $g01$, $g06$, $g07$, $g08$, $g11$, $g12$, $g18$ and $g19$, while Static2 DDE was better in four problems: $g03$, $g05$, $g13$ and $g24$.

After taking more evaluations to reach the feasible region, A-DDE was able to improve the first feasible solution in more problems with respect to Static and Static2 DDE. This behavior suggests that A-DDE enters the feasible region from a more promising region, based on a better exploration of the search space due to the suitable parameter values. However, this issue requires further and more detailed research.

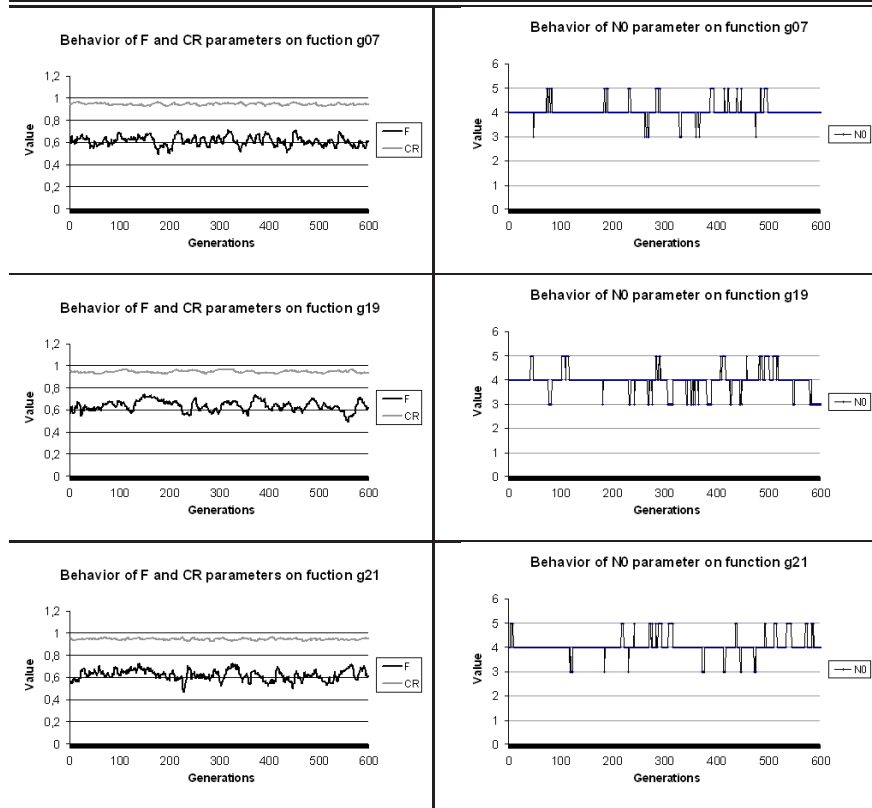


Fig. 10 Average values for F , CR y NO parameters in each generation on the run located in the median value out of 30 independent runs for problems g07, g19 and g21. The parameter values keep oscillating during all the run and the final results were better with respect to both Static DDE versions.

7.6 Experiment 6

In order to compare the final results obtained with A-DDE with respect to state-of-the-art approaches, a summary of statistical values on the first 13 test problems (the most used for comparison in the specialized literature) are presented in Table 6. The approaches used for comparison are: (1) The Generic Framework for constrained optimization by Venkatraman & Yen [35], where the search is divided in two phases, one where only the feasibility of solutions is considered and another one where the feasibility and the optimization of the objective function are taken into account, (2) the self-adaptive penalty function by Tessema & Yen [34], where a parameter-free penalty function is used to deal with the constraints of the problem and (3) a mathematical programming approach combined with a mutation operator by Takahama & Sakai [31]. The comparison shows that A-DDE is indeed very com-

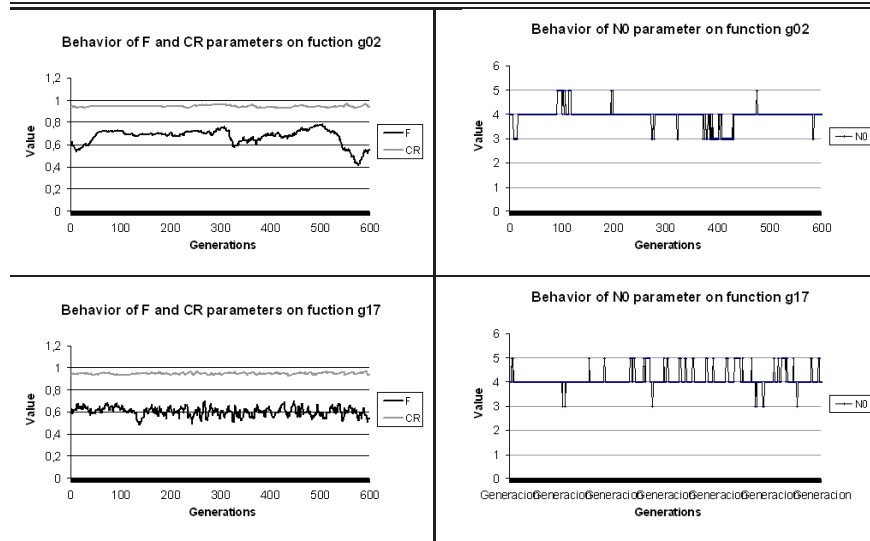


Fig. 11 Average values for F , CR y NO parameters in each generation on the median value out of 30 independent runs for problems g02 and g17. The parameter values keep oscillating during all the run and the final results were not better with respect to both Static DDE versions.

petitive with other evolutionary approaches for constrained optimization based on the quality (best result obtained so far) and consistency (better mean and standard deviation values) of the final results.

8 Conclusions and Future Work

In this chapter, a deterministic and self-adaptive parameter control mechanisms were added to a competitive DE-based approach, called Diversity Differential Evolution (DDE) to solve constrained optimization problems. The proposed approach, called Adaptive-DDE (A-DDE) considered the encoding of three parameters per each vector in the population, two from the original DE (F and CR) and one related with the constraint-handling mechanism (NO , number of trial vectors generated per target vector). Traditional mutation and crossover DE operators were used to self-adapt these three values per each vector. Furthermore, other parameter which controls the bias in the search to keep infeasible solutions located in promising areas of the search space (based on the objective function value, regardless of feasibility), called S_r was deterministically controlled by a decreasing function, focusing first on keeping good infeasible solutions and, after that, maintaining mostly good feasible solutions and discarding those infeasible ones.

Table 4 95%-Confidence intervals for the Evals performance measure in the three DE algorithms compared. The best values are remarked in **boldface**.

Problem	Evals		
	Adaptive DDE	Static	Static2
g01	[5134,5158]	[4349,4375]	[4852,4875]
g02	[62,62]	[61,61]	[61,61]
g03	[5396,5440]	[5038,5084]	[4989,5023]
g04	[63,63]	[62,63]	[65,65]
g05	[18926,18948]	[20825,20850]	[18820,18858]
g06	[1237,1244]	[1282,1291]	[1221,1231]
g07	[2723,2734]	[2432,2443]	[2590,2605]
g08	[143,146]	[152,154]	[163,166]
g09	[163,165]	[188,192]	[165,167]
g10	[4137,4158]	[4236,4259]	[4486,4509]
g11	[2681,2715]	[2393,2419]	[2161,2190]
g12	[79,79]	[78,78]	[85,85]
g13	[30866,30973]	[34440,34551]	[29012,29164]
g14	[106738,106899]	-	-
g15	[11332,11354]	[11528,11545]	[10362,10386]
g16	[1312,1323]	[1159,1168]	[1199,1210]
g17	[33804,33865]	[37488,37553]	[30929,31013]
g18	[10785,10818]	[10282,10310]	[10434,10468]
g19	[64,64]	[62,62]	[64,64]
g20	-	-	-
g21	[38214,38289]	[47745,47856]	[34957,35127]
g22	-	-	-
g23	[59992,60107]	-	-
g24	[62,62]	[62,63]	[62,62]

A-DDE was extensively compared with respect to the original DDE and also with respect to other state-of-the-art approaches. Six experiments were conducted and the following findings were obtained:

- A-DDE maintained the very competitive performance of the original DDE and it also was able to improve the final results in some test problems.
- A-DDE's performance was clearly superior with respect to a DDE version where just random values were generated per each parameter within adequate limits. The self-adaptive mechanism seems to be effective in most of the test problems.
- A-DDE convergence behavior was similar in most cases with respect to the original DDE. However, in some problems with equality constraints A-DDE was able to avoid local optimum solutions.
- An oscillating behavior dominated the self-adaptive mechanism on the three parameters encoded on each vector in the population. The effect was more remarked in $F \in [0.5, 0.7]$ and in $NO \in [3, 5]$, whereas $CR \approx 0.9$ was almost a constant in all the test problems. These results indicate that DDE requires (1) different scale values for the search directions generated in the process, (2) to allow each target vector to generate at least 3 trail vectors and (3) to let them be more similar to the mutation vector instead of being similar to the trial vector.

Table 5 95%-Confidence intervals for the Progress Ratio performance measure in the three DE algorithms compared. The best values are remarked in **boldface**.

Problem	Progress Ratio		
	Adaptive DDE	Static	Static2
g01	[1.004,1.013]	[1.064,1.073]	[0.900,0.906]
g02	[1.071,1.074]	[1.060,1.062]	[1.005,1.008]
g03	[1.199,1.216]	[1.255,1.273]	[1.411,1.427]
g04	[0.77E-01,0.78E-01]	[0.63E-01,0.63E-01]	[0.62E-01,0.63E-01]
g05	[0.83E-06,0.85E-06]	[0.17E-05,0.18E-05]	[0.20E-05,0.21E-05]
g06	[0.375,0.379]	[0.446,0.450]	[0.396,0.400]
g07	[1.689,1.699]	[1.849,1.857]	[1.750,1.761]
g08	[1.465,1.479]	[1.691,1.701]	[1.479,1.492]
g09	[2.622,2.647]	[2.522,2.558]	[2.492,2.528]
g10	[0.468,0.470]	[0.481,0.483]	[0.484,0.487]
g11	[0.52E-01,0.53E-01]	[0.58E-01,0.59E-01]	[0.30E-01,0.31E-01]
g12	[0.102,0.104]	[0.123,0.124]	[0.100,0.102]
g13	[0.11E-02,0.11E-02]	[0.89E-03,0.97E-03]	[0.34E-02,0.36E-02]
g14	[0.47E-01,0.47E-01]	-	-
g15	[0.11E-05,0.12E-05]	[0.10E-05,0.11E-05]	[0.92E-06,0.96E-06]
g16	[0.215,0.216]	[0.205,0.207]	[0.207,0.209]
g17	[0.12E-02,0.12E-02]	[0.41E-03,0.44E-03]	[0.61E-03,0.64E-03]
g18	[0.731,0.738]	[0.763,0.771]	[0.710,0.714]
g19	[3.124,3.129]	[3.150,3.155]	[2.990,2.997]
g20	-	-	-
g21	[0.54E-01,0.55E-01]	[0.31E-01,0.31E-01]	[0.35E-01,0.35E-01]
g22	-	-	-
g23	[0.290,0.294]	-	-
g24	[0.397,0.401]	[0.310,0.313]	[0.506,0.517]

- The results obtained in the two performance measures showed that A-DDE requires more evaluations to reach the feasible region with respect to the original DDE. However, A-DDE is capable of generating a better improvement inside it.
- A-DDE provided very competitive results with respect to some state-of-the-art approaches.

Part of the future work derived from the present research is to analyze more in depth how A-DDE approaches the feasible region with respect to the original DDE as to get more knowledge on the effects of the parameter control mechanisms added. Moreover, we will use A-DDE to solve complex engineering design problems. Finally, we will test other type of special operators in order to self-adapt the parameters encoded in each vector of the population.

References

1. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

Table 6 Statistical results obtained by A-DDE with respect to those provided by state-of-the-art approaches on 13 benchmark problems. Values in **boldface** mean that the global optimum or best know solution was reached, values in *italic* mean that the obtained result is better (but not the optimal or best known) with respect to the approaches compared. No results reported for problems g12 and g13 were found in [35]

Problem/BKS	Statistic	Venkatraman & Yen [35]	Tessema & Yen [34]	Takahama & Sakai [31]	A-DDE
g01 -15.000	Best	-15.000	-15.000	-15.000	-15.000
	Median	-15.000	-14.966	-15.000	-15.000
	Worst	-12.000	-13.097	-15.000	-15.000
	St. Dev.	8.51E-01	7.00E-01	6.40E-06	7.00E-06
g02 -0.803619	Best	-0.803190	-0.803202	-0.803619	-0.803605
	Median	-0.755332	-0.789398	-0.785163	-0.777368
	Worst	-0.672169	-0.745712	<i>-0.754259</i>	-0.609853
	St. Dev.	3.27E-02	1.33E-01	1.30E-02	3.66E-02
g03 -1.000	Best	-1.000	-1.000	-1.000	-1.000
	Median	-0.949	-0.971	-1.000	-1.000
	Worst	-1.000	-0.887	-1.000	-1.000
	St. Dev.	4.89E-02	3.01E-01	8.50E-14	9.30E-12
g04 -30665.539	Best	-30665.531	-30665.401	-30665.539	-30665.539
	Median	-30663.364	-30663.921	-30665.539	-30665.539
	Worst	-30651.960	-30656.471	-30665.539	-30665.539
	St. Dev.	3.31E+00	2.04E+00	4.20E-11	3.20E-13
g05 5126.497	Best	5126.510	5126.907	5126.497	5126.497
	Median	5170.529	5208.897	5126.497	5126.497
	Worst	6112.223	5564.642	5126.497	5126.497
	St. Dev.	3.41E+02	2.47E+02	3.50E-11	2.10E-11
g06 -6961.814	Best	-6961.179	-6961.046	-6961.814	-6961.814
	Median	-6959.568	-6953.823	-6961.814	-6961.814
	Worst	-6954.319	-6943.304	-6961.814	-6961.814
	St. Dev.	1.27E+00	5.88E+00	1.30E-10	2.11E-12
g07 24.306	Best	24.411	24.838	24.306	24.306
	Median	26.736	25.415	24.306	24.306
	Worst	35.882	33.095	24.307	24.306
	St. Dev.	2.61E+00	2.17E+00	1.30E-04	4.20E-05
g08 -0.095825	Best	-0.095825	-0.095825	-0.095825	-0.095825
	Median	-0.095825	-0.095825	-0.095825	-0.095825
	Worst	-0.095825	-0.092697	-0.095825	-0.095825
	St. Dev.	0	1.06E-03	3.80E-13	9.10E-10
g09 680.63	Best	680.76	680.77	680.63	680.63
	Median	681.71	681.24	680.63	680.63
	Worst	684.13	682.08	680.63	680.63
	St. Dev.	7.44E-01	3.22E-01	2.90E-10	1.15E-10
g10 7049.248	Best	7060.553	7069.981	7049.248	7049.248
	Median	7723.167	7201.017	7049.248	7049.248
	Worst	12097.408	7489.406	7049.248	7049.248
	St. Dev.	7.99E+02	1.38E+02	4.70E-06	3.23E-4
g11 0.75	Best	0.75	0.75	0.75	0.75
	Median	0.75	0.75	0.75	0.75
	Worst	0.81	0.76	0.75	0.75
	St. Dev.	9.30E-03	2.00E-03	4.90E-16	5.35E-15
g12 -1.000	Best	NA	-1.000	-1.000	-1.000
	Median	NA	-1.000	-1.000	-1.000
	Worst	NA	-1.000	-1.000	-1.000
	St. Dev.	NA	1.41E-04	3.90E-10	4.10E-9
g13 0.053942	Best	NA	0.053941	0.053942	0.053942
	Median	NA	0.054713	0.053942	0.053942
	Worst	NA	0.885276	0.438803	0.438803
	St. Dev.	NA	2.75E-01	6.90E-02	9.60E-02

2. J. Brest, V. Žumer, and M. S. Maučec. Control Parameters in Self-Adaptive Differential Evolution. In B. Filipič and J. Šilc, editors, *Bioinspired Optimization Methods and Their Applications*, pages 35–44, Ljubljana, Slovenia, October 2006. Jožef Stefan Institute.
3. U. K. Chakraborty, editor. *Advances in Differential Evolution*. Studies in Computational Intelligence Series. Springer-Verlag, Heidelberg, Germany, 2008.
4. C. A. C. Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.
5. N. Cruz-Cortés, D. Trejo-Pérez, and C. A. C. Coello. Handling Constraints in Global Optimization using an Artificial Immune System. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Artificial Immune Systems. 4th International Conference, ICARIS 2005*, pages 234–247, Banff, Canada, August 2005. Springer. Lecture Notes in Computer Science Vol. 3627.
6. K. Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.
7. A. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Verlag, 2003.
8. G. Eiben and M. Schut. New Ways to Calibrate Evolutionary Algorithms. In P. Siarry and Z. Michalewicz, editors, *Advances in Metaheuristics for Hard Optimization*, Natural Computing, pages 153–177. Springer, Heidelberg, Germany, 2008.
9. L. J. Fogel. *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*. John Wiley & Sons, New York, 1999.
10. J. H. Holland. *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, Michigan, 1975.
11. V. L. Huang, A. K. Qin, and P. N. Suganthan. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 324–331, Vancouver, BC, Canada, July 2006. IEEE.
12. K. A. D. Jong. *Evolutionary Computation. A Unified Approach*. MIT Press, 2006.
13. S. Kukkonen and J. Lampinen. Constrained Real-Parameter Optimization with Generalized Differential Evolution. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 911–918, Vancouver, BC, Canada, July 2006. IEEE.
14. J. Lampinen. A Constraint Handling Approach for the Differential Evolution Algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 2, pages 1468–1473, Piscataway, New Jersey, May 2002. IEEE Service Center.
15. R. Landa-Becerra and C. A. Coello Coello. Optimization with Constraints using a Cultured Differential Evolution Approach. In H. B. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, volume 1, pages 27–34, New York, June 2005. Washington DC, USA, ACM Press. ISBN 1-59593-010-8.
16. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, CEC, March 2006. Available at http://www.lania.mx/~emezura/documentos/tr_cec06.pdf.
17. J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Comput.*, 9(6):448–462, 2005.
18. E. Mezura-Montes and C. A. C. Coello. Identifying On-line Behavior and Some Sources of Difficulty in Two Competitive Approaches for Constrained Optimization. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 2, pages 1477–1484, Edinburgh, Scotland, September 2005. IEEE Press.
19. E. Mezura-Montes and C. A. Coello Coello. Constrained Optimization via Multiobjective Evolutionary Algorithms. In J. Knowles, D. Corne, and K. Deb, editors, *Multiobjective Problem Solving from Nature*, pages 53–75. Springer, Heidelberg, 2008.
20. E. Mezura-Montes, C. A. Coello Coello, and E. I. Tun-Morales. Simple Feasibility Rules and Differential Evolution for Constrained Optimization. In R. Monroy, G. Arroyo-Figueroa,

- L. E. Sucar, and H. Sossa, editors, *Proceedings of the 3rd Mexican International Conference on Artificial Intelligence (MICAI'2004)*, pages 707–716, Heidelberg, Germany, April 2004. Springer Verlag. Lecture Notes in Artificial Intelligence No. 2972.
21. E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello. Promising Infeasibility and Multiple Offspring Incorporated to Differential Evolution for Constrained Optimization. In H. Beyer and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, volume 1, pages 225–232, New York, June 2005. Washington DC, USA, ACM Press.
 22. E. Mezura-Montes, J. Velázquez-Reyes, and C. A. C. Coello. Modified Differential Evolution for Constrained Optimization. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 332–339, Vancouver, BC, Canada, July 2006. IEEE.
 23. Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Germany, 2nd edition, 2004.
 24. Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
 25. K. Miettinen, M. Makela, and J. Toivanen. Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *Journal of Global Optimization*, 27(4):427–446, December 2003.
 26. A. E. Muñoz-Zavala, A. Hernández-Aguirre, E. R. Villa-Diharce, and S. Botello-Rionda. PESO+ for Constrained Optimization. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 935–942, Vancouver, BC, Canada, July 2006. IEEE.
 27. K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, 2005.
 28. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.
 29. T. P. Runarsson and X. Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.
 30. H.-P. Schwefel, editor. *Evolution and Optimization Seeking*. Wiley, New York, 1995.
 31. T. Takahama and S. Sakai. Constrained Optimization by Applying the α Constrained Method to the Nonlinear Simplex Method with Mutations. *IEEE Transactions on Evolutionary Computation*, 9(5):437–451, October 2005.
 32. T. Takahama and S. Sakai. Constrained Optimization by the ε Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 308–315, Vancouver, BC, Canada, July 2006. IEEE.
 33. M. F. Tasgetiren and P. N. Suganthan. A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 340–354, Vancouver, BC, Canada, July 2006. IEEE.
 34. B. Tessema and G. G. Yen. A Self Adaptive Penalty Function Based Algorithm for Constrained Optimization. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 950–957, Vancouver, BC, Canada, July 2006. IEEE.
 35. S. Venkatraman and G. G. Yen. A Generic Framework for Constrained Optimization Using Genetic Algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4):424,435, August 2005.