A COMPARATIVE STUDY OF TECHNIQUES FOR HANDLING INTEGER VARIABLES USING DIFFERENTIAL EVOLUTION

Vinicius Kreischer

valmeida@lncc.br

Laboratório Nacional de Computação Científica

Av. Getúlio Vargas, 333 - Quitandinha, 25651-075, Petrópolis, Rio de Janeiro, Brasil

Eduardo Krempser

krempser@lncc.br

Fundação Oswaldo Cruz

Av. Brasil, 4365 - Manguinhos, 21040-900, Rio de Janeiro, Rio de Janeiro, Brasil

Helio J. C. Barbosa

hcbm@lncc.br

Laboratório Nacional de Computação Científica

Av. Getúlio Vargas, 333 - Quitandinha, 25651-075, Petrópolis, Rio de Janeiro, Brasil

Universidade Federal de Juiz de Fora

Rua José Lourenço Kelmer, s/n - Campus Universitário, 36036-900, Juiz de Fora, Minas Gerais, Brasil

Abstract. *This paper presents a review of some popular techniques found in the literature concerning variants of the original Differential Evolution (DE) algorithm aiming at its application to problems with integer variables. Furthermore, three different mapping techniques are applied to a set of 24 unconstrained integer programming problems in order to establish a comparative analysis using a tool known as Performance Profiles. The results show the influence of problem dimensionality in the performance of the variants studied.*

Keywords: *Differential Evolution, Integer Programming, Mapping Techniques*

1. INTRODUCTION

Differential Evolution (DE) is a simple and efficient stochastic metaheuristic for global optimization with continuous variables proposed in (Storn & Price, 1995). As in other Evolutionary Algorithms (EAs), a population of initial candidate solutions, also known as individuals, are perturbed during several generations by evolutionary operators in order to explore the search space and find better solutions. The results obtained by DE in practical real-world problems have drawn the attention of a large number of researchers resulting in a growing number of publications.

As pointed out in (Das & Suganthan, 2011) some of the reasons that make DE an attractive tool for optimization are: the reduced number of control parameters, and the simplicity to implement, when compared to other EAs; the applicability to problems where an explicit analytical formulation is not available or which lack continuity and/or differentiability; and scalability to complex and/or high dimensional problems. Besides, the results obtained by its application to a wide range of practical problems have proven its robustness.

Although DE was initially proposed for problems with continuous variables, a few works have been proposed to extend its applicability to integer valued problems, which can be found in different areas of study. These works rely on two different approaches, called direct and indirect. In the first approach, the variables of the candidate solutions are encoded using integer representation and the mutation and recombination operators are modified to match this representation. In the indirect approach, the variables are kept as real values, and recombination and mutation operators are not altered. However, mapping techniques are employed so that a real to integer conversion is carried out before a fitness value is computed for a candidate solution.

In order to establish a comparative analysis, a tool known as Performance Profiles was used to assess the results obtained for three different mapping techniques applied to a set of 24 unconstrained integer programming problems. The results obtained show the influence of problem dimensionality in the performance of the variants studied.

The rest of the paper is organized as follows. Section 2 presents a brief literature review of some popular techniques concerning variants of the original DE aiming at its application to problems with integer variables. In section 3 the canonical DE algorithm is introduced. Three different mapping techniques are presented in section 4. Details about the computational experiments are given in section 5 and the results obtained are shown in section 6. Some conclusions drawn from the experiments are given in section 7.

2. LITERATURE REVIEW

As mentioned in (Datta & Figueira, 2013), the success obtained by DE as an optimizer in continuous space problems has motivated investigations concerning its application to discrete and integer valued problems too. As a result, some variants of the original DE algorithm have been proposed to handle this class of problem, where they can be classified as indirect or direct approaches.

In the indirect approach, the parameters of the candidate solutions are encoded using real values and the mutation operator is not altered. However, before the evaluation of a new trial solution, these values are mapped to integer ones by employing a predefined method. A relatively simple approach is presented in (Lampinen & Zelinka, 1999), where continuous values

are converted to integer ones by performing a truncation operation. In (Price et al., 2005) it is argued that the rounding of the real values to the nearest integer value might also be a good option. In the algorithm described in (Deng et al., 2011), each decision variable consists of two domains, where one is formed by real values in the narrow range $[0, 1]$ and the another is given by integer values. Through a mapping operator, values present in the domains are connected. Due to their simplicity, the previously described mapping techniques, detailed in section 4, are evaluated in this work.

Although other indirect approaches can be found in the literature, they lack either the simplicity of the original DE algorithm or were designed to be applied exclusively to combinatorial problems. That is the case of the DE variants described in (Li & Zhang, 2014) and (Onwubolu & Davendra, 2006). In the first work, a mixed truncation procedure for generating integer values during the mutation process is employed alongside an orthogonal crossover and a simplified quadratic interpolation. The later work presents a forward and backward transformation for flow shop scheduling problems.

In the direct approach the variables are given by integer values and the mutation operator is altered in order to match this type of representation. In (Dong et al., 2013), for example, a variant of DE is proposed for permutation-based problems making use of the permutation of integers as the individual representation and redefining the addition and subtraction operations of the mutation operator. A method which employs operations on sets instead of the classical arithmetic operations is described in (Maravilha et al., 2013) for general combinatorial optimization problems. A similar approach is detailed in (Liu et al., 2013) where a set-based representation scheme alongside the redefinition of the mutation and crossover operators are employed. In (Prado et al., 2010) the difference between two candidate solutions is defined as a differential list of movements in the search space. As it can be seen, the majority of direct approaches concern the application of DE variants to combinatorial problems and, therefore, are not considered in this work.

3. DIFFERENTIAL EVOLUTION

The main idea of the DE algorithm is to evolve a population of NP individuals, given by vectors of real parameters, over several generations towards a desired solution. This iterative process, depicted in Fig. 1 and Alg.1, begins with the definition of the control parameters (NP, CR, F, GEN) of the algorithm. After that, an initial population is randomly generated and its members are evaluated such that an objective function value is assigned to each one of them. Using the mutation and recombination operators, a new solution, which is a perturbation of an existing one, is generated for each individual in the population. Finally, after the evaluation of the newly-formed solution, a replacement operator selects which individual will be given a place in the population of the next generation. The following subsections (3.1, 3.2, 3.3, and 3.4) describe these operations in detail.

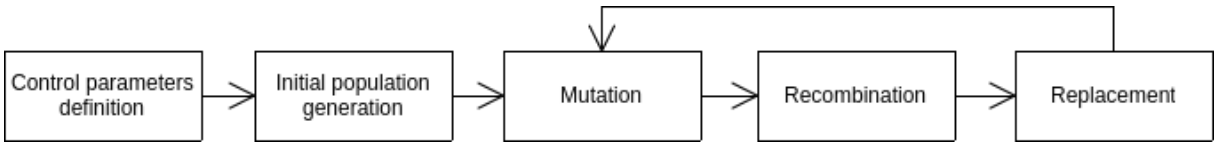


Figure 1: Main stages of the DE algorithm.

Algorithm 1 Basic DE algorithm

```
1: procedure DE(NP, CR, F, GEN)
2:     G = 0
3:     CREATERANDOMINITIALPOPULATION(NP)
4:     Evaluate f(x⃗_{i,G})                                  ▷ ∀i, i = 1, ..., NP
5:     for G = 1 : GEN do
6:         for i = 1 : NP do
7:             Mutation
8:             Recombination
9:             Evaluate f(t⃗_{i,G})
10:            Replacement
11:        end for
12:    end for
13: end procedure
```

3.1 Initial Population Generation

Since DE employs vector differences to search for better solutions it is important that the initial population cover as much as possible the search space constrained by upper ($\overrightarrow{x}_{max}$) and lower ($\overrightarrow{x}_{min}$) bounds. This property is attained by generating uniformly randomized individuals in the initial generation. The j-th parameter of the i-th individual is obtained as

$$x_{i,j,0} = x_{min,j} + \textit{rand}_{i,j}[0,1].(x_{max,j} - x_{min,j}) \tag{1}$$

where $\textit{rand}_{i,j}[0,1]$ represents a uniformly distributed number in the range $[0,1]$.

3.2 Mutation

A mutant vector ($\overrightarrow{m_i}$) is produced for each solution, called target vector, present in the current population, by means of the mutation operator. Although several variants of the original mutation scheme have been proposed since the inception of DE, a common aspect is the use of scaled vector differences to perturb existing solutions. Thus, as pointed out in (Talbi, 2009), the mutation in the DE algorithm can be seen as a self-referential operator that directs the search towards good solutions.

The differentiation between the many existing mutation strategies is given by the way that the vectors employed are selected in the population, and the number of scaled differences used. The mutation scheme for the original DE algorithm, known as $DE/rand/1$, is given by

$$m_{i,j} = x_{r1,j} + F(x_{r2,j} - x_{r3,j}) \tag{2}$$

The indices r_1, r_2, and r_3 are mutually distinct integers randomly generated within the range $[1, NP]$. The scaling factor $F \in (0,1]$ applied to the differential vector is one of the control parameters of the algorithm.

3.3 Recombination

After the mutation operation is performed, a recombination operator takes place in order to build a new solution. This solution, known as trial vector ($\overrightarrow{t_i}$), is obtained through the copy of parameter values either from the mutant vector ($\overrightarrow{m_i}$) or the target vector ($\overrightarrow{t_i}$). The way the parameters are selected from the mutant and target vectors characterize the different models of recombination. This work employs the binomial recombination model, also known as uniform crossover, which is given by

$$t_{i,j} = \begin{cases} m_{i,j}, & \text{if } rand_{i,j}(0,1) \leq CR \text{ or } j = rand_i(0,D) \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{3}$$

where $CR \in [0,1]$ is a control parameter of the algorithm and represents (approximately) the number of values copied from the mutant vector to the trial vector. The value j, obtained through the function $rand_i(0,D)$, denotes a randomly generated index to be copied from the mutant and ensures that the trial vector is not an exact copy of its corresponding target vector.

3.4 Replacement

The replacement operator is responsible for comparing the objective function values of the trial ($\overrightarrow{t_i}$) and target ($\overrightarrow{x_i}$) vectors and determining which one will take a place in the next generation population. For this, an elitist replacement is considered, where the trial vector will replace its corresponding target vector if it has less or equal objective function value (considering a minimization problem). The replacement operator is given by

$$\overrightarrow{x}_{i,G+1} = \begin{cases} \overrightarrow{t}_{i,G+1}, & \text{if } f(\overrightarrow{t}_{i,G+1}) \leq f(\overrightarrow{x}_{i,G}) \\ \overrightarrow{x}_{i,G}, & \text{otherwise.} \end{cases} \tag{4}$$

4. MAPPING TECHNIQUES

The original mutation operator of the DE algorithm was proposed to deal exclusively with real valued problems. Therefore, in order to maintain it unaltered when applied to problems with integer variables, mapping techniques must be employed. These methods utilize specific procedures so that a real to integer conversion is carried out before a fitness value is computed for a candidate solution.

After the conversion, two different approaches can be used: in the first one, the integer values obtained through the mapping are maintained as the individual parameters. In the second approach, these values are employed exclusively in the evaluation of the candidate solution. The last approach is the one used in the computational experiments performed here.

This paper employs three different mapping techniques in order to establish a comparative analysis between them. In the first technique, described in (Deng et al., 2011), each decision variable has two fields, where one is given by float-encoding values belonging to a narrow interval [0,1] and the other represents the integer values utilized to evaluate the individual.

Using the function described in Eq. (5) the float values are decoded into integer values.

$$I(x) = \begin{cases} a, & x \in [0, \frac{1}{n}] \\ ..., & ... \\ b, & x \in [\frac{n-1}{n}, 1] \end{cases} \tag{5}$$

In Eq. (5), a and b denote the lower and upper bounds of the integer variables, respectively, and n represents equally sized subranges of $[0, 1]$. Hence, a one-to-one mapping can be constructed between the subranges and the integers.

The other two approaches employ the rounding to the nearest integer value (Price et al., 2005) or a truncation operation where only the integer part of the number is kept (Lampinen & Zelinka, 1999). Employing these methods, the values -9.76 and 0.2, for example, are mapped to the values -10 and 0 by the rounding operation, and to the values -9 and 0 by the truncation procedure.

5. COMPUTATIONAL EXPERIMENTS

The computational experiments have the objective of providing a comparative analysis between the mapping techniques described in the previous section when employed alongside the original DE scheme (i.e. $DE/rand/1/bin$). For this, a set of twenty-four unconstrained integer programming problems, described at the appendix, were selected. The formulation of this class of problem is given in Eq. (6), where Z is the set of integers, and S is a non necessarily bounded set.

$$\text{min } f(\overrightarrow{x}),\ \overrightarrow{x} \in S \subseteq Z^n \tag{6}$$

The values of the DE control parameters adopted were: $CR = 0.8$, $F = 0.6$, $NP = 50$. Besides that, 50000 objective function calls were allowed. Thirty independent runs, using different seeds for generating pseudo-random values, were performed for each combination mapping technique/optimization problem.

It is also important to note that all the test problems selected present boundary constraints over the decision variables. Therefore, a method to repair the values yielded by the mutation operator that fall outside the allowed set of values must be employed. Equation (7) shows how the values are repaired for the DE version which uses the mapping technique described in (Deng et al., 2011), while Eq. (8) demonstrates the repair method performed by the DE versions which use either the rounding or the truncation procedures.

$$m_{i,j} = \begin{cases} m_{i,j}, & \text{if } 0 \leq m_{i,j} \leq 1 \\ rand(0,1), & \text{otherwise} \end{cases} \tag{7}$$

$$m_{i,j} = \begin{cases} m_{i,j}, \text{ if } x_{j,max} \leq m_{i,j} \leq x_{j,min} \\ x_{min,j} + \textit{rand}_{i,j}[0,1].(x_{max,j} - x_{min,j}), \text{ otherwise} \end{cases} \tag{8}$$

Table 1 presents the dimensionality of the test problems considered in the computational experiments. As it can be seen, functions 1-8 are non-scalable while functions 9-24 are employed

Table 1: Dimensionality of the test problems employed in the computational experiments.

Test Problem	Dimensionality	Test Problem	Dimensionality
1	5	13	10, 25, 50, 75, 100
2	2	14	10, 25, 50, 75, 100
3	2	15	10, 25, 50, 75, 100
4	2	16	10, 25, 50, 75, 100
5	4	17	10, 25, 50, 75, 100
6	4	18	10, 25, 50, 75, 100
7	5	19	10, 25, 50, 75, 100
8	4	20	10, 25, 50, 75, 100
9	10, 25, 50, 75, 100	21	10, 25, 50, 75, 100
10	10, 25, 50, 75, 100	22	10, 25, 50, 75, 100
11	10, 25, 50, 75, 100	23	10, 25, 50, 75, 100
12	10, 25, 50, 75, 100	24	10, 25, 50, 75, 100

using different number of parameters. To make it easier to visualize and to interpret the results of the computational experiments, an analytical tool known as Performance Profiles (Dolan & Moré, 2002) is used. For this, first it is necessary to select a measure ($t_{p,A}$) representative of the relative performance of the algorithm $a_i \in A = \{a_1, ..., a_n\}$ applied to the test problem $p_i \in P = \{p_1, ..., p_m\}$. Given the definition of $t_{p,a}$, the performance ratio $r_{p,a}$ can be described as

$$r_{p,a} = \frac{t_{p,a}}{\min\{t_{p,a} : a \in A\}} \tag{9}$$

Although each $t_{p,a}$ or $r_{p,a}$ is worth considering by itself, it is desirable to assess the performance of the solvers in A on a large set of problems P in a graphical form suitable for human inspection. Denoting the cardinality of a set A by $|A|$ and given the definition of $\rho(\tau)$

$$\rho_{p,a} = \frac{1}{n_p}|\{p \in P : r_{p,a} \leq \tau| \tag{10}$$

then $\rho(\tau)$ is the fraction of the problems in P such that the solver $a_j \in A$ is able to find better solutions in a factor $r_{p,a} \leq \tau$ compared to the best performance observed.

In (Barbosa et al., 2010) it is suggested that the area under the ρ_a curve ($AUC_a = \int \rho_a(t)dt$) is an overall performance/measure for solver a in the problem set P: the larger the AUC the higher the solver efficiency. This property is employed to assess the results obtained in the computational experiments.

6. RESULTS

Tables 2, 3, 4, 5 and 6 demonstrate the results obtained in the computational experiments for the scalable test problems (TP) with 10, 25, 50, 75 and 100 dimensions, respectively. The average (avg), median, and standard deviation (std) values for each combination mapping technique/test problem are presented and the best ones are colored to make the identification easier.

Table 2: Results obtained for the scalable test problems with 10 dimensions.

	Scalable Test Problems (9-24) - 10D								
	(Deng et al., 2011) $[0,1] \Longrightarrow \mathbb{Z}$			(Price et al., 2005) rounding			(Lampinen & Zelinka,1999) truncation		
	avg	median	std	avg	median	std	avg	median	std
TP9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.0000
TP10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP11	1.067	2.000	1.015	1.200	2.000	0.997	2.000	2.000	0.000
TP12	0.133	0.000	0.730	0.900	0.000	2.746	9.000	9.000	0.000
TP13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP15	0.332	0.332	0.077	0.322	0.340	0.077	0.219	0.216	0.084
TP16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP17	0.067	0.000	0.368	0.235	0.000	0.630	0.067	0.000	0.365
TP18	0.000	0.000	0.000	0.000	0.000	0.000	0.967	1.000	0.183
TP19	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
TP20	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
TP21	0.000	0.000	0.000	0.000	0.000	0.000	100.000	100.000	0.000
TP22	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
TP23	0.007	0.000	0.021	0.011	0.000	0.033	0.025	0.017	0.028
TP24	0.000	0.000	0.000	0.000	0.000	0.000	0.240	0.248	0.045

One might note the absence of a table describing the results for the non-scalable test problems. This omission is due to the fact that all mapping techniques obtained the same values (average, median and standard deviation) when evaluated over the set of problems selected.

Figures 2 and 3 demonstrate the performance profile graphs and bar graphs, respectively, obtained for each combination mapping technique/problem regarding the median values of the runs. The presentation of the results was divided based on the dimensionality adopted for the set of test problems. Besides, a separation between the results for non-scalable (Test Problems 1-8) and scalable (Test Problems 9-24) functions was also established.

Through Fig. 2-(f) and Fig. 3-(f) it can be observed that the mapping techniques employed in the experiments yielded the same results for the set of non-scalable test problems. In the following, the allowed number of objective function calls was reduced in order to verify a possible modification in the relative behavior of the techniques. Nevertheless, no alteration in the previous results was observed. This fact might be related to the low dimensionality or complexity of the test problems.

In the analysis of the results obtained for the set of scalable test problems, it is important to highlight the change of the rankings for the mapping techniques regarding the adopted problem dimensionality. Figures 2-(a) and 3-(a) demonstrate the superiority of the technique proposed in (Deng et al., 2011) for the 10-D case. However, a decrease in its performance can be observed as the dimensionality of the test problems is increased. For the truncation procedure, an opposite behavior can be noticed. In the 10-D and 25-D cases this technique produces the worst results among the three selected mapping techniques, but it turns out to be the most efficient in the 50-D, 75-D and 100-D cases. Finally, the rounding procedure exhibits an intermediate performance

Table 3: Results obtained for the scalable test problems with 25 dimensions.

	Scalable Test Problems (9-24) - 25D								
	(Deng et al., 2011) $[0,1] \Longrightarrow \mathbb{Z}$			(Price et al., 2005) rounding			(Lampinen & Zelinka,1999) truncation		
	avg	median	std	avg	median	std	avg	median	std
TP9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP11	15.200	2.000	52.186	1.933	2.000	0.365	2.000	2.000	0.000
TP12	42.867	24.000	37.093	58.567	24.000	51.141	24.000	24.000	0.000
TP13	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP15	0.007	0.000	0.019	0.004	0.000	0.015	0.002	0.000	0.010
TP16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP17	21848.5	21909.1	5130.1	21209.3	20529.0	5487.7	25861.1	25834.3	5322.0
TP18	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP19	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
TP20	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
TP21	0.000	0.000	0.000	0.000	0.000	0.000	604.300	625.000	113.379
TP22	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
TP23	0.000	0.000	0.000	0.000	0.000	0.000	0.007	0.007	0.000
TP24	0.016	0.000	0.086	0.000	0.000	0.000	0.125	0.129	0.024

Table 4: Results obtained for the scalable test problems with 50 dimensions.

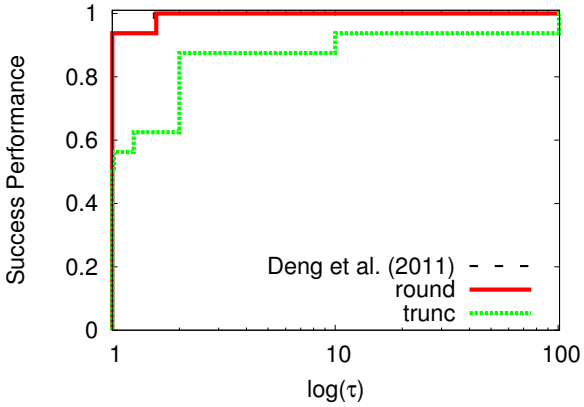
	Scalable Test Problems (9-24) - 50D								
	(Deng et al., 2011) $[0,1] \Longrightarrow \mathbb{Z}$			(Price et al., 2005) rounding			(Lampinen & Zelinka,1999) truncation		
	avg	median	std	avg	median	std	avg	median	std
TP9	16.767	16.500	4.994	16.633	15.500	5.991	2.433	2.000	1.977
TP10	19.900	19.500	4.809	20.567	18.000	8.997	5.400	5.000	4.223
TP11	4887.46	4455.50	3351.45	4705.50	3980.500	2973.71	2.00	2.00	0.00
TP12	838.567	847.500	305.500	812.100	841.500	275.094	49.000	49.000	0.000
TP13	5.133	4.000	2.209	4.600	5.000	2.238	0.000	0.000	0.000
TP14	0.009	0.000	0.021	0.017	0.000	0.026	0.006	0.000	0.017
TP15	1.116	1.105	0.056	1.126	1.116	0.043	1.095	1.090	0.036
TP16	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
TP17	93359.9	93776.5	7872.5	94802.2	96014.5	8431.9	94982.6	94205.7	7039.6
TP18	0.046	0.049	0.028	0.351	0.352	0.023	0.008	0.000	0.012
TP19	20.433	20.000	7.551	20.267	19.000	5.546	23.100	23.000	6.546
TP20	48.300	48.000	10.353	44.833	44.500	9.270	49.967	47.500	10.294
TP21	4285.23	3768.50	2049.44	4955.96	4939.00	1885.06	8262.13	7470.00	2533.15
TP22	20.400	19.500	5.679	20.133	19.000	6.404	22.667	22.000	5.973
TP23	0.736	0.740	0.165	0.744	0.752	0.099	0.807	0.834	0.151
TP24	2.712	2.751	0.206	2.810	2.806	0.173	2.759	2.739	0.219

Table 5: Results obtained for the scalable test problems with 75 dimensions.

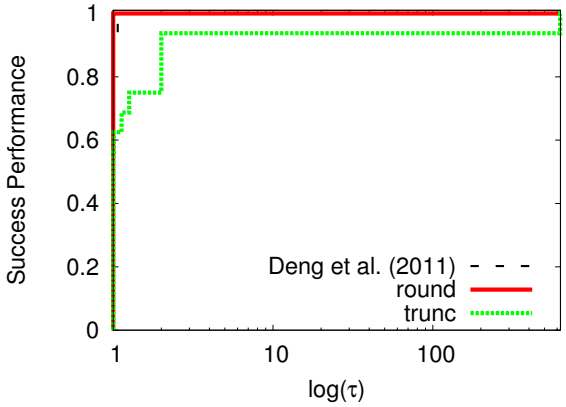
	(Deng et al., 2011) $[0,1] \Longrightarrow \mathbb{Z}$			(Price et al., 2005) rounding			(Lampinen & Zelinka,1999) truncation		
	avg	median	std	avg	median	std	avg	median	std
TP9	111.267	112.000	19.759	107.467	106.000	22.316	80.567	78.000	20.296
TP10	322.467	296.000	101.061	331.867	325.000	98.066	267.300	248.500	114.338
TP11	57151.6	56559.0	15674.2	53464.9	54827.0	12626.1	969.8	264.5	1679.4
TP12	2527.76	2559.50	424.36	2533.40	2664.00	532.60	183.73	74.00	158.05
TP13	24.467	24.000	3.954	24.400	24.000	3.944	4.800	4.000	2.384
TP14	0.293	0.290	0.045	0.284	0.281	0.031	0.172	0.172	0.033
TP15	4.144	3.912	1.090	4.457	4.429	1.193	3.965	3.975	0.872
TP16	1.533	1.000	1.408	0.733	0.000	1.0810	0.000	0.000	0.000
TP17	154200	153798	13471	155810	157390	12394	156837	156781	12197
TP18	550.567	536.500	98.801	506.100	497.000	91.314	525.767	525.000	69.684
TP19	949.20	961.50	204.47	965.90	959.50	289.24	1083.23	1088.00	277.50
TP20	538.933	492.000	110.242	516.733	510.500	105.579	551.233	529.000	84.420
TP21	764458	749213	221419	674249	693781	185856	793284	760557	270943
TP22	972.86	926.00	274.11	1021.03	958.50	257.45	1094.76	1060.50	290.64
TP23	3.144	3.018	0.628	3.170	3.061	0.576	3.381	3.250	0.694
TP24	7.467	7.526	0.624	7.850	7.947	0.892	7.534	7.421	0.855

Table 6: Results obtained for the scalable test problems with 100 dimensions.

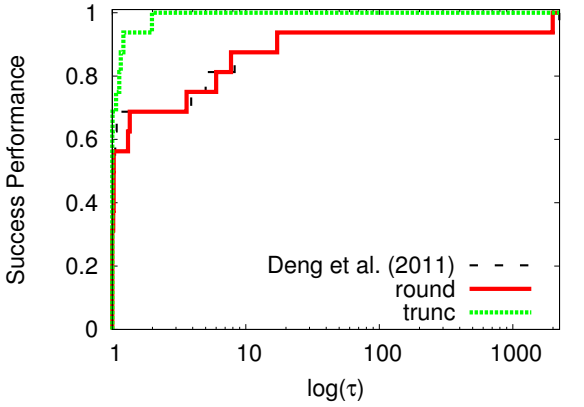
	Scalable Test Problems (9-24) - 100D								
	(Deng et al., 2011) $[0,1] \Longrightarrow \mathbb{Z}$			(Price et al., 2005) rounding			(Lampinen & Zelinka,1999) truncation		
	avg	median	std	avg	median	std	avg	median	std
TP9	265.667	265.500	39.821	258.300	255.500	39.362	230.667	236.000	44.046
TP10	1685.63	1534.00	508.839	1662.03	1482.50	439.764	1374.80	1286.00	441.518
TP11	232033	232253	37678	217506	211858	33731	30040	25803	19267
TP12	5491.13	5352.00	1095.08	4824.50	4800.50	666.98	1258.73	1296.00	622.77
TP13	48.067	46.000	7.956	43.733	42.000	6.617	14.400	14.000	3.692
TP14	0.513	0.532	0.070	0.503	0.500	0.064	0.363	0.352	0.059
TP15	14.551	14.793	3.085	14.787	14.173	3.137	15.787	15.721	4.135
TP16	10.233	11.000	3.683	9.000	9.000	3.301	0.033	0.000	0.183
TP17	230212	230718	10457	225244	226697	15574	220921	221195	11779
TP18	1761.03	1752.00	261.32	1762.40	1771.50	239.23	1818.23	1808.50	300.90
TP19	10259.0	10391.5	2550.4	9683.3	9735.5	2352.6	9655.2	9778.0	2309.6
TP20	1780.53	1752.00	244.85	1798.33	1781.00	243.35	1828.63	1783.50	312.20
TP21	1.62E+7	1.51E+7	3.1E+6	1.42E+7	1.35E+7	3.8E+6	1.57E+7	1.52E+7	3.0E+6
TP22	9532.13	9172.50	3084.42	9376.36	8844.50	2638.91	10569.16	10104.50	2888.48
TP23	23.469	22.540	5.505	22.610	22.565	5.805	23.485	24.745	4.805
TP24	21.188	21.207	0.070	21.211	21.213	0.038	21.159	21.223	0.237



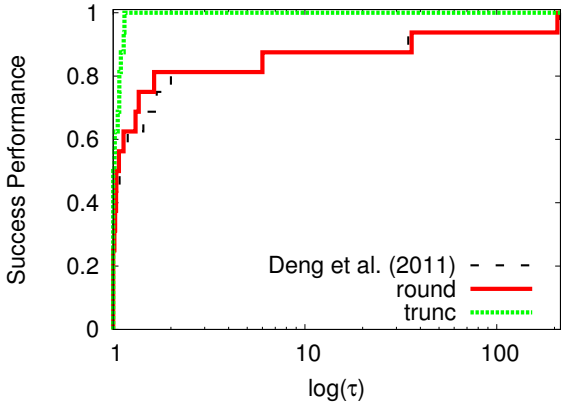
(a) Scalable Test Problems with 10 dimensions



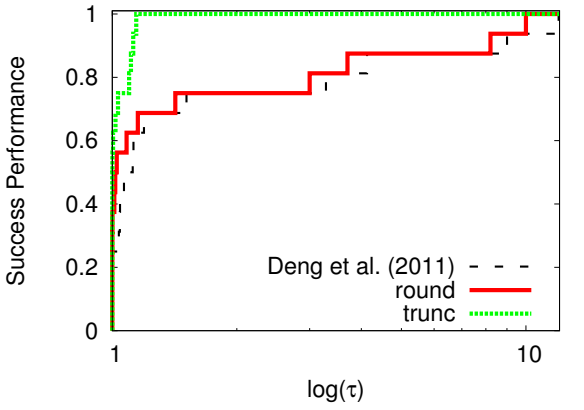
(b) Scalable Test Problems with 25 dimensions



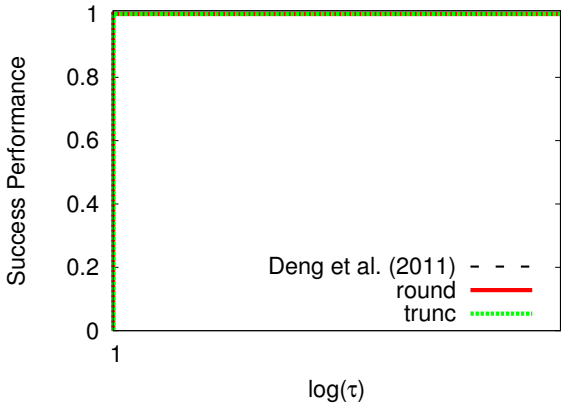
(c) Scalable Test Problems with 50 dimensions



(d) Scalable Test Problems with 75 dimensions

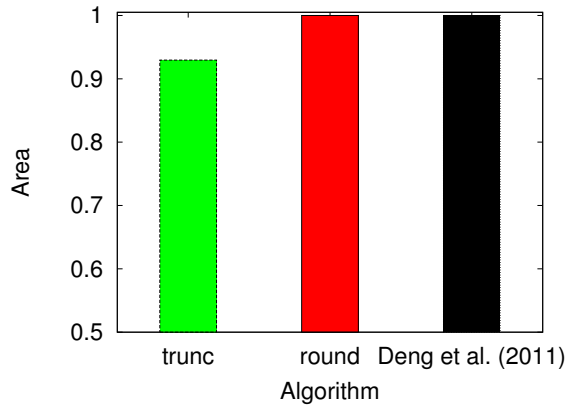


(e) Scalable Test Problems with 100 dimensions

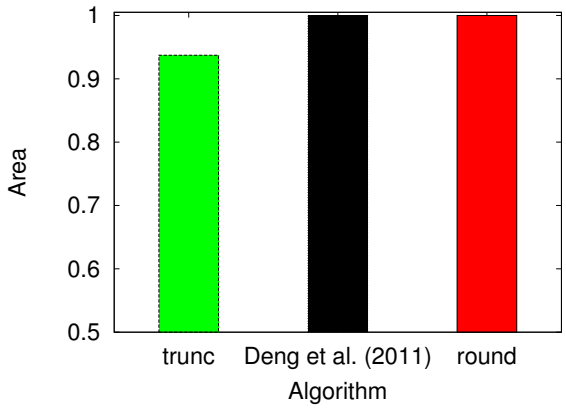


(f) Non-scalable Test Problems

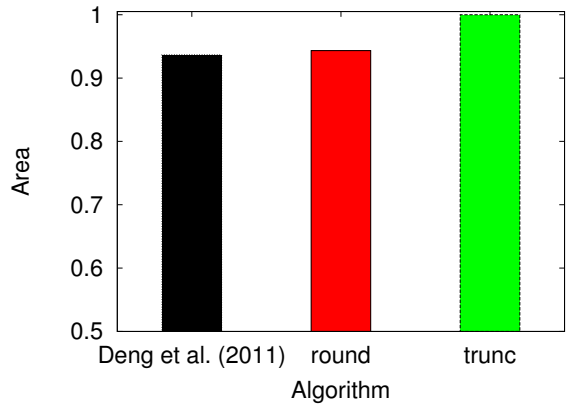
Figure 2: Performance Profile graphs based on the median values of the executions for the 24 test problems.



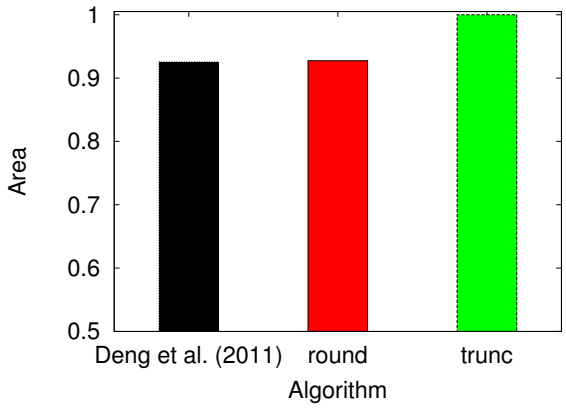
(a) Scalable Test Problems with 10 dimensions



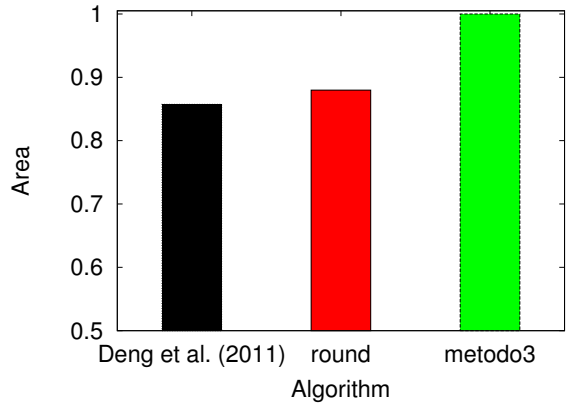
(b) Scalable Test Problems with 25 dimensions



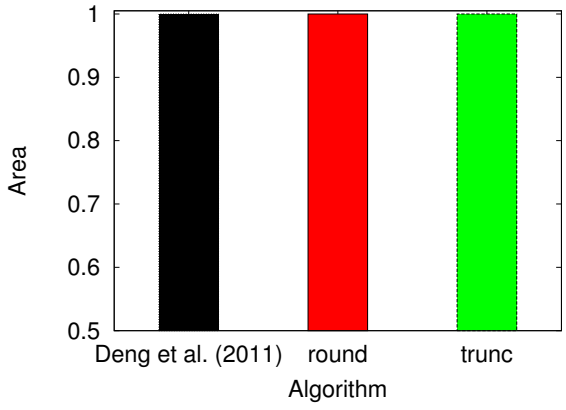
(c) Scalable Test Problems with 50 dimensions



(d) Scalable Test Problems with 75 dimensions

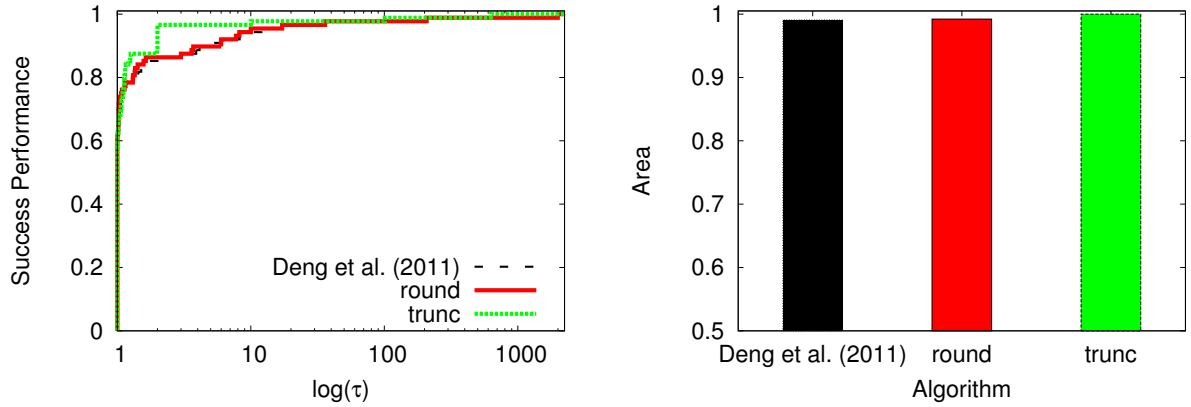


(e) Scalable Test Problems with 100 dimensions



(f) Non-scalable Test Problems

Figure 3: Area under the curve in the Performance Profile graphs.



(a) Performance Profile graph

(b) Area under the curve

Figure 4: Performance Profile results considering, simultaneously, all test problems (scalable and non-scalable) with a different number of dimensions (10-D, 25-D, 50-D, 75-D and 100-D) for the case of scalable functions.

in all, but the 25-D case.

Figure 4 depicts the performance profile results obtained when considering all test problems (scalable and non-scalable) with a different number of dimensions (10-D, 25-D, 50-D, 75-D and 100-D) for the case of scalable functions. The analysis of the results suggest that the truncation procedure, which obtains a slight advantage, may be considered the most robust mapping technique between those selected for this work.

The results obtained indicate that the technique employed in the conversion between real and integer values influences the quality of solutions for the set of problems considered in this work. Besides that, the increase in the dimensionality of the test problems impacts the effectiveness of each mapping technique. This fact can be observed specially for the procedure described in (Deng et al., 2011), which has a performance decline as the dimensionality of the problems is increased, and might suggest that more complex techniques are not suitable for higher dimensional problems.

7. CONCLUSIONS

This paper presented some popular techniques found in the literature concerning variants of the original differential evolution algorithm aiming at its application to problems with integer variables. A comparative analysis between three different mapping techniques was conducted in order to assess their relative performances when evaluated over a set of twenty-four unconstrained integer programming test problems.

The results obtained indicate the competitiveness of the mapping technique proposed in (Deng et al., 2011) -where real values confined in a narrow range $[0, 1]$ are converted to integer values through a mapping operator- only when applied to low dimensionality (10-D and 25-D) problems. The rounding procedure exhibits an intermediate performance in most cases, except for 25-D problems, where it achieves the best results. For higher dimensional problems (50-D, 75-D and 100-D), the best performance is obtained by the truncation technique. Also, when considering the Performance Profiles including all test functions (scalable with a different

number of parameters and non-scalable), it becomes clear that the truncation procedure is more robust than the other mapping techniques considered.

Some perspectives for future work involve experiments concerning the test problems employed in this paper with an even higher number of decision variables (i.e. $D > 100$) besides the use of additional sets of test functions, including real-world optimization problems. Moreover, the development of more effective and robust mapping techniques appears as an attractive alternative.

APPENDIX

This appendix presents the test functions employed in the computational experiments of this work. A separation between non-scalable and scalable test functions was performed in order to make it easier to assess and visualize the results.

Non-scalable Test Problems

1) Test Problem 1 (Glankwahmdee et al., 1979):

$$\min f_1(x) = -(15\ 27\ 36\ 18\ 12)x + x^T \begin{vmatrix} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & -32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{vmatrix} x$$

$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2, 3, 4, 5$$

with best known solutions $x^* = (0, 11, 22, 16, 6)$ and $x^* = (0, 12, 23, 17, 6)$

$$\text{with } f_1(x^*) = -737$$

2) Test Problem 2 (Himmelblau's function) (Glankwahmdee et al., 1979) :

$$\min f_2(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2$$

with global minimum solution $x^* = (3, 2)$ and $f_2(x^*) = 0$.

3) Test Problem 3 (Glankwahmdee et al., 1979):

$$\min f_3(x) = (9x_1^2 + 2x_2^2 - 11)^2 + (3x_1 + 4x_2^2 - 7)^2$$

$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2$$

with global minimum solution $x^* = (1, 1)$ and $f_3(x^*) = 0$.

4) Test Problem 4 (2D Rosenbrock function) (Glankwahmdee et al., 1979):

$$\min f_4(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$
$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2$$
$$\text{with global minimum solution } x^* = (1, 1) \text{ and } f_4(x^*) = 0.$$

5) Test Problem 5 (Glankwahmdee et al., 1979):

$$\min f_5(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$
$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2, 3, 4$$
$$\text{with global minimum solution } x^* = (0, 0, 0, 0) \text{ and } f_5(x^*) = 0.$$

6) Test Problem 6 (Colville function) (Ng et al., 2005):

$$\min f_6(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 +$$
$$(1 - x_3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$$
$$\text{such that } -10 \leq x_i \leq 10,\ x_i \in \mathbb{Z},\ i = 1, 2, 3, 4$$
$$\text{with global minimum solution } x^* = (1, 1, 1, 1) \text{ and } f_6(x^*) = 0.$$

7) Test Problem 7 (Vegh et al., 2011):

$$\min f_7(x) = 737 - (15\ 27\ 36\ 18\ 12)(x - e_1)$$
$$+(x - e_1)^T \left| \begin{array}{ccccc} 35 & -20 & -10 & 32 & -10 \\ -20 & 40 & -6 & -31 & 32 \\ -10 & -6 & 11 & -6 & -10 \\ 32 & -31 & -6 & 38 & -20 \\ -10 & 32 & -10 & -20 & 31 \end{array} \right| (x - e_1)$$
$$\text{where } e_1 = (1, 0, 0, 0, 0) \text{such that } -30 \leq x_i \leq 30,\ x_i \in \mathbb{Z},\ i = 1, 2, 3, 4, 5$$
$$\text{with global minimum solution } x^* = (1,11,22,16, 6) \text{ and } (1,12,23,17,6)$$
$$\text{with } f_7(x^*) = 0.$$

8) Test Problem 8 (Vegh et al., 2011):

$$\min f_8(x) = (x_1 + 10x_2 - 28)^2 + 5(x_3 - x_4 + 12)^2 + (x_2 - 2x_3 + 8)^2 +$$
$$10(x_1 - x_4 + 9)^2$$
$$\text{such that } -20 \leq x_i \leq 20,\ x_i \in \mathbb{Z},\ i = 1, 2, 3, 4$$
$$\text{with global minimum solution } x^* = (8, 2, 5, 17) \text{ and } f_8(x^*) = 0.$$

Scalable Test Problems

9)Test Problem 9 (Günter, 1994):

$$\min f_9(x) = \sum_{i=1}^{n} |x_i|$$

$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0,\ i = 1, 2, ..., n \text{ and } f_9(x^*) = 0.$$

10) Test Problem 10 (Sphere function) (Günter, 1994):

$$\min f_{10}(x) = \sum_{i=1}^{n} x_i^2$$

$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0,\ i = 1, 2, ..., n \text{ and } f_{10}(x^*) = 0.$$

11) Test Problem 11 (Ng et al., 2005):

$$\min f_{11}(x) = (x_1 - 1)^2 + (x_n - 1)^2 + n \sum_{i=1}^{n-1} (n - i)(x_i^2 - x_{i+1})^2$$

$$\text{such that } -5 \leq x_i \leq 5,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 1,\ i = 1, 2, ..., n \text{ and } f_{11}(x^*) = 0.$$

12) Test Problem 12 (Rosenbrock function) (Ng et al., 2005):

$$\min f_{12}(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

$$\text{such that } -5 \leq x_i \leq 5, x_i \in \mathbb{Z}, i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 1,\ i = 1, 2, ..., n \text{ and } f_{12}(x^*) = 0.$$

13) Test Problem 13 (Ng et al., 2005):

$$\min f_{13}(x) = \sum_{i=1}^{n} x_i^4 + (\sum_{i=1}^{n} x_i)^2$$

$$\text{such that } -5 \leq x_i \leq 5,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0,\ i = 1, 2, ..., n \text{ and } f_{13}(x^*) = 0.$$

14) Test Problem 14 (Ackley function) (Zhu & Fan, 2009):

$$\min f_{14}(x) = -20exp(-0.2\sqrt{n^{-1} \sum_{i=1}^{n} x_i^2}) - exp(n^{-1} \sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$$

$$\text{such that } -30 \leq x_i \leq 30,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0,\ i = 1, 2, ..., n \text{ and } f_{14}(x^*) = 0.$$

15) Test Problem 15 (Griewank function) (Zhu & Fan, 2009):

$$\min f(x) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$$

$$\text{such that } -600 \leq x_i \leq 600,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0,\ i = 1, 2, ..., n \text{ and } f_{15}(x^*) = 0.$$

16) Test Problem 16 (Rastrigin function) (Zhu & Fan, 2009):

$$\min f_{16}(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i)]$$

$$\text{such that } -5 \leq x_i \leq 5,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0, i = 1, 2, ..., n \text{ and } f_{16}(x^*) = 0.$$

17) Test Problem 17 (Mohan & Nguyen, 1999):

$$\min f_{17}(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} \frac{i}{2} x_i)^6$$

$$\text{such that } -100 \leq x_i \leq 100,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 0,\ i = 1, 2, ..., n \text{ and } f_{17}(x^*) = 0.$$

18) Test Problem 18 (Mohan & Nguyen, 1999):

$$\min f_{18}(x) = 1 - exp[(-\tfrac{1}{60}) \sum_{i=1}^{30} x_i^2]$$

$$\text{such that } 0 \leq x_i \leq 5,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x^* = 0,\ i = 1, 2, ..., n \text{ and } f_{18}(x^*) = 0.$$

19) Test Problem 19 (Modified Sphere function) (Vegh et al., 2011):

$$\min f_{19}(x) = \sum_{i=1}^{n} (x_i - i)^2$$

$$\text{such that } -n \leq x_i \leq n,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with gloal minimum solution } x_i^* = i,\ i = 1, 2, ..., n \text{ and } f_{19}(x^*) = 0.$$

20) Test Problem 20 (Vegh et al., 2011):

$$\min f_{20}(x) = \sum_{i=1}^{n} |x_i - 2i|$$

$$\text{such that } -2n \leq x_i \leq 2n,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 2i,\ i = 1, 2, ..., n \text{ and } f_{20}(x^*) = 0.$$

21) Test Problem 21 (Vegh et al., 2011):

$$\min f_{21}(x) = \sum_{i=1}^{n} i^2 (x_i - i)^2$$

$$\text{such that } -n \leq x_i \leq n,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = i,\ i = 1, 2, ..., n \text{ and } f_{21}(x^*) = 0.$$

22) Test Problem 22 (Modified Rastrigin function) (Vegh et al., 2011):

$$\min f_{22}(x) = 10n + \sum_{i=1}^{n} [(x_i - i)^2 - 10cos(2\pi(x_i - i))]$$

$$\text{such that } -n \leq x_i \leq n,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = i,\ i = 1, 2, ..., n \text{ and } f_{22}(x^*) = 0.$$

23) Test Problem 23 (Modified Griewank function) (Vegh et al., 2011):

$$\min f_{23}(x) = 1 + \sum_{i=1}^{n} \frac{(x_i - 3i)^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i - 3i}{\sqrt{3i}})$$

$$\text{such that } -3n \leq x_i \leq 3n,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 3i,\ i = 1, 2, ..., n \text{ and } f_{23}(x^*) = 0.$$

24) Test Problem 24 (Modified Ackley function) (Vegh et al., 2011):

$$\min f_{24}(x) = -20exp(-0.2\sqrt{n^{-1} \sum_{i=1}^{n} (x_i - 4(n - i + 1)^2)}) -$$

$$exp(n^{-1} \sum_{i=1}^{n} cos(x_i - 4(n - i + 1))) + 20 + e$$

$$\text{such that } 0 \leq x_i \leq 4n,\ x_i \in \mathbb{Z},\ i = 1, 2, ..., n$$

$$\text{with global minimum solution } x_i^* = 4i,\ i = n, ..., 2, 1 \text{ and } f_{24}(x^*) = 0.$$

Acknowledgements

The authors acknowledge the support from FAPERJ and CNPq (grant 310778/2013-1).

REFERENCES

Barbosa, H. J. C., Bernardino, H. S.,& Barreto, A. M. S., 2010. Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In IEEE Congress on Evolutionary Computation, pp. 1-8.

Das, S. & Suganthan, P. N., 2011. Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation, vol. 15, n. 1, pp. 4-31.

Datta, D. & Figueira, J. R., 2013. A real–integer–discrete-coded Differential Evolution. Aplied Soft Computing, vol. 13, n. 9, pp. 3884-3893.

Deng, C., Liang, C., Zhao, B., Yang, Y., & Deng, A., 2011. Structure-Encoding Differential Evolution for Integer Programming. JSW, vol. 6, n. 1, pp. 140-147.

Dolan, E.D. & Moré, J.J., 2002. Benchmarking optimization software with performance profiles. Mathematical Programming In Mathematical Programming, vol. 91, n. 2, pp. 201-213.

Dong, Y., Guo, Q., & Tang, L., 2013. A Pointer-based Discrete Differential Evolution. In IEEE Congress on Evolutionary Computation, pp. 3064-3071.

Glankwahmdee, A., Liebman, J. S., & Hogg, G. L., 1979. Unconstrained discrete nonlinear programming. Engineering Optimization, vol. 4, n. 2, pp. 95-107.

Günter, R., 1994. An evolutionary algorithm for integer programming. In Davidor, Y., Schwefel, H., & Männer, R., eds, Parallel Problem Solving from Nature — PPSN III, vol. 866, pp. 139-148. Springer Berlin Heidelberg.

Lampinen, J.& Zelinka, I. 1999. Mixed integer-discrete-continuous optimization by Differential Evolution-Part 1: the optimization method. Czech Republic. Brno Univ. of Tech., pp. 77-81.

Li, H. & Zhang, L., 2014. A Discrete Hybrid Differential Evolution Algorithm for Solving Integer Programming Problems. Engineering Optimization, vol. 46, n. 9, pp. 1238-1268.

Liu, Y. L., Chen, W., Zhan, Z., Lin, Y., Gon, Y., & Zhang, J., 2013. A Set-Based Discrete Differential Evolution Algorithm. In 2013 IEEE International Conference on Systems, Man, and Cybernetics, pp.1347-1352.

Maravilha, A. L., Ramírez, J. A., & Campelo, F., 2013. A New Algorithm Based on Differential Evolution for Combinatorial Optimization. In 2013 BRICS Congress on 1st Computational BRICS Countries Intelligence Congress & on 11th Computational Brazilian Congress Intelligence on Computational Intelligence, pp. 60-66.

Mohan, C. & Nguyen, H. T., 1999. A Controlled Random Search Technique Incorporating the Simulated Annealing Concept for Solving Integer and Mixed Integer Global Optimization Problems. Comput. Optim. Appl., vol. 14, n. 1, pp. 103-132.

Ng, C., Zhang, L., Duan, L., & Tian, W., 2005. Discrete Filled Function Method for Discrete Global Optimization. Computational Optimization and Applications, vol. 31, n. 1, pp. 87-115.

Onwubolu, G. & Davendra, D., 2006. Scheduling flow shops using Differential Evolution algorithm. European Journal of Operational Research, vol. 171, n. 2, pp. 674-692.

Prado, R. S., Silva, R. C. P., Guimarães, F. G., & Neto, O. M., 2010. Using Differential Evolution for Combinatorial Optimization: A General Approach. In SMC, pp.11-18.

Price, K., Storn, R., & Lampinen, J. A., 2005. Differential Evolution: A Practical Approach to Global Optimization. Springer Science & Business Media.

Storn, R. & Price, K., 1995. Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. TR-95-012, ICSI.

Talbi, E., 2009. Metaheuristics: From Design to Implementation. Wiley Publishing.

Vegh, V., Pierens, G. K., & Tieng, Q. M., 2011. A variant of Differential Evolution for discrete optimization problems requiring mutually distinct variables. International Journal of Innovative Computing, Information and Control, vol. 7, n. 2, pp. 897-914.

Zhu, W. & Fan, H., 2009. A discrete dynamic convexized method for nonlinear integer programming. Journal of Computational and Applied Mathematics, vol. 223, n. 1, pp. 356-373.