ISSN 2007-9737

Self-Adaptive Differential Evolution Hyper-Heuristic with Applications in Process Design

Hernán Peraza-Vázquez[1], Aidé M. Torres-Huerta[1], Abelardo Flores-Vela[2]

[1] Instituto Politécnico Nacional, CICATA-Tamaulipas, Mexico

[2] Instituto Politécnico Nacional, CMP, Mexico

{hperaza, atorresh, afloresv}@ipn.mx

Abstract. The paper presents a differential evolution (DE)-based hyper-heuristic algorithm suitable for the optimization of mixed-integer non-linear programming (MINLP) problems. The hyper-heuristic framework includes self-adaptive parameters, an ε-constrained method for handling constraints, and 18 DE variants as low-level heuristics. Using the proposed approach, we solved a set of classical test problems on process synthesis and design and compared the results with those of several state-of-the-art evolutionary algorithms. To verify the consistency of the proposed approach, the above-mentioned comparison was made with respect to the percentage of convergences to the global optimum (NRC) and the average number of objective function evaluations (NFE) over several trials. Thus, we found that the proposed methodology significantly improves performance in terms of NRC and NFE.

Keywords. Processes synthesis, mixed-integer non-linear programming (MINLP) problems, differential evolution (DE), hyper-heuristics.

1 Introduction

A constrained optimization problem is usually written as a non-linear programming problem (NLP) [34] of the following form:

$$\begin{aligned} &\text{Minimize } f(X), \\ &\text{Subject to: } g_i(X) \le 0, \quad i = 1,..,p, \\ &\qquad\qquad h_j(X) = 0, \quad j = p+1,..,m, \\ &\qquad\qquad x_k^{(L)} \le x_k \le x_k^{(U)}, \quad k = 1,..,D. \end{aligned} \tag{1}$$

In the above NLP problem, the function f is the objective function, where $f(X): R^D \to R$ there are D variables, $X = (x_1,..,x_D)$ is a vector of size D, $X \in R^D$, where R^D represents the entire search space, g_i are the inequality constraints, h_j are the equality constraints, and $x_K^{(L)}$, $x_K^{(U)}$ are the lower-bound constraints and upper-bound constraints, respectively. Further, p is the number of inequality constraints and $m - p$ is the number of equality constraints.

The equality constraints can be transformed into the inequality form, and then, they can be combined with the other inequality constraints as follows:

$$G_i(X) = \begin{cases} \max\{g_i(X), 0\} & i = 1,..,p, \\ |h_j(X)| & i = p+1,..,m. \end{cases} \tag{2}$$

Thus, the optimization goal is to find a feasible vector X to minimize the objective function.

When the vector X contains a subset of μ and ν vectors of continuous real variables and integer variables, respectively, $|\mu| + |v| = |X| = D$, the NLP problem becomes a mixed-integer non-linear programming problem (MINLP).

Non-convex NLPs and MINLPs are commonly found in real-world situations. Therefore, the scientific community continues to develop new approaches for obtaining optimal solutions with acceptable computational time in various engineering and industrial fields. For example, in design optimization, the design objective could be simply to minimize the cost or maximize the

efficiency of production; on the other hand, the objective could be more complex, e.g., controlling the highly non-linear behavior of pH neutralization processes in a chemical plant. The need to solve practical NLP/MINLP problems has led to the development of a large number of heuristics and metaheuristics over the last two decades [27,33,36]. Metaheuristics, which are emerging as effective alternatives for solving NP-hard optimization problems, are strategies for designing or improving very general heuristic procedures with high performance in order to find (near-)optimal solutions; the goal is efficient exploration (diversification) and exploitation (intensification) of the search space. For example, we can take advantage of the search experience to guide search engines by applying learning strategies or incorporating probabilistic decisions.

Strategies such as differential evolution (DE), ant and bee algorithms, particle swarm optimization (PSO), and cuckoo search have been effectively applied to many research areas, including process design [2,3]. Nevertheless, these approaches have a drawback in that they require the setting of several parameters and components, e.g., population size, number of generations, recombination probability, mutation operator, and selection function, as well as the handling of constraints. Therefore, selecting the best combination of these parameters/components leads to complexity of the metaheuristic algorithms. In other words, the various possible combinations of the parameters drastically affect the performance of the algorithms.

Nowadays, methodologies such as hyper-heuristics minimize human interference in the tuning and design of heuristics or metaheuristics adapted for solving a problem in a particular domain [3]. In this paper, an approach for solving non-convex MINLP problems is presented. Our analysis is based on a DE hyper-heuristic methodology, which is able to choose from among 18 DE models for low-level heuristics and tune the most important parameters through a self-adaptation mechanism.

The remainder of this paper is organized as follows. Section 2 describes the DE algorithm. Section 3 outlines the hyper-heuristic algorithm. Section 4 reviews some related studies. Section 5 describes the proposed approach. Section 6 presents an illustrative example to show how a population evolves through generations before reaching the global optimum. Section 7 describes a set of problems on process synthesis and design for an experimental setup to show the applicability and efficiency of our approach in the case of non-convex MINLP problems. Section 8 presents the corresponding results. Finally, Section 9 summarizes our findings and concludes the paper with a brief discussion on the scope for future work.

2 Differential Evolution (DE) Algorithm

Since its implementation in 1995 by Storn and Price [37,38], DE has gained wide acceptance because it is particularly easy to work with, having only a few control variables that remain fixed throughout the entire optimization procedure. DE is a search method that uses a set of vectors $x_{i,G}$ as the population in each generation. The algorithm starts from a randomly generated initial population until a satisfactory one is obtained. The population size does not change during the evolutionary process; thus, the algorithm is a population-based stochastic search technique classified as floating-point encoded. A DE pseudo-code is shown in Figure 1.

The main concept underlying DE is a new schema to generate vectors. The mechanism is as follows. A new vector is generated by adding the weighted difference between two member vectors of the population to a third member (see Figure 1 and focus on line 11).

Both vectors, i.e., the newly generated vector and the original vector, are rated by an evaluation method. The vector with the best fitness is chosen, and it replaces the losing vector in this comparison (see Figure 1: the If statement in line 16 and the corresponding $Else$ starting in line 18). There are several variants of the mutation scheme of DE. The notation used in the literature is $DE/\phi/\varphi/\psi$, where ϕ denotes the base vector to be disturbed, i.e., the mechanism for constructing the mutant vector, φ denotes the number of pairs of vectors to be disturbed, and ψ denotes the crossover type (exp: exponential; bin: binomial). Several DE schemes are presented in Appendix A.

Line 11 contains the mutation operator, where $r_1 \neq r_2 \neq r_3$ are randomly generated vectors $\in [1, NP]$ and NP is the population size. F is a scaling factor that typically $\in (0,1]$; it controls the amplification of the difference vector. Lines 10 to 14 contain the crossover operator, which is represented by the $If-Else$ statement, where the crossover constant is denoted by CR $\in [0,1]$ and j_{rand} is a randomly chosen index $\in \{1,2,..,D\}$; D is the number of variables in the problem. CR and F are user-defined parameters. CR is highly sensitive to the property and complexity of the problem, while F is related to the convergence speed. The DE model described above is known as DE/rand/1/bin, where rand denotes the base vector to be disturbed, 1 denotes the number of pairs of vectors to be disturbed, and bin denotes the recombination adopted.

```
1  Begin
2    G=0
3    Create a random initial population X_{i,G}, ∀i, i = 1,..,NP
4    Evaluate f(X_{i,G})
5    For G = 1 to MAX_GENERATIONS Do
6      For i = 1 to NP Do
7        Select randomly r1 ≠ r2 ≠ r3 :
8        j_rand = randind(1, D)
9        For j = 1 to D Do
10         If (rand_j[0, 1) < CR or j = j_rand) Then
11           V^i_{j,G} = X_{r1,G} + F * (X_{r2,G} − X_{r3,G})
12         Else
13           V^i_{j,G} = X^i_{j,G}
14         End If
15       End For
16       If f(V_{i,G+1}) ≤ f(X_{i,G}) Then
17         X_{i,G+1} = V_{i,G+1}
18       Else
19         X_{i,G+1} = X_{i,G}
20       End If
21     End For
22     G= G+1
22   End For
23 End
```

Fig. 1. Differential evolution algorithm

3 Hyper-Heuristics

A hyper-heuristic is a search method or learning mechanism for selecting or generating simpler heuristics to solve computational search problems. The hyper-heuristic framework consists of two main parts: a high-level methodology and a number of low-level heuristics. Given a particular problem instance or class of instances, the high-level method provides the means to exploit the strength of multiple low-level heuristics, where each heuristic can be useful at different stages of the search. The solution is either accepted or rejected based on an acceptance criterion. The heuristic selection and acceptance methods are the most important components of a hyper-heuristic. The main feature of the hyper-heuristic approach is that the high-level heuristic performs a search over the space of the low-level heuristics rather than a direct solution space. A domain barrier between the levels prevents any problem-specific information from being passed to the hyper-heuristic level, thereby allowing for selection from among the low-level heuristics without the need for domain knowledge. The development of hyper-heuristics is mainly motivated by the need for algorithms that are more generally applicable than most current implementations of search methods. The low-level heuristics can be designed in advance or created simultaneously during runtime from a set of potential components. Thus, the hyper-heuristic approach aims to reuse the heuristics over unseen instances and raise the level of generality at which an optimization system can operate.

In the hyper-heuristic framework, the high-level heuristic has no knowledge of the problem-domain concealed in the low-level one. In turn, the low-level heuristic is not aware of the learning mechanism used to choose its heuristic (DE models) in the high level. This process introduces the concept of plug-and-play of heuristics (DE models).

In recent years, hyper-heuristics have been employed in several applications such as the bin packing problem [29], 2D strip packing [8], production scheduling [30], constraint satisfaction problems [28], and the vehicle routing problem [25]. In addition, some hyper-heuristics use a metaheuristic as a high-level methodology or mechanism to select or generate low-level

heuristics, with effective and encouraging results. A survey of hyper-heuristics can be found in [7].

4 Related Work

In the existing literature, it is possible to find several DE-based approaches for solving constrained optimization problems.

In [23], Lampinen proposed an extension of DE. The method consists of a modification to the selection operator with a new selection criterion for handling the constraint functions. The selection is based on Pareto dominance in the effective constraint function space, and the approach does not introduce any extra search parameters to be set by the user. A DE/rand/1/bin strategy was used.

In [21], a penalty function is designed to handle the constraints and a co-evolution model is incorporated into a DE algorithm to perform evolutionary search in spaces of solutions and penalty factors. Both evolve interactively and self-adaptively; thus, a satisfactory solution and suitable penalty factors can be obtained simultaneously. A DE/best-rand/1/bin strategy was used.

The aim of the approach proposed by Mezura et al. in [26] is to increase the probability that each parent generate a better offspring by allowing each generation to generate more than one offspring using a different mutation operator that uses information of the best solution and the current parent to find new search directions. A DE/rand/1/bin strategy was used.

In [24], Mallipeddi et al. showed how a compendium of constraint-handling techniques used with evolutionary algorithms can be effectively applied to differential evolution. These include the superiority of feasible solutions, self-adaptive penalty, ε-constraint, and stochastic ranking. The authors showed that the effectiveness of conventional DE in solving a numerical optimization problem depends on the selected mutation strategy and its associated parameter values. Thus, different optimization problems require different mutation strategies with different parameter values. The DE/rand/2/bin and DE/current-to-rand/1/bin strategies were used.

A DE variant considered as a state-of-the-art algorithm, namely, SaDE [32], incorporates a learning strategy in the mutation phase, which probabilistically selects one out of two available learning strategies, DE/rand/1/bin or DE/current-to-best/2/bin, and applies it to the current population. Furthermore, the control parameter CR is self-adapted based on the previous learning experience, and a quasi-Newton method is used as a local search method.

In [31], the SaDE algorithm was compared with several parameter-adaptive DE variants. It was found that the SaDE algorithm could evolve suitable strategies and parameter values as the evolution progressed and that the learning period parameter had an insignificant impact on the performance. In addition, the algorithm was more effective in obtaining high-quality solutions over a suite of 26 bound-constrained numerical optimization problems.

In [20], the original *search method* in the SaDE algorithm was substituted by a sequential quadratic programming method.

A comparison between a neighborhood search strategy and the SADE algorithm can be found in [45]. This strategy affects the F parameter, which is related to the convergence speed. Thus, it is effective in escaping from local optima when searching environments without prior knowledge about what kind of search step size is preferred. A hybridization of SaDE with the neighborhood search algorithm led to the following approach: the neighborhood search strategy, the learning strategy in the mutation phase, and three self-adaptive mechanisms for the three parameters, namely, the scale factor F, the crossover rate CR, and the mutation strategy. The authors reported that such a hybridization is significantly superior to both neighborhood search algorithm and SaDE individually.

In [4], a self-adaptive mechanism for changing two DE control parameters, F and CR, during the optimization process with a small and varying population size was presented. A DE/rand/1/bin strategy was used.

In [42], a success-history-based adaptive DE was proposed. The strategy uses historical memory in order to adapt the control parameters F and CR.

The use of eigenvectors of the covariance matrix of individual solutions, which makes the crossover rotationally invariant in DE, was

Table 1. Design comparison of DE-HH & SADE

	SADE	DE-HH
Mutation Operation	Max 4 Strategies: DE/rand/1 De/current to best/2 DE/rand/2 DE/current-to-rand/1	Max 9 Strategies over a Hyper-heuristic framework. See Appendix B.
Crossover Operation	Binomial	Binomial or Exponential (Self-Adaptive)
Selection Operation	The fitness value of each trial vector is compared to that of its corresponding target vector in the current population. (traditional DE, Figure 1)	ε-level comparison defined by a lexicographical order, eqs.11 and 12.
Handling Constraints	Method based on superiority of feasible solutions.	Epsilon-Constrained Method
Parameter Adaptation	**NP**: User-specified value **CR:** Self-adaptive (learning experience) **F:** random values in the range (0, 2] normal distribution of mean 0.5 and standard deviation 0.3	**NP**: User-specified value **CR:** Self-adaptive (learning experience) **F:** Self-adapted according to eq.6
Local Search Method	Sequential Quadratic Programming (SQP)	Neighborhood Search

proposed by Guo et.al in [18]. The incorporation of eigenvector-based crossover in six state-of-the art DE variants showed either solid performance gains or statistically identical behavior. The concept of opposition-based learning has been applied to improve the performance of metaheuristic algorithms and machine-learning algorithms. The method tries to find a better candidate solution by simultaneously considering an estimate point and its corresponding opposite estimate.

In [19], a partial opposition-based learning methodology was applied to an adaptive DE algorithm.

In [6], an adaptive DE based on competition among several strategies was used. The approach uses a rotation-invariant current-to-best mutation in the algorithm. The aim is to increase the efficiency of DE on rotated or composite functions.

In [43], a hyper-heuristic based on DE was proposed. The approach consists of two phases. The first phase is responsible for selecting the type of recombination to be adopted (either bin or exp). At the beginning of the search process, a training stage based on the maximum number of generations and a random descent selection mechanism is required to initialize the expected values for each of the DE variants. The second phase is responsible for selecting the specific model to be applied for generating the next generation. Random selection and roulette wheel selection mechanisms are used. Stochastic ranking is incorporated for handling the constraints. Twelve crossover model strategies are used as low-level heuristics in the hyper-heuristic framework.

Further details about recent research on hyper-heuristics based on DE can be found in [16].

In spite of the above-mentioned efforts, there remains a considerable scope for improving DE performance, e.g., by using strategies or proposing new strategies of self-adaptation for parameter control, mutation, or constraint handling, or by applying learning mechanisms that have not been used previously in DE frameworks.

5 Proposed Approach

The motivation of our approach is to solve non-convex MINLP problems with applications in process design by using a DE-based hyper-heuristic algorithm. Our framework includes self-adaptive parameters, an **ε**-constrained method for handling constraints, **9** mutation model strategies, and a binomial and exponential crossover model; the combination mutation-crossover allows to have

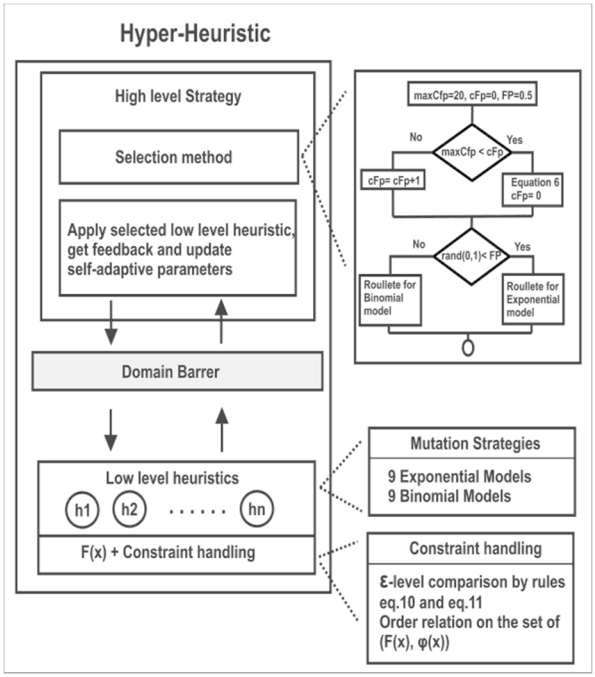

Fig. 2. Overall flow of our approach

a maximum of 18 Differential Evolution models. The overall flow of our approach is shown in Figure 2 and the pseudo-code is shown in Figure 3.

Our framework includes the strategies described in subsections 5.1 to 5.5.

5.1 Self-Adaptive Parameters

The use of the DE self-adaptive mechanism to make a DE solver more robust and efficient is reported in [5]; in addition, its advantages and disadvantages are discussed there.

Our self-adaptation scheme focuses on the three main components of the DE algorithm that directly affect the performance and the quality of the solution, namely, the mutation, the crossover, and the scale factor F. Lines 9 to 15 in Figure 1 show the use of these components without a self-adaptation mechanism. In order to achieve auto-tuning of these parameters, a ***learning period*** was incorporated. The basic idea is to define a specified number of generations to collect data and a counter of iterations. When the counter exceeds the number of generations proposed, it will be reset once the variable is updated with a new value.

5.2 Self-Adaptation of Crossover Rate CR

Our crossover strategy is based on SaDE [32,45], where the CRm variable is set to 0.5 initially; after a determined number of generations, CRm will be updated according to equation 3. Thus, CRm is used as the mean value in the Gaussian function given by equation 4 in order to compute the crossover rate (CR) that will be used in the recombination method:

$$CRm = \frac{1}{|CR_{rec}|}\sum_{K=1}^{CR_{rec}} CR_{rec}(k), \tag{3}$$

$$CR_i = N_i(CR_m, 0.1). \tag{4}$$

The proposed crossover includes a strategy of selection to choose between a binomial or exponential method, in contrast to the SaDE algorithm that only uses binomial crossover.

5.3 Self-Adaptation of the Scale Factor F

Because the scale factor F is related to the convergence speed, its self-adaptation strategy incorporates a move-generation mechanism as a neighborhood search operator in the DE algorithm. This can be observed in [45, 46], where the scale factor F is replaced by equation 5:

$$F_i = \begin{cases} N_i(0.5, 0.3), & if\ U_i(0,1) < Fp, \\ \partial_i, & otherwise, \end{cases} \tag{5}$$

where $N_i(0.5, 0.3)$ denotes a Gaussian random number with mean 0.5 and standard deviation 0.3, and ∂_i denotes a Cauchy random variable with scale parameter t=1. In our approach, Fp in equation 5 will be self-adapted according to equation 6:

$$Fp = \frac{\sum TSGRN}{\sum (TSGRN + CRN)}, \tag{6}$$

```
1  Begin
2    G=0 //initialization
3    CR_cont = 0, cFg = 0, CR_max = 20, maxContCrSel = 20, cCrSel = 20
4    maxContcFg = 20, CRm = 0.5, CrSel = 0.5, Fp = 0.5, K ∈ [0.3, 0.9]
5    Create a random initial population X_{i,G}, ∀i, i = 1,..,NP
6    For G = 1 to MAX_GENERATIONS Do
7      Evaluate f(X_{i,G})
8      Select X_best in generation G
9      CR = N(CRm, 0.1)
10     CR_rec[CR_cont] = CR
11     If (CR_max < CR_cont) Then
12       CR_m = 1/|CR_rec| Σ_{K=1}^{|CR_rec|} CR_rec(k)
13       reset to zero CR_cont
14     Else
15       CR_cont = CR_cont + 1
16     End If
17     If (maxContCrSel < cCrSel) Then
18       CrSel = Σ Total success Variant Exponential recombination / Σ Total success Variants Exponential & Binomial recombination
19       Update probability percentages for roulette selection
20       reset to zero cCrSel
21     Else
22       cCrSel= cCrSel+1
23     End If
24     If (rand(0, 1) < CrSel) Then
25       Select one of the nine exponential recombinations by roulette wheel method
26     Else
27       Select one of the nine binomial recombinations by roulette wheel method
28     End If
29     If (maxContcFg < cFp) Then
30       Fp = Σ Total success Gaussian random number / Σ Total success Gaussian random number & Cauchy random number
31     Else
32       cFp= cFp+1
33     End If
34     If (rand(0, 1) < Fp) Then
35       F = Gaussian random variable with mean 0.5 and standard deviation 0.3
36     Else
37       F = Cauchy random variable
38     End If
39     For i = 1 to NP Do
40       Select randomly r1 ≠ r2 ≠ r3 ≠ r4 ≠ r5
41       compute X_{r1,G}, X_{r2,G},..X_{r5,G}
42       If exponential recombination Then
43         Generate V^i_{j,G} by exponential recombination method, selected in line 25
44         (using K,CR,Fp, F, X_{r1,G}..X_{r5,G})
45       Else
46         j_rand = randind(1, D) Used in the binomial model
47         Generate V^i_{j,G} by binomial recombination method, selected in line 27
48         (using K,CR,Fp, F, X_{r1,G}..X_{r5,G})
49       End If
50       If (f(V_{i,G+1}), φ(V_{i,G+1})) <_ε (f(X_{i,G}), φ(X_{i,G})) Then
51         X_{i,G+1} = V_{i,G+1}
52       Else
53         X_{i,G+1} = X_{i,G}
54       End If
55     End For
56     G= G+1
57   End For
58 End
```

Fig. 3. Differential Evolution (DE)-based hyper-heuristic pseudo-code

where $TSGRN$ denotes the success Gaussian random number and CRN is the success Cauchy random number.

5.4 Self-Adaptation of Mutation Strategies

The impact of the various DE search operators on the exploration/exploitation of the search space is not the same. Certain mutation operators are more oriented toward exploitation, e.g., DE/best/1, whereas others are more oriented toward exploration, e.g., DE/rand/1 [13]. Thus, it can be difficult to choose the most efficient mutation operator, and a problem-dependent parameter may affect the performance of the algorithm. A determined combination of DE parameters can be suitable for one problem but unsuitable for another [44]. In order to raise the level of generality, our approach incorporates in the hyper-heuristic framework a selection method for choosing the type of recombination to be applied for generating the next population, either exponential or binomial, by a random process. The method includes a variable, CrSel, which is set to 0.5 initially; after a determined number of generations, CrSel will be self-adapted according to equation 7 (see Figure 3: lines 17 to 23).

This set of heuristics consists of nine promising mutation strategies reported in the literature, each for the binomial and exponential models; a roulette wheel method is used to choose the mutation variant to be adopted, see Figure 4.

The maximum number of possible crossover-mutation combinations are 18. These 18 DE-models are used as low-level heuristics shown in Appendix B. For example, if the type of recombination selected is exponential, the roulette wheel method will choose from among the nine mutation strategies for exponential recombination, starting with a probability of 1/9 for each strategy to be selected; the probabilities will be updated when CrSel is updated (see line 19 in Figure 3). The complete mutation strategies used in our approach are presented in Appendix B.

$$CrSel = \frac{\sum TSVER}{\sum (TSVER + BR)}, \tag{7}$$

where $TSVER$ denotes the success variant exponential recombination and BR is the success binomial recombination.

5.5 Constraint Handling

The ε-constrained method was proposed by Takahama [40]. It is based on the definition of a *constraint violation* $\phi(x)$ that is obtained from equation 8 or equation 9, which are adopted as a penalty in penalty function methods.

$$\phi(x) = \max\left\{\max_j\{0, g_j(x)\}, \max_j|h_j(x)|\right\} \tag{8}$$

$$\phi(x) = \sum_j \left\|\max\{g_i(x), 0\}\right\|^p + \sum_j \left\|h_j(x)\right\|^p \tag{9}$$

where $p \in Z^+$ and $\phi(x)$ is the maximum of all constraints or the sum of all constraints. Thus, $\phi(x)$ indicates by how much a search point x violates the constraints and the membership in the feasible region F. Feasible solutions exist in S, where $F \subseteq S$ and S is the search space. The values of $\phi(x)$ that can be obtained are given by equation 10.

$$\begin{cases} \phi(x) = 0 & (x \in F), \\ \phi(x) > 0 & (x \notin F), \end{cases} \tag{10}$$

An order relation on the set $(f(x), \phi(x))$ is known as an ε level comparison and defined by a lexicographic order in which $\phi(x)$ precedes $f(x)$, favoring the feasibility of *x* over the minimization of $f(x)$. The comparisons are made according to the rules given by equations 11 and 12:

$$(f(x_1), \phi(x_1)) <_\varepsilon (f(x_2), \phi(x_2)) \Leftrightarrow \begin{cases} f(x_1) < f(x_2), & if \quad \phi(x_1), \phi(x_2) < \varepsilon \\ f(x_1) < f(x_2), & if \quad \phi(x_1) = \phi(x_2) \\ \phi(x_1) < \phi(x_2) & otherwise \end{cases} \tag{11}$$

$$(f(x_1), \phi(x_1)) \leq_\varepsilon (f(x_2), \phi(x_2)) \Leftrightarrow \begin{cases} f(x_1) \leq f(x_2), & if \quad \phi(x_1), \phi(x_2) < \varepsilon \\ f(x_1) \leq f(x_2), & if \quad \phi(x_1) = \phi(x_2) \\ \phi(x_1) \leq \phi(x_2) & otherwise \end{cases} \tag{12}$$

The opposite cases where $\varepsilon = 0 (<_0 \; and \leq_0)$ and $\varepsilon = \infty (<_\infty \; and \leq_\infty)$ are equivalent to the

lexicographic order in which the constraint violation $\phi(x)$ precedes the function value $f(x)$ on the one hand and to the ordinal comparison < and ≤ between function values on the other hand, respectively. In order to obtain high-quality solutions, the **ε** level is statically controlled by equation 13. It is updated until the generation counter *k* reaches the control generation T_c. When the generation counter exceeds T_c, the **ε** level is reset to zero in order to obtain solutions without constraint violation. Note that *cp* is a user-defined parameter for controlling the reduction speed of the **ε** tolerance.

$$\varepsilon_s(0) = \phi(x_\theta),$$
$$\varepsilon_s(k) = \begin{cases} \varepsilon_s(0)(1-\frac{k}{T_c})^{cp}, & 0 < k < T_c, \\ 0, & k \geq T_c. \end{cases} \quad (13)$$

The **ε**-constrained method with static control was incorporated into our approach. We assume that p=1 in equation 9 for a simple sum of constraints.

For the function evaluation, in order to handle integer variables, real values are converted into integer values by truncation. The handling of binary variables is given by equation 14.

$$y_i = \begin{cases} 0, if & x_i \leq 0.5 \\ 1, & otherwise, \end{cases} \quad (14)$$

where x_i is a continuous variable, $0 \leq x_i \leq 1$. For the boundary constraint, the same handling mechanism that is used for continuous variables is applied (0 is assigned to the lower bound and 1 is assigned to the upper bound).

5.6 Our Approach versus SaDe: A Comparison between Designs

The SaDE algorithm uses only a binomial crossover operator and 2 mutation strategies selected by a random process [32]. In the last version, it incorporates 4 mutation strategies selected by the roulette method improving the performance [20].

The comparative analysis of binomial and exponential crossover variants provides information about the influence of the crossover parameter on the behavior of DE. The dependence between the mutation probability and the crossover parameter is linear in the binomial case and nonlinear in the exponential one. The use of both types of crossover together makes the algorithms more robust [49]. Nevertheless, it is not possible to generalize, since a combination of parameters can be effective for a problem or instance of one problem and ineffective for another. In order to combat these weaknesses, our approach encapsulates a set of predefined heuristics (DE models as low-level heuristic) for the given problem, a fitness evaluation function, and a specific search space. The high-level heuristic decides which low-level heuristic (DE model) will be chosen. This can be achieved with a learning mechanism that evaluates the quality of the heuristic (DE model) solutions, so that they can become general enough to solve unseen instances of a given problem. In the hyper-heuristic framework, it is possible to add or remove low-level heuristics without the need to code the entire algorithm again. The main design differences between SaDE and our approach are presented in Table 6 (Illustrative example).

Consider the following *quadratically* constrained linear program taken from [34]:

$$\begin{aligned} \min \quad & x_1 + x_2 \\ s.t. \quad & x_1^2 + x_x^2 \leq 4, \\ & x_1^2 + x_x^2 \geq 1, \\ & x_1 - x_2 \leq 1, \\ & x_2 - x_1 \leq 1, \\ & -2 \leq x \leq 2, \end{aligned} \quad (15)$$

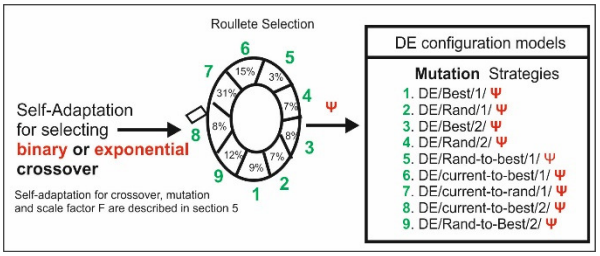

Fig. 4. Crossover and mutation over a hyper-heuristic framework

ISSN 2007-9737

Table 2. Comparison between DE-HH and SADE algorithm

		SADE				DE-HH			
Problem	**Best Reported**	**Best**	**Worst**	**Mean**	**Std.**	**Best**	**Worst**	**Mean**	**Std.**
1	2.0000	2.0000	1.8269	1.9711	5.91E-02	2.0000	1.9237	1.9865	2.63E-02
2	2.1240	2.1236	2.0786	2.1093	1.91E-02	2.1240	2.0936	2.1171	1.06E-02
3	1.0765	1.0693	1.0438	1.0695	1.02E-02	1.0762	1.0534	1.0719	6.98E-03
4	99.2452	99.2450	99.1184	99.2202	4.14E-02	99.2451	99.1963	99.2318	1.83E-02
5	3.5574	3.5461	3.0530	3.4649	1.52E-01	3.5574	3.4384	3.5294	3.62E-02
6	32217.4	32216.9657	32214.9618	32216.1482	6.50E-01	32217.4350	32215.3681	32216.6453	8.52E-01
7	38499.8	38498.217	38495.328	38497.03	1.33E+00	38499.7619	38496.532	38498.9411	1.29E+00

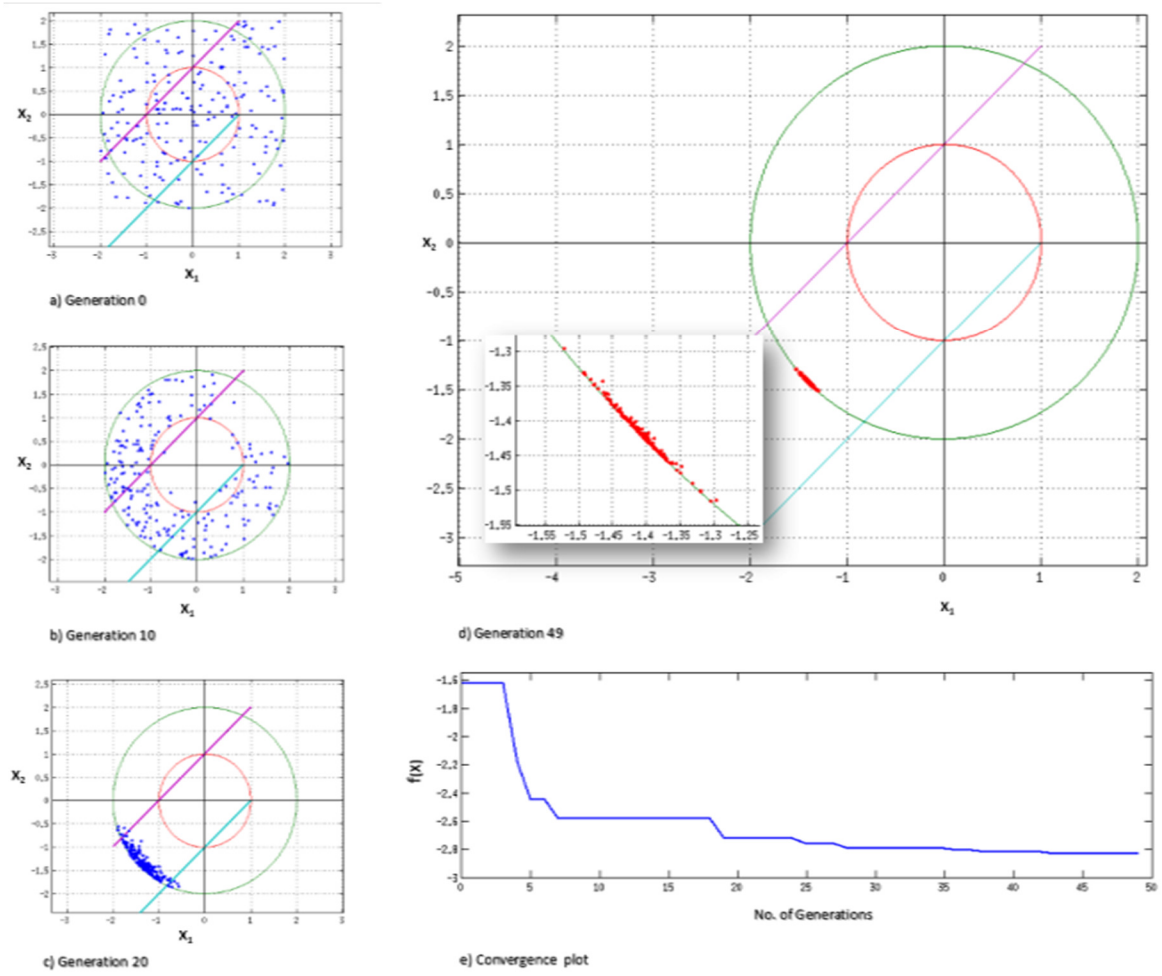

Fig. 5. Evolution of the population from the initial generation (a) to the end generation (d) and the full trace of the convergence plot (e)

where the global optimum reported in the literature is x=(-1.414214, -1.414214) with f= -2.828427. Two local solutions are at x=(-1,0) with f= -1 and x=(1,0) with f=1. Thus, the solution is required to be in the region bounded by all the constraints. Figure 5 shows how this problem is solved by applying the proposed approach.

In order to have sufficient data and show their behavior, the parameters used are the population size $Np = 200$ and the maximum number of generations MaxGen=50; each of the self-adaptive variables, namely, CRsel, fp, and CRm, starts with a value of 0.5. After 20, 20, and 5 generations, respectively, the variables are updated (***"learning period")***. The algorithm was able to find the global optimum in 0.018 s, with 10200 evaluations of the objective function.

In an experimental test with a population size of $Np = 25$ and the maximum number of generations MaxGen=30, the global optimum was found in 0.00343 s, with 775 evaluations of the objective function.

7 Case Studies

Seven problems from the field of chemical engineering, which involves complex non-convex optimization problems with continuous and discrete variables, were considered in the present study. Definitions of these Benchmark Problems are presented in Appendix A.

8 Results

Our approach, namely, the DE-HH algorithm, is implemented in C and compiled using GCC version 4.8.2. All computations were carried out on a standard PC (Linux Kubuntu 14.04 LTS, Intel core i5, 2.20 GHz, 4 GB).

The reliability and efficiency of our approach were compared with those of several state-of-the-art algorithms reported in the literature. The comparison involves the mean values of ten experiments for each problem; it includes two parts as follows.

In the first part, a comparison is made against the SaDE algorithm [20] measuring convergence speed and quality of results. The results are based on the best, the worst, mean, and standard deviation; these are listed in Table 2. A convergence graph for each problem was plotted. The graph shows the median run of the total runs with termination by the max number of function evaluations obtained. We use the function value of the problem without penalties, f(x), and the fitness value of best-known solution, f(x*). In the log graphs, the x-axis corresponds to the number of function evaluations and the y-axis corresponds to the log (f(x)-f(x*)), see Figures 6 to 12.

The comparison shows that our approach improves the best results reported of the SaDE algorithm, including better mean and standard deviation.

In the second part, a comparison is made against genetic algorithm (GA), simplex-simulated annealing (M-SIMPSA & M-SIMPSA-pen variant), evolution strategies (ES), and modified differential evolution (MDE); all these algorithms are reported in [1], while particle swarm optimization (R-PSO_c) is reported in [47].

Table 3. DE-HH Results

Problem	NFE	NRC	CPU-Time
1	420	100	0.002111
2*	440	100	0.002169
3	1020	100	0.007012
4*	1680	100	0.012646
5	6030	100	0.047243
6	2020	100	0.026431
7	14600	100	3.841227

Table 4. Percent reduction in NFE due to DE-HH as compared with the best algorithm reported

Problem No.	Our approach	% Reduction in NFE by DE-HH	Best algorithm reported
1	DE-HH	30.81 %	M-SIMPSA
2*		10.20%	MDE
3		41.68%	ES
4*		6.51%	MDE
5		10.14%	ES
6		20.35%	ES
7		2.67%	R_PSO_C

Table 5. Comparison of DE-HH, GA, M-SIMPSA, M-SIMPSA-pen, ES, MDE & R-PSO_c

Problem no.	ratio NFE/NRC						
	GA	M-SIMPSA	M-SIMPSA-Pen	ES	MDE	R-PSO_c	DE-HH
1	67.87	6.13	162.82	15.18	7.05	--	**4.20**
2*	139.39	127.49	144.40	22.55	4.90	35.00	**4.40**
3	1070.46	#/0	380.42	17.49	19.74	--	**10.20**
4*	224.89	147.38	422.95	**/0	17.97	40.00	**16.80**
5	1712.96	371.816	657.22	67.10	119.14	300.00	**60.30**
6	371.67	315.057	357.43	25.36	54.95	--	**20.20**
7	α	#/0	2799.33	**/0	405.50	166.66	**146.00**

Execution halted, ** Converged to a non-optimal solution,
--such results were not available for the corresponding algorithm
α: 225176 of NFE and zero of NRC reported

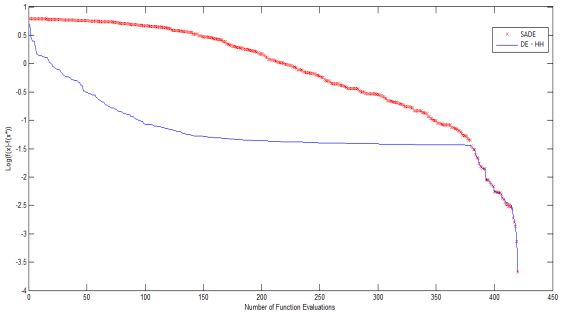

Fig. 6. Convergence Graph Problem 1

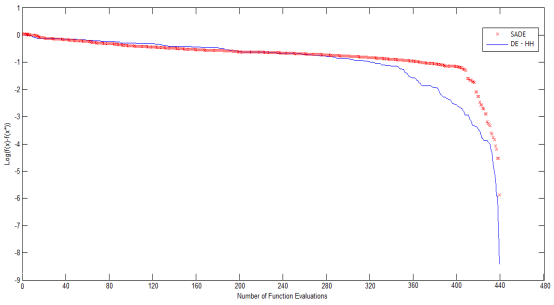

Fig. 7. Convergence Graph Problem 2

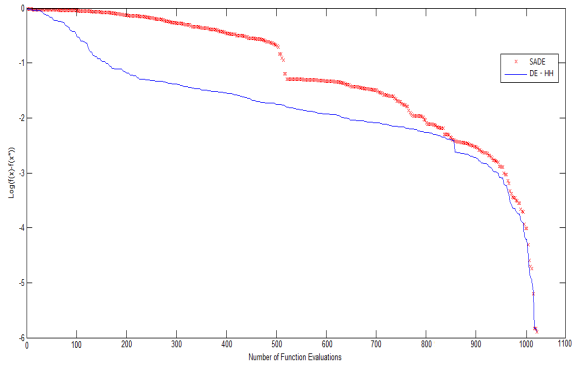

Fig. 8. Convergence Graph Problem 3

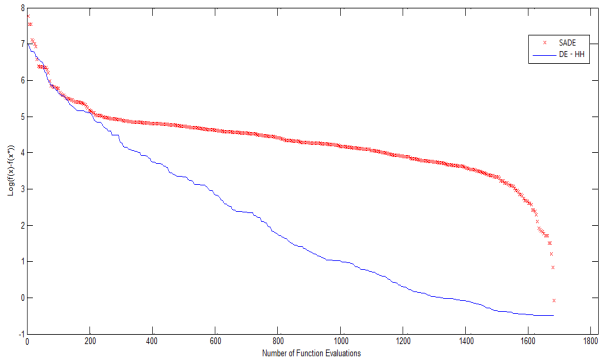

Fig. 9. Convergence Graph Problem 4

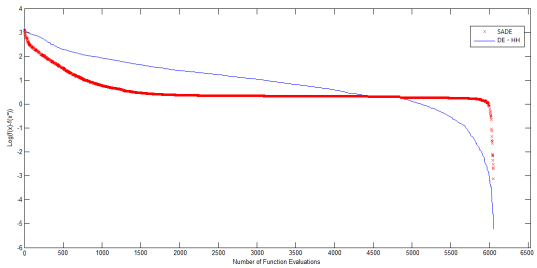

Fig. 10. Convergence Graph Problem 5

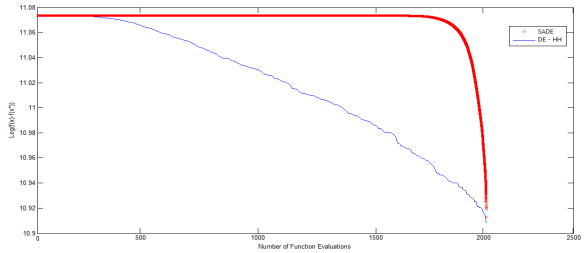

Fig. 11. Convergence Graph Problem 6

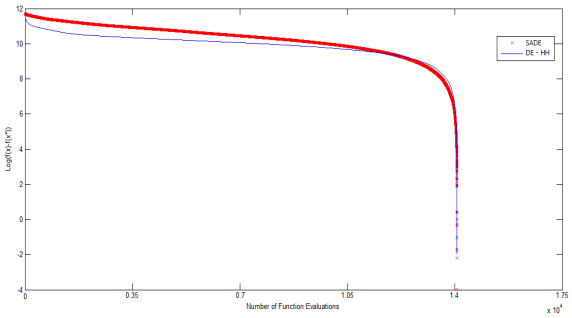

Fig. 12. Convergence Graph Problem 7

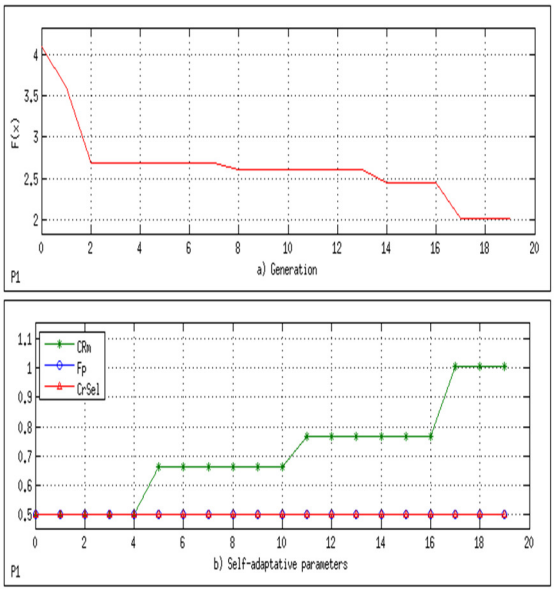

Fig. 13. Problem 1

The results are based on the percentage of convergences to the global optimum (NRC) and the average number of objective function evaluations (NFE); these are listed in Table 3, where the CPU time in seconds is also reported. The best results obtained are listed in Table 4. For Problem 1, a comparison shows that the NFE for DE-HH is around 30.82\% less than that for M-SIMPSA and 93.82\% less than that for GA.

Our approach improves the NFE and NRC of the M-SIMPSA reported as the best for this problem.

For Problem 2, the NFE for DE-HH is 97.3% less than that for GA. Moreover, for Problems 1 to 7, the NFE for DE-HH is around 40.42%, 10.20%, 48.33%, 6.51%, 49.38%, 63.23%, and 64% less, respectively, than that for MDE. For Problem 7, our approach improves the NFE and NRC of the R_PSO_c algorithm, which has been reported as the best. A summarized comparison of DE-HH with the best algorithms reported for each problem is presented in Table 5.

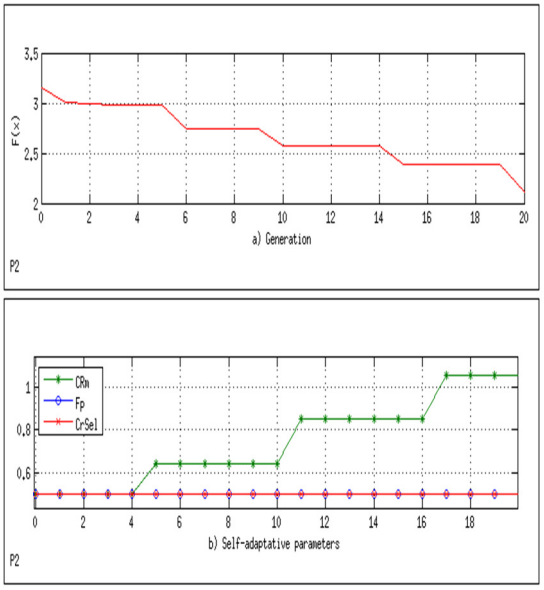

Fig. 14. Problem 2*

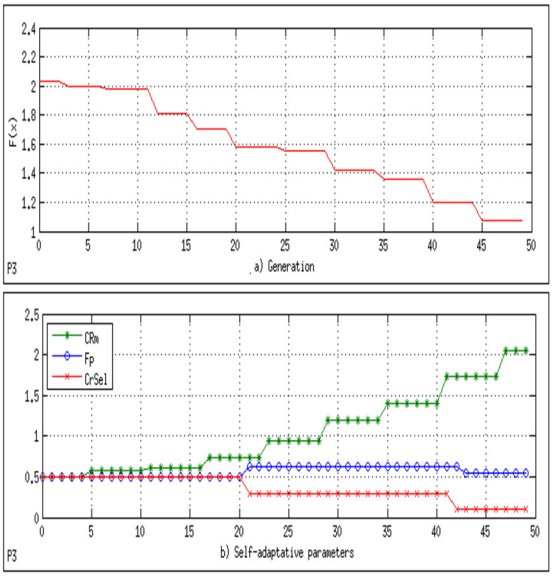

Fig. 15. Problem 3

The convergence plot, labeled as (a), and the behavior of the self-adaptive parameters (CRm, Fp, and CrSel), labeled as (b), for each problem are shown in Figures 4 to 10.

For the convergence plot, the x-axis corresponds to the number of generations and the y-axis corresponds to the values of $f(x)$. For the self-adaptive parameters, the x-axis corresponds to the number of generations and the y-axis corresponds to the obtained values of CRm, Fp, and CrSel. Observe that in Figures 5 to 7, the y-axis appears with different scales on the left and right sides. More specifically, the y-axis on the left corresponds to Fp and CrSel, and the y-axis on the right corresponds to CRm. CPU time in seconds.

9 Conclusions

This paper proposed a differential-evolution-based hyper-heuristic (DE-HH) approach for the optimization of mixed-integer non-linear programming (MINLP) problems. Self-adaptive mechanisms of the control parameters in the DE algorithm are carried out over the hyper-heuristic framework. The constraints are handled by the epsilon-constrained method. The choice functions of the proposed framework can adaptively select appropriate low-level heuristics from a set of 18 DE variants. Additional mutation strategies and different crossover schemes can also be applied to the hyper-heuristic framework in order to adapt it to a particular problem.

We conducted experimental studies on test instances of process synthesis and design that represent difficult non-convex optimization problems often encountered in the field of chemical engineering. The results, which were based on the percentage of convergences to the global optimum (NRC) and the average number of objective function evaluations (NFE), showed that DE-HH can find a global optimum reliably and efficiently, improving, on average, the NFE by 17.48% as compared to the best algorithms reported while maintaining the NRC at 100%. The results, which were based on the best, the worst, mean and standard deviation, showed that our approach exhibits better high quality results for all benchmark problems than SaDE algorithm, including better mean and standard deviation.

In a runtime it is possible to find DE models untouched (unused) by the hyper-heuristic.

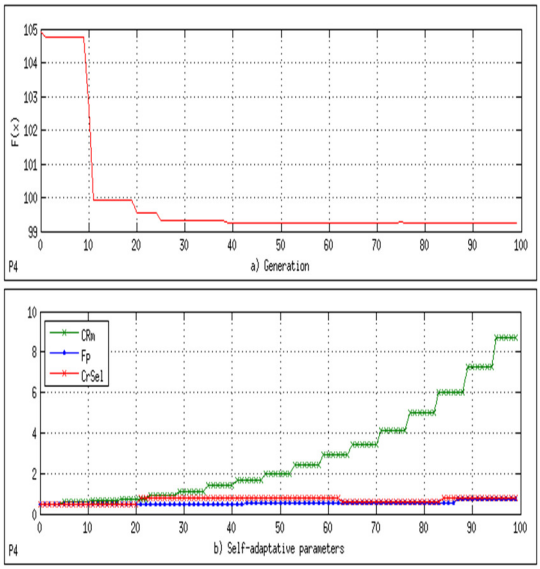

Fig. 16. Problem 4*

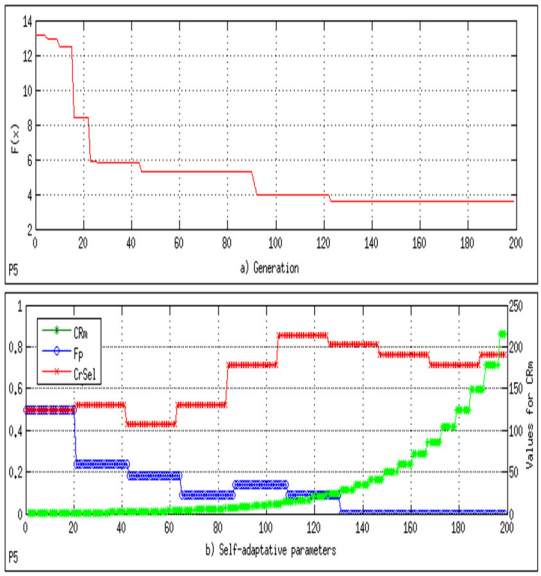

Fig. 17. Problem 5

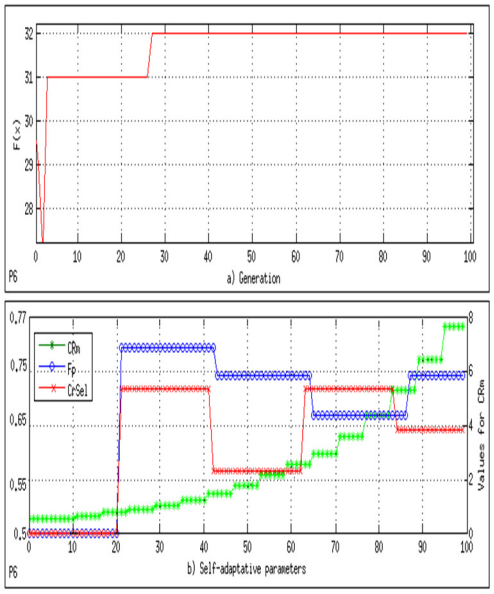

Fig. 18. Problem 6 (maximization)

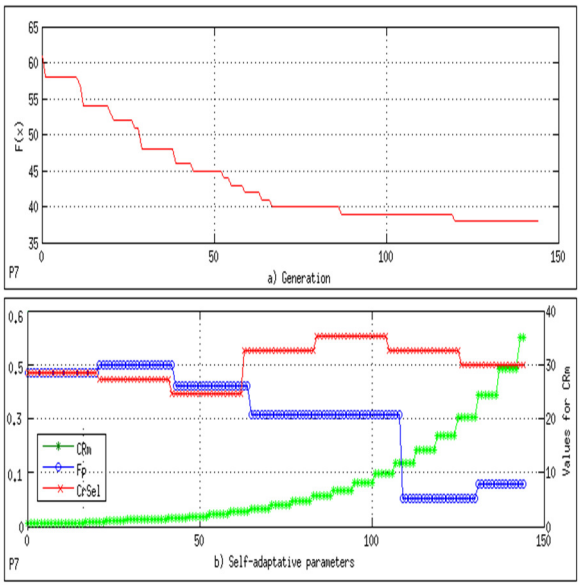

Fig. 19. Problem 7

Thereby, it is unknown which of the 18 DE models are more requested for a particular problem.

Also, the contribution of each model in obtain an optimal solution, the effects of adding (or subtracting) of more DE models and its repercussions in the quality of results are unknown. Therefore, directions for future work include an analysis of sensitivity of variables and models over a hyper-heuristic environment. In addition, it is desirable to deal with larger problem

instances to improve the percentage of convergences to the global optimum and improve the average number of objective function evaluations using parallelization strategies for hyper-heuristics, e.g., GPU computing and multi-core resources.

Acknowledgements

The first author acknowledges the facilities provided by Instituto Politécnico Nacional for this project.

Appendix A

Problem 1. (Process synthesis problem). This problem has one real variable, one binary variable, one non-linear inequality constraint, and one linear inequality constraint. The problem was proposed in [21]; it has also been solved in [9,10,14,34]:

$$\begin{aligned} \min \quad & f(x,y) = 2x + y \\ s.t. \quad & 1.25 - x^2 - y \le 0 \\ & x + y \le 1.6 \\ & 0 \le x \le 1.6 \\ & y \in \{0,1\} \end{aligned} \tag{16}$$

The global optimum is (x,y;f)=(0.5,1;2). There is a local minimum at x=1.118 and y=0 with f=2.236.

Problem 2. (Process synthesis and design problem). This problem has two real variables, one binary variable, one non-linear equality constraint, and one linear inequality constraint. It was proposed in [21] and has also been studied in [9, 35]:

$$\begin{aligned} \min \quad & f(x_1, x_2, y) = -y + 2x_1 + x_2 \\ s.t. \quad & 1.25 - x^2 - y \le 0 \\ & x_1 - 2\exp(-x_2) = 0 \\ & 0.5 \le x_1 \le 1.4 \\ & y \in \{0,1\} \end{aligned} \tag{17}$$

The global optimum is $(x_1, x_2, y : f) =$. $(1.375, 0.375, 1; 2.124)$.

Problem 2*. (Process synthesis and design problem). Problem 2 can also be formulated without the non-linear equality constraint with the same global optimum:

$$\begin{aligned} \min \quad & f(x_1, y) = -y + 2x_1 - \ln(x_1 / 2), \\ s.t. \quad & -x_1 - \ln(x_1 / 2) + y \le 0, \\ & 0.5 \le x_1 \le 1.4 \\ & y \in \{0,1\} \end{aligned} \tag{18}$$

Problem 3. (Process flowsheeting problem). This non-convex problem has two real variables, one binary variable, one non-linear inequality constraint, and two linear inequality constraints. It was studied in [15] and has also been solved in [9,10]:

$$\begin{aligned} \min \quad & f(x_1, x_2, y) = -0.7y + 5(x_1 - 0.5)^2 + 0.8, \\ s.t. \quad & -\exp(-x_1 - 0.2) - x_2 \le 0, \\ & x_2 + 1.1y \le -1.0, \\ & x_1 - 1.2y \le 0.2, \\ & -2.22554 \le x_2 \le 1, \\ & y \in \{0,1\} \end{aligned} \tag{19}$$

The global optimum is $(x_1, x_2, y : f) =$(0.94194, -2.1,1;1.07654).

Problem 4. (Two-reactor problem). This problem, taken from [22], has seven real variables, two

Table 6. Constants of Problem 6

a_1=85.334407	a_5=80.512490	a_9=9.30096100
a_2=0.0056858	a_6=0.0071317	a_{10}=0.0047026
a_3=0.0006262	a_7=0.0029955	a_{11}=0.0012547
a_4=0.0022053	a_8=0.0021813	a_{12}=0.0019085

Table 7. Values of constants of S_{ij} and T_{ij} of Problem 7

S_{ij}			t_{ij}		
2	3	4	8	20	8
4	6	3	16	4	4

binary variables, two non-linear variables, four non-linear equality constraints, and four linear inequality constraints. The objective is to select one of two candidate reactors in order to minimize the production cost. This problem has also been solved in [9, 10, 11, 12].

$$\begin{aligned}
\min \quad & f(x_1,y_1,y_2,v_1,v_2) = 7.5y_1 + 5.5y_2 + 7v_1 + 6v_2 + 5x \\
s.t. \quad & y_1 + y_2 = 1, \\
& z_1 = 0.9[1-\exp(-0.5v_1)]x_1, \\
& z_2 = 0.8[1-\exp(-0.4v_2)]x_2, \\
& z_1 + z_2 = 10, \\
& x_1 + x_2 = x, \\
& z_1y_1 + z_2y_2 = 10, \\
& v_1 \le 10y_1, \\
& v_2 \le 10y_2, \\
& x_1 \le 20y_1, \\
& x_2 \le 20y_2, \\
& x_1,x_2,z_1,z_2,v_1,v_2 \ge 0 \\
& y_1,y_2 \in \{0,1\}
\end{aligned} \tag{20}$$

The global optimum is $(x, y_1, y_2, v_1, v_2; f) =$ (13.36227,1,0,3.514237,0;99.245209).

Problem 4*. (Two-reactor problem) Problem 4 can be reformulated without equality constraints as follows:

$$\begin{aligned}
\min \quad & f(y_1,v_1,v_2) = 7.5y_1 + 55(1-y_1) + 7v_1 + 6v_2 + \\
& 50\frac{1-y_1}{0.8[1-\exp(-0.4v_2)]} + \\
& 50\frac{y_1}{0.9[1-\exp(-0.5v_1)]} \\
s.t. \quad & 0.9[1-\exp(-0.5v_1)] - 2y_1 \le 0, \\
& 0.8[1-\exp(-0.4v_2)] - 2(1-y_1) \le 0. \\
& v_1 \le 10, \\
& v2 \le 10(1-y_1) \\
& v_1,v_2 \ge 0, \\
& y_1 \in \{0,1\}
\end{aligned} \tag{21}$$

The global optimum is the same as that in Problem 4.

Problem 5. (Process synthesis problem). This problem features non-linearities in both continuous and binary variables, and it has seven degrees of freedom. The problem was studied in [9,10,14,35].

$$\begin{aligned}
\min \quad & f(x_1,x_2,x_3,y_1,y_2,y_3,y_4) = (y_1-1)^2 + (y_2-1)^2 + \\
& (y_3-1)^2 - \ln(y_4+1) + (x_1-1)^2 + (x_2-2)^2 + (x_3-3)^2 \\
s.t. \quad & y_1+y_2+y_3+x_1+x_2+x_3 \le 5, \\
& y_3^2 + x_1^2 + x_2^2 + x_3^2 \le 5.5, \\
& y_1 + x_1 \le 1.2, \\
& y_2 + x_2 \le 1.8, \\
& y_3 + x_3 \le 2.5, \\
& y_4 + x_1 \le 1.2, \\
& y_2^2 + x_2^2 \le 1.64, \\
& y_3^2 + x_3^2 \le 4.25, \\
& y_2^2 + x_3^2 \le 4.64, \\
& x_1,x_2,x_3 \ge 0, \\
& y_1,y_2,y_3,y_4 \in \{0,1\}
\end{aligned} \tag{22}$$

The global optimum is $(x_1, x_2, x_3, y_1, y_2, y_3, y_4; f) =$ (0.2,1.28062,1.95448, 1, 0,0,1; 3.557473).

Problem 6. (Process design problem). This is a maximization problem studied in [9,10]. It has three real variables, two integer variables, and three inequality constraints:

$$\begin{aligned}
\max \quad & f(x_1,x_2,x_3,y_1,y_2) = -5.37854x_1^2 - \\
& 0.835689y_1x_3 - 37.29329y_1 + 40792.141 \\
s.t. \quad & a_1 + a_2y_2x_3 + a_3y_1x_2 - a_4x_1x_3 \le 92, \\
& a_5 + a_6y_2x_3 + a_7y_1x_2 - a_8x_1^2 - 90 \le 20, \\
& a_9 + a_{10}x_1x_3 + a_{11}y_1x_1 + a_{12}x_1x_2 - 20 \le 5, \\
& 27 \le x_1,x_2,x_3 \le 45, \\
& y_1 \in \{78,...,102\}, \text{int}\, eger, \\
& y_2 \in \{33,...,45\}, \text{int}\, eger,
\end{aligned} \tag{23}$$

where $a_1 - a_{12}$ are constants, the values of which are listed in Table 6.

The global optimum (for any combination of x_2, y_2 is $(x_1, x_3, y_1; f) = (27, 27, 78; 32217.4)$.

Problem 7. (Multi-product batch plant (MPBP)). This is a multi-product batch problem with M serial processing stages, where fixed amounts Q_i of N products must be produced. The objective is to determine for each stage j, the number of parallel

Table 8. 18 DE – models (9 mutation strategies for each exponential and binomial crossover strategies)

#	Exponential strategy	#	Binomial strategy	Representation
1	DE/best/1/exp	10	DE/best/1/bin	$V_{i,G} = X_{best,G} + F * (X_{r1,G} - X_{r2,G})$
2	DE/rand/1/exp	11	DE/rand/1/bin	$V_{i,G} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G})$
3	DE/best/2/exp	12	DE/best/2/bin	$V_{i,G} = X_{best,G} + F * (X_{r1,G} + X_{r2,G} - X_{r3,G} - X_{r4,G})$
4	DE/rand/2/exp	13	DE/rand/2/bin	$V_{i,G} = X_{r5,G} + F * (X_{r1,G} + X_{r2,G} - X_{r3,G} - X_{r4,G})$
5	DE/rand-to-best/1/exp	14	DE/rand-to-best/1/bin	$V_{i,G} = X_{i,G} + F * (X_{best,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G})$
6	DE/current-to-rand/1/exp	15	DE/current-to-rand/1/bin	$V_{i,G} = X_{i,G} + K * (X_{r3,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G})$
7	DE/current-to-best/1/exp	16	DE/current-to-best/1/bin	$V_{i,G} = X_{i,G} + K * (X_{best,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G})$
8	DE/current-to-best/2/exp	17	DE/current-to-best/2/bin	$V_{i,G} = X_{i,G} + K * (X_{best,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G}) + F * (X_{r3,G} - X_{r4,G})$
9	DE/rand-to-best/2/exp	18	DE/rand-to-best/2/bin	$V_{i,G} = X_{i,G} + F * (X_{best,G} - X_{i,G}) + F * (X_{r1,G} - X_{r2,G}) + F * (X_{r3,G} - X_{r4,G})$

units N_j together with the respective sizes V_j, and for each product i, the corresponding batch sizes B_i and cycle times T_{Li}. The data are the horizon time H, size factor S_{ij}, processing times t_{ij} of product i in stage j, required production Q_i,and cost coefficients α_j and β_j. This problem, studied in [9,10,17,21,35], has the following mathematical formulation:

$$\begin{aligned} \min \quad & f = \sum_{j=1}^{M} \alpha N_j V_j^{\beta}, \\ s.t. \quad & \sum_{i=1}^{N} \frac{Q_i T_{Li}}{B_i} \le H, \\ & V_j \ge S_{ijB_i}, \\ & N_j T_{L_i} \ge t_{ij}, \\ & 1 \le N_j \le N_j^u, \\ & V_j^l \le V_j \le V_j^u, \\ & T_{Li}{}^l \le T_{Li} \le T_{Li}{}^u, \\ & B_j^l \le B_j \le B_j^u, \end{aligned} \tag{24}$$

where for the specific problem considered, M=3, N=2, H=6000, α_j=250, β_j=0.6, N_j^u=3, V_j^l=250, V_j^u =2500, Q_1=40000, and Q_2=20000. The values of $T_{Li}{}^l$, $T_{Li}{}^u$, B_j^l, and B_j^u are given by $T_{Li}{}^l = max(t_{ij} / N_j^u)$, $T_{Li}{}^u = max(t_{ij})$, $B_i^l = \frac{Q_i}{H} T_{Li}^l$, and $B_i^u = \min\left\{ Q_i, \min\left(\frac{V_j^u}{S_{ij}}\right) \right\}$.

The values of S_{ij} and t_{ij} [i=1-2 (rows); and j=1-3 (columns)] are given in Table 7. The problem has 7 real variables, 3 integer variables, and 18 inequalities constraints. The global optimum is $(N_1, N_2, N_3, V_1, V_2, V_3, B_1, B_2, T_1, T_2; f)$ = (1,1,1,480,720,960,240,120,20,16;38499.8).

Various mutation strategies for each exponential and binomial crossover strategies are presented in Table 8.

References

1. **Angira, R. & Babu, B. (2006).** Optimization of process synthesis and design problems: A modified differential evolution approach. *Chemical Engineering Science,* Vol. 61, No. 14, pp. 4707–4721. DOI: 10.1016/j.ces.2006.03.004.
2. **Babu, B.V. & Angira, R. (2006).** Modified differential evolution (MDE) for optimization of non-

linear chemical processes. *Computers & Chemical Engineering*, Vol. 30, No. 6-7, pp. 989–1002. DOI: 10.1016/j.compchemeng.2005.12.020.

3. **Boussad, I., Lepagnot, J., & Siarry, P. (2013).** A survey on optimization metaheuristics. *Inf. Sci.*, Vol. 237, pp. 82–117. DOI: 10.1016/j.ins.2013.02.041.
4. **Brest, J., Boskovic, B., Zamuda, A., Fister, I., & Maucec, M.S. (2012).** Self-adaptive differential evolution algorithm with a small and varying population size. *IEEE Congress on Evolutionary Computation,* pp. 1–8. DOI: 10.1109/CEC.2012.6252909.
5. **Brest, J., Zamuda, A., & Boskovic, B. (2015).** Adaptation in the Differential Evolution. **Fister Jr.I. (Eds),** *Adaptation and Hybridization, in computational Intelligence,* Vol. 18 of Adaptation, Learning, and Optimization. Springer International Publishing, pp.55–68. DOI: 10.1007/978-3-319-14400-9_2.
6. **Bujok, P., Tvrdik, J., & Polakova, R. (2014).** Differential evolution with rotation-invariantmutation and competing-strategies adaptation. *IEEE Congress on Evolutionary Computation* (*CEC*), pp. 2253–2258. DOI: 10.1109/CEC.2014.6900626.
7. **Burke, E.K., Gendreau, M., Hyde, M.R., Kendall, G., Ochoa, G., O. zcan, E., & Qu, R. (2013).** Hyper-heuristics: a survey of the state of the art. *JORS,* Vol. 64, No. 12, pp. 1695–1724.
8. **Burke, E.K., Hyde, M.R., Kendall, G., & Woodward, J., (2010).** A Genetic Programming Hyper-Heuristic Approach for Evolving 2-D Strip Packing Heuristics. *IEEE Trans. Evolutionary Computation*, Vol. 14, No. 6, pp. 942–958. DOI: 10.1109/TEVC.2010.2041061.
9. **Cardoso, M., Salcedo, R., de Azevedo, S., & Barbosa, D. (1997).** A simulated annealing approach to the solution of MINLP problems. *Computers & Chemical Engineering*, Vol. 21, No. 12, pp. 1349–1364. DOI: 10.1016/S0098-1354(97)00015-X.
10. **Costa, L. & Oliveira, P. (2001).** Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. *Computers & Chemical Engineering*, Vol. 25, No. 2-3, pp. 257–266. DOI: 10.1016/S0098-1354(00)00653-0.
11. **Diwekar, U. & Rubin, E. (1993).** Efficient handling of the implicit constraints problem for the ASPEN MINLP synthesizer. *Industrial & engineering chemistry research,* Vol. 32, No. 9, pp. 2006–2011. DOI: 10.1021/ie00021a023.
12. **Diwekar, U.M., Grossmann, I.E., & Rubin, E.S. (1992).** An MINLP process synthesizer for a sequential modular simulator. *Industrial & engineering chemistry research,* Vol. 31, No. 1, pp. 313–322. DOI: 10.1021/ie00001a042.
13. **Epitropakis, M.G., Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., & Vrahatis, M.N. (2011).** Enhancing Differential Evolution Utilizing Proximity-Based Mutation Operators. *IEEE Trans. Evolutionary Computation*, Vol. 15, No. 1, PP. 99–119. DOI: 10.1109/TEVC.2010.2083670.
14. **Floudas, C., Aggarwal, A., & Ciric, A. (1989).** Global optimum search for nonconvex NLP and MINLP problems. *Computers & Chemical Engineering,* Vol. 13, No. 10, pp. 1117–1132. DOI: 10.1016/0098-1354(89)87016-4.
15. **Floudas, C.A. (1995).** *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications.* Oxford University Press.
16. **Goncalves, R., Kuk, J., Almeida, C., & Venske, S. (2015).** MOEA/D-HH: A Hyper-Heuristic for Multi-objective Problems. **Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (Eds.),** *Evolutionary Multi-Criterion Optimization*.
17. **Grossmann, I.E. & Sargent, R.W. (1979).** Optimum design of multipurpose chemical plants. *Industrial & Engineering Chemistry Process Design and Development,* Vol. 18, No. 2, pp. 343–348. DOI. 10.1021/i260070a031.
18. **Guo, S. & Yang, C. (2014).** Enhancing Differential Evolution Utilizing Eigenvector-Based Crossover Operator. *IEEE Transactions on Evolutionary Computation*, Vol. 19, No. 1, pp. 31-49. DOI: 10.1109/TEVC.2013.2297160.
19. **Hu, Z., Bao, Y., & Xiong, T. (2014).** Partial opposition-based adaptive differential evolution algorithms: Evaluation on the CEC 2014 benchmark set for real-parameter optimization. *IEEE Congress on Evolutionary Computation* (*CEC*), pp. 2259–2265. DOI: 10.1109/CEC.2014.6900489.
20. **Huang, V.L., Qin, A.K., & Suganthan, P.N. (2006).** Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. *IEEE Congress on evolutionary computation*, pp. 17–24. DOI: 10.1109/CEC.2006.1688311.
21. **Kocis, G.R. & Grossmann, I.E. (1988).** Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Industrial & Engineering Chemistry Research*, Vol. 27, No. 8, pp. 1407–1421. DOI: 10.1021/ie00080a013.
22. **Kocis, G.R. & Grossmann, I.E. (1989).** A modelling and decomposition strategy for the MINLP optimization of process flowsheets. *Computers & Chemical Engineering,* Vol. 13, No. 7, pp. 797–819. DOI: 10.1016/0098-1354(89)85053-7.

23. **Lampinen, J. (2002).** Multi-Constrained Nonlinear Optimization by the Differential Evolution Algorithm. **Roy, R., Koppen, M., Ovaska, S., Furuhashi, T., Hoffmann, F. (Eds.),** *Soft Computing and Industry,* Springer London, pp. 305–318. DOI: 10.1007/978-1-4471-0123-9_26.
24. **Mallipeddi, R. & Suganthan, P.N. (2010).** Differential evolution with ensemble of constraint handling techniques for solving CEC 2010 benchmark problems. *IEEE Congress on Evolutionary Computation*, pp. 1–8. DOI: 10.1109/CEC.2010.5586330.
25. **Marshall, R.J., Johnston, M., & Zhang, M. (2014).** Hyper-heuristics, Grammatical Evolution and the Capacitated Vehicle Routing Problem. *Proceedings of the Conference Companion on Genetic and Evolutionary Computation Companion* (*GECCO*) Comp'14 ACM, New York, NY, USA, pp. 71–72. DOI: 10.1145/2598394.2598407.
26. **Mezura-Montes, E. & Coello, C.A.C. (2011).** Constraint handling in nature inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation,* Vol. 1, No. 4, pp. 173–194. DOI: 10.1016/j.swevo.2011.10.001.
27. **Munawar, A., Wahib, M., Munetomo, M., & Akama, K. (2011).** Advanced genetic algorithm to solve MINLP problems over GPU. *IEEE Congress on Evolutionary Computation*, pp. 318–325. DOI: 10.1109/CEC.2011.5949635.
28. **Ortiz-Bayliss, J.C., Terashima-Marin, H., Ozcan, E., & Parkes, A.J. (2011).** On the idea of evolving decision matrix hyper-heuristics for solving constraint satisfaction problems. **Krasnogor, N., Lanzi, P.L. (Eds.),** *GECCO*, ACM, pp. 255–256. DOI: 10.1145/2001858.2002002.
29. **Pillay, N. (2012).** A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem. *South African Computer Journal,* Vol. 48, pp. 31–40. DOI: 10.18489/sacj.v48i1.87.
30. **Pillay, N. (2012).** Evolving hyper-heuristics for the uncapacitated examination timetabling problem. *JORS*, Vol. 63, No. 1, pp. 47–58. DOI: 10.1057/jors.2011.12
31. **Qin, A.K., Huang, V.L., & Suganthan, P.N. (2009).** Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evolutionary Computation*, Vol. 13, No. 2, pp. 398–417. DOI: 10.1109/TEVC.2008.927706.
32. **Qin, A.K. & Suganthan, P.N. (2005).** Self-adaptive differential evolution algorithm for numerical optimization. *IEEE Congress on Evolutionary Computation,* pp. 1785–1791. DOI: 10.1109/CEC.2005.1554904.
33. **Robertson, G., Geraili, A., Kelley, M., & Romagnoli, J.A. (2014).** An active specification switching strategy that aids in solving nonlinear sets and improves a VNS/TA hybrid optimization methodology. *Computers & Chemical Engineering*, Vol. 60, pp. 364–375. DOI: 10.1016/j.compchemeng.2013.10.002.
34. **Ryoo, H. & Sahinidis, N. (1995).** Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, Vol. 19, No. 5, pp. 551–566. DOI: 10.1016/0098-1354(94)00097-2.
35. **Salcedo, R. (1992).** Solving nonconvex nonlinear programming and mixed-integer nonlinear programming problems with adaptive random search. *Industrial & Engineering Chemistry Research,* Vol. 31, No. 1, pp. 262–273. DOI: 10.1021/ie00001a037.
36. **Schluter, M., Egea, J.A., & Banga, J.R. (2009).** Extended ant colony optimization for nonconvex mixed integer nonlinear programming. *Computers & OR*, Vol. 36, No. 7, pp. 2217–2229. DOI: 10.1016/j.cor.2008.08.015.
37. **Storn, R. & Price, K. (1995***). Differential Evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces*.
38. **Storn, R. & Price, K. (1996).** Minimizing the Real Functions of the ICEC'96 Contest by Differential Evolution. *International Conference on Evolutionary Computation*, pp. 842–844.
39. **Takahama, T. & Sakai, S. (2012).** Efficient Constrained Optimization by the Constrained Rank-Based Differential Evolution. *IEEE Congress on Evolutionary Computation*, pp. 1–8. DOI: 10.1109/CEC.2010.5586545.
40. **Takahama, T. & Sakai, S. (2013).** Efficient constrained optimization by the constrained differential evolution with rough approximation using kernel regression. *IEEE Congress on Evolutionary Computation*, pp. 1334–1341. DOI: 10.1109/CEC.2013.6557719.
41. **Takahama, T., Sakai, S., & Iwane, N. (2005).** Constrained Optimization by the epsilon Constrained Hybrid Algorithm of Particle Swarm Optimization and Genetic Algorithm. **Zhang, S., Jarvis, R. (Eds.),** *Australian Conference on Artificial Intelligence*, Vol. 3809 of Lecture Notes in Computer Science, Springer, pp. 389–400. DOI: 10.1007/11589990_41.
42. **Tanabe, R. & Fukunaga, A. (2013).** Evaluating the performance of SHADE on CEC 2013 benchmark problems. *IEEE Congress on Evolutionary*

Computation, pp. 1952–1959. DOI: 10.1109/CEC.2013.6557798.

43. **Tinoco, J.C.V. & Coello, C.A.C. (2012).** hypDE: A Hyper-Heuristic Based on Differential Evolution for Solving Constrained Optimization Problems. **Schutze, O., Coello, C.A.C., Tantar, A.-A., Tantar, E., Bouvry, P., Moral, P.D., & Legrand, P. (Eds.),** *EVOLVE*, Vol. 175 of Advances in Intelligent Systems and Computing, Springer, pp. 267–282. DOI: 10.1007/978-3-642-31519-0_17.
44. **Wolpert, D.H. & Macready, W.G. (1997).** No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 67–82. DOI: 10.1109/4235.585893.
45. **Yang, Z., Tang, K., & Yao, X. (2008).** Self-adaptive differential evolution with neighborhood search. *IEEE Congress on Evolutionary Computation*, pp. 1110–1116. DOI: 10.1109/CEC.2008.4630935.
46. **Yang, Z., Yao, X., & He, J. (2008).** Making a Difference to Differential Evolution. **Siarry, P. & Michalewicz, Z. (Eds.),** *Advances in Metaheuristics for Hard Optimization*, Natural Computing Series, Springer, pp. 397–414. DOI: 10.1007/978-3-540-72960-0_19.
47. **Yiqing, L., Xigang, Y., & Yongjian, L. (2007).** An improved PSO algorithm for solving nonconvex NLP/MINLP problems with equality constraints. *Computers & Chemical Engineering*, Vol. 31, No. 3, pp. 153–162. DOI: 10.1016/j.compchemeng.2006.05.016.
48. **Huang, F., Wang, L., & He, Q., (2007).** An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation*, Vol. 186, No. 1, pp. 340–356. DOI: 10.1016/j.amc.2006.07.105.
49. **Tvrdík, J. (2008).** Adaptive differential evolution and exponential crossover. *Proceedings of the International Multiconference on Computer Science and Information Technology* (*IMCSIT*), pp. 927–931.

Hernán Peraza-Vázquez received the B.Sc degree in Computer Science from the Faculty of Mathematics of the University of Yucatán, Mexico, and the M.Sc. degree in Computer Science from the ITCM, Tamaulipas, Mexico. Currently, M.Sc. Peraza is a Ph.D. candidate in Advanced Technology at CICATA IPN Altamira.

Aidé M. Torres-Huerta received her Ph.D. in Metallurgy and Materials from ESIQIE – IPN, in 2004. Currently Dra. Torres is a member of the Mexican National System of Researchers (SNI) level II. Her areas of interests are analysis of degradation process, synthesis and characterization of nanostructures, simulation and optimization of process in the field of chemical engineering. She has co-authored of more than fifty scientific papers and book chapters in these fields.

Abelardo Flores-Vela received his Ph.D. in Organic Chemistry from CINVESTAV - IPN, in 1988. Currently Dr. Flores is a member of the Mexican National System of Researchers (SNI) level I and is Head of the Mexican Center of Cleaner Production (CMP+L IPN Mexico). His areas of interests are cleaner industrial production, processes synthesis and design optimization in various engineering and industrial fields, highlighting the synthesis and characterization of polymers.

Article received on 18/12/2015; accepted on 09/02/2016.
Corresponding author is Hernán Peraza-Vázquez.