Performance Comparison of Local Search Operators in Differential Evolution for Constrained Numerical Optimization Problems

Saúl Domínguez-Isidro
Department of Artificial Intelligence
Universidad Veracruzana
Xalapa, Veracruz, México
Email: sdominguezisidro@gmail.com

Efrén Mezura-Montes
Department of Artificial Intelligence
Universidad Veracruzana
Xalapa, Veracruz, México
Email: emezura@uv.mx

Guillermo Leguizamón
Research and Development Laboratory
in Computational Intelligence
Universidad de San Luis,
San Luis, Argentina
Email: legui@unsl.edu.ar

***Abstract*—This paper analyzes the relationship between the performance of the local search operator within a Memetic Algorithm and its final results in constrained numerical optimization problems by adapting an improvement index measure, which indicates the rate of fitness improvement made by the local search operator. To perform this analysis, adaptations of Nealder-Mead, Hooke-Jeeves and Hill Climber algorithms are used as local search operators, separately, in a Memetic DE-based structure, where the best solution in the population is used to exploit promising areas in the search space by the aforementioned local search operators. The ε-constrained method is adopted as a constraint-handling technique. The approaches are tested on thirty six benchmark problems used in the special session on "Single Objective Constrained Real-Parameter Optimization" in CEC'2010. The results suggest that the algorithm coordination proposed is suitable to solve constrained problems and those results also show that a poor value of the improvement index measure does not necessarily reflect on poor final results obtained by the MA in a constrained search space.**

I. Introduction

Evolutionary algorithms (EAs) [1] and swarm intelligence algorithms (SIAs) [2] have been widely used to solve constrained numerical optimization problems (CNOPs) also known as constrained nonlinear optimization problems [3]. However, the complexity of many CNOPs has led to the emergence of more sophisticated algorithms either combining operators or mixing different algorithms, which help to get better results. Within this range of algorithms are those which are based on the interaction of two processes, global and local search, where the local search is activated within the generation cycle of the global search, to get the advantage of both searches. This kind of algorithms are known as Memetic Algorithms (MAs).

There are many studies where MAs have been used to solve different type of optimization problems [4], [5]. However, particularly for CNOPs, there is an issue which is not usually analyzed, to the best of the authors' knowledge, and it is the relationship between the performance of the local search and the competitive final results obtained by the MA. This is the main motivation for this work, where three direct local search operators (Nealder-Mead, Hooke-Jeeves and Hill Climber) are added, separately, to Differential Evolution (DE), a highly competitive global search algorithm, with the aim to analyze the performance of each local search operator and the final results obtained by each MA variant. The success of the local search operator is computed by a measure taken from the specialized literature and explained later in the document.

In this work, the ε-constrained method [6] is adopted as constraint-handling mechanism because it is the same used in the best approach presented in the CEC'2010 competition on constrained-real parameter optimization. The ε-constrained method transforms the CNOP into an unconstrained numerical optimization problem by using a so-called ε level comparison.

The contents of this paper are organized as follows: the problem statement is described in Section II. Section III presents a brief review of DE-based memetic algorithms for CNOPs, while in Section IV the proposed approaches in this work are detailed. The performance measure adopted in this comparison is presented in Section V. The experiments and results are summarized in Section VI. Finally, in Section VII, some conclusions are shown and the future work is defined.

II. Problem Statement

The problem tackled in this paper is the constrained numerical optimization problem (CNOPs), which is defined, without loss of generality, as follows:
Minimize

$$F(\vec{x}), \qquad \vec{x} = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n \tag{1}$$

Subject to

$$g_i(\vec{x}) \leq 0, \qquad i = 1, \dots, m \tag{2}$$

$$h_j(\vec{x}) = 0, \qquad j = 1, \dots, p \tag{3}$$

where $\vec{x}$ is the vector of solutions, m is the number of inequality constraints, and p is the number of equality constraints. The search space $\mathcal{S}$ is defined by lower and upper limits $L_k \leq x_k \leq U_k$ for each $x_k \in \mathbb{R}$. The feasible region $\mathcal{F}$ is the set of all solutions which satisfy the constraints functions (Equations 2 and 3). Usually the equality constraints are transformed into inequality constraints as follows [7]: $|h_j(\vec{x})| - \delta \leq 0$, where δ is the tolerance allowed.

III. Related Work

The integration of local search operators in evolutionary algorithms is a good alternative to improve results in different types of optimization problems including CNOPs. Therefore, in this Section a set of Differential-Evolution-based MAs for CNOPs are described below.

According to the local search operator, the related approaches can be grouped into three sets: (1) direct-based operators, (2) gradient-based operators and (3) special operators.

A. *Direct-Based Operators*

Muelas et al. in [8]. proposed a MA based on DE to solve continuous optimization problems. The authors used the multiple trajectory search algorithm as local search operator which was activated after a certain number of generations and was applied to the best solution of the population. Mandal et al. in [9] used an hybrid-mutation strategy to solve real-world optimization problems. The authors combined two techniques in the mutation process (DE/current-to-best/2 and DE/rand/1/bin). Moreover, the authors used the Solis and Wet's algorithm [10] as local search operator, which was applied using an activation frequency criterion. Hernández et al. in [11] proposed a MA based on DE as global search and a hill climber method as local search operator to solve CNOPs. After the selection process in DE, the local search was activated for the best solution of the population. In a recent proposal [12] the authors improved the algorithm combining two mutation operators. Domínguez et al. in [13] proposed a MA to solve CNOPs, the algorithm combined DE as global search and Powell's conjugate direction method as local search operator. The local search was activated during the main cycle of DE and was applied to the best, the worst and a random selected solution. Zhang et al. in [14] proposed a MA that used DE and Hooke-Jeeves as local search operator to solve continuous optimization problems. The local search was activated by means of a probability for each solutions of the population during the DE process. Piotrowski in [15] combined DE and a local search operator inspired by the Nealder-Mead method to solve continuous optimization problems. The local search operator was activated by a probability and was applied to the best solution of the population during the DE generations. Vakil et al. in [16] proposed a memetic DE with a differential-bidirectional-random-search method as local search operator which was applied to all solutions during the global search process. The algorithm was tested in continuous optimization problems.

B. *Gradient-Based Operators*

Takahama and Sakai in [17] used DE with exponential crossover (known as DE/rand/1/exp) and the ε-constrained method to solve CNOPs. They also adapted a gradient-based mutation as local search mechanism, which was activated for the solutions after applying the crossover operator. The authors presented an improved version based on a new control for the ε-tolerance. In a recent work [18], they proposed two novel mechanisms to control boundary constraints to further improve their approach.

C. *Special Operators*

Liu et al. in [19] proposed a co-evolution-memetic-based algorithm to solve CNOPs. They used two different population within DE. One population aimed to minimize the objective function regardless of constraints, and the other one ensured to minimize the constraints violation regardless of the objective function. The authors used a Gaussian mutation originally adopted in real coded genetic algorithms GAs as local-search-like operator, which was applied to the population when the best solution kept unchanged for several generations. Menchaca and Coello in [20] proposed a hybrid DE algorithm to solve CNOPs. They combined DE with a Nealder-Mead based operator (called simplex operator). Using a frequency, the simplex operator was activated to generate new solutions of the population. Zhao et al. in [21] proposed a MA based on DE as global search and Cauchy distribution used as local search operator which was applied to the best solution at each generation. The algorithm was designed to solve continuous optimization problems. Pescador and Coello in [22] implemented a crossover-memetic-based algorithm to solve CNOPs. They proposed a DE with simplex crossover as local search mechanism. The authors applied the local search on the neighborhood of the best and the worst solutions of the population and was activated at each generation of the DE algorithm. Pan et al. in [23] proposed a DE algorithm with a local search operator based on Cauchy mutation to solve large-scale optimization problems. The local search operator was activated inside the DE generations when the global best solution did not improve on a certain number of generations.

IV. Proposed Approach

Section III included different approaches where DE was combined with a local search operator to solve different type of problems including CNOPs. All studies reported competitive results and good performance of the MAs. However, the performance of the local search operator was not measured so as to get a better understanding about its contribution to the competitive final results. Therefore, this research is focused on measuring the performance of three direct local search operators (Nealder-Mead, Hooke-Jeeves and Hill Climber) added separately into Differential Evolution, with the aim to relate it to the final results obtained by each MA variant in a constrained search space. The goal is to get a better understanding of the type of performance required by a local search operator in presence of constraints.

A. *Algorithmic Coordination*

The interaction between global and local search in this research work is the same for each approach compared. Three issues were considered in the global-local search coordination: (1) exploitation area, (2) application frequency and (3) type of replacement.

1) Exploitation Area: According to Mezura-Montes et al. in [24] the DE variant used in this work (DE/rand/1/bin) has a good performance regarding exploration capabilities in constrained spaces. Due to that, in this proposal the best solution in the population was used to exploit promising areas in the search space by applying it the local search operator while using DE/rand/1/bin as the global search algorithm.

2) Application Frequency: Inspired in [19], [23] the local search operator is activated if the best solution in the population does not improve after T generations. Besides, the algorithm considers a probability ψ given by Eq. (4) to activate the local search operator. Although ψ was designed for a particular combinatorial optimization problem, in previous experiments (not reported here due to space restrictions) it worked well in CNOPs.

$$\psi = 1 - \left| \frac{J_{avg} - J_{best}}{J_{worst} - J_{best}} \right| \tag{4}$$

where J_{best}, J_{worst} and J_{avg} are the best, worst and average of the fitness function values in the population, respectively. According to Neri et al. in [25] ψ is a population diversity index which measures it in terms of fitness. The population has high diversity when $\psi \approx 1$ and low diversity when $\psi \approx 0$. In this work, a higher diversity index value means a higher probability to apply local search.

3) Type of Replacement: In this work, Lamarckian learning is used, i.e., the solution generated by the local search operator (X_{new}) is always kept for the next generation. The algorithm randomly selects a solution of the population (except the solution with the best fitness value) to be replaced by X_{new}. This is because it could be the case that X_{new} is worse than the best solution in the current population.

B. Global Search Algorithm

The MAs proposed in this work are based on DE which was proposed by Storn and Price [26]. According to the algorithmic coordination presented in Section IV-A the global search is described in Algorithm 1, where the gray lines mark the elements to integrate the local search operator within DE/rand/1/bin.

In DE, a solution to the optimization problem is known as a vector $X_{G,i} = (x_{G,i,j}, \ldots, x_{G,i,D})$ where $X_{G,i}$ represents a vector i at generation G, and D is the number of decision variables (i.e. the search space dimensionality, $D = n$). In the same way, a population is represented as $P_G = (X_{G,i}, \ldots, X_{G,Pmax})$ where $Pmax$ is the fixed population size at each generation G. *MaxFEs* is the maximum number of fitness evaluations allowed for the algorithm.

C. Memetic DE and Nealder-Mead (MDE+NM)

In this approach, the Nealder-Mead method [27] was integrated in Algorithm 1 as local search operator. The algorithm needs three user-defined parameters: (1) the expansion factor γ, (2) the contraction factor β and (3) the termination criteria $maxIter$. This method works through expansions and contractions during the main loop, see Eq. (5):

$$X_{new} = \begin{cases} (1+\gamma)X_c - \gamma X_h \text{ if } f(X_r) < f(X_l) \text{ (expansion)} \\ (1-\beta)X_c + \beta X_h \text{ if } f(X_r) \geq f(X_h) \text{ (contraction)} \\ (1+\beta)X_c - \beta X_h \text{ if } f(X_g) < f(X_r) < f(X_h) \text{(contraction)} \\ X_r \text{ , otherwise} \end{cases} \tag{5}$$

where X_h, X_l and X_g are the worst, the best and the next to the worst vector of the simplex population respectively. X_r is the reflected point which is calculated by $X_r = 2X_c - X_h$. Finally X_c is the centroid vector, see Eq. (6).

$$X_c = \frac{1}{D} \sum_{i=1, i \neq h}^{D+1} X_i \tag{6}$$

Algorithm 1 Memetic DE

```
1: Randomly generate an initial population of vectors $P_0 = (X_{0,i}, \ldots, X_{0,Pmax})$
2: Calculate the fitness of each vector in the initial population.
3: Set the $\varepsilon$ value using Eq. (12)
4: repeat
5:   for $i \leftarrow 1, Pmax$ do
6:     Randomly select $r0,r1,r2 \in [1, Pmax]$ and $r0 \neq r1 \neq r2 \neq i$
7:     Randomly select $J_{rand} \in [1, D]$
8:     for $j \leftarrow 1, D$ do
9:       if $rand_j \leq Cr$ Or $j = Jrand$ then
10:        $u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j})$
11:      else
12:        $u_{G,i,j} = x_{G,i,j}$
13:      end if
14:    end for
15:    if $U_{G,i} \leq_\varepsilon X_{G,i}$ using Eq. (10) then
16:      $X_{G+1,i} = U_{G,i}$
17:    else
18:      $X_{G+1,i} = X_{G,i}$
19:    end if
20:  end for
21:  if No improvement counter $\geq T$ then
22:    Reset the no improvement counter
23:    Calculate $\psi$ value using Eq. (4)
24:    if rand(0,1) $\leq \psi$ then
25:      Set $X_{new} \leftarrow$ Local_Search_Operator($X_{G,best}$)
26:      Set $X_{G,rand} \leftarrow X_{new}$ where $X_{G,rand} \neq X_{G,best}$
27:    end if
28:  end if
29:  if There are equality constraints then
30:    Modify $\delta$ using Eq. (13)
31:  end if
32:  Update $\varepsilon$ value using Eq. (11)
33:  $G = G + 1$
34: until MaxFEs is reached
```

where $D + 1$ corresponds to the number of vectors in the simplex array and D is the number of decision variables of each vector, i.e, $D = N$. The whole process is shown in Algorithm 2.

Algorithm 2 Nealder-Mead Local search

```
1: Create an initial simplex applying Eq. (7)
2: repeat
3:   Find $X_h$, $X_l$ and $X_g$
4:   Calculate $X_c$ using Eq. (6)
5:   Calculate the reflected vector $X_r = 2X_c - X_h$
6:   Set $X_{new}$ using Eq. (5)
7:   Set $X_h \leftarrow X_{new}$
8: until MaxIter
9: return $X_l$
```

In this approach the initial simplex is an array of vectors $S = X_0, X_1, \ldots X_N$ generated from a initial vector Xs, see Eq. (7):

$$X_{i,j} = \begin{cases} Xs_j + rand[0,1] \text{ if } rand(0,1) \leq 0.5 \\ Xs_j - rand[0,1] \text{ , otherwise} \end{cases} \tag{7}$$

D. Memetic DE and Hooke-Jeeves (MDE+HJ)

The Hooke-Jeeves method [27] was combined with Algorithm 1 as local search operator. The algorithm works by creating a set of search directions iteratively. Basically, the algorithm needs three user-defined parameters: (1) the variable increments Δ, (2) a step reduction factor $\alpha > 1$ and (3) the termination criteria $maxIter$. Hooke-Jeeves has two types of moves in the search space: the exploratory move and the pattern move.

1) Exploratory move: The current vector is perturbed in positive and negative directions along each variable one at a time and the best vector is recorded, see Algorithm 3, where $X_{(k)}$ is the vector at iteration k and D is the vector dimension.

Algorithm 3 Exploratory Move

```
1:  $X \leftarrow X_{(k)}$
2:  for $i \leftarrow 1, D$ do
3:      Calculate $X \leftarrow (X_i)$, $U \leftarrow (X_i + \Delta)$, $V \leftarrow (X_i - \Delta)$
4:      Set $X_i \leftarrow$ getBest($X$,$U$,$V$)
5:  end for
6:  if The new vector is different than the initial vector then
7:      return success
8:  else
9:      return failure
10: end if
```

2) Pattern move: This movement is applied to the best vector $X_{(k)}$ at iteration k to find a new vector X_p, see Eq. (8).

$$X_p = X_{(k)} + (X_{(k)} - X_{(k-1)}) \tag{8}$$

The complete local search operator is described in Algorithm 4.

Algorithm 4 Hooke-Jeeves Local search

```
1:  Set $X_k$ by exploratory move with $Xs$ using Algorithm 3
2:  repeat
3:      if Success then
4:          repeat
5:              Perform a pattern move using Eq. (8)
6:              Perform an exploratory move with $X_p$ using Algorithm 3.
                Let the result be the new vector.
7:          until New vector is worse than the previous vector
8:      end if
9:      Set $\Delta \leftarrow \Delta/\alpha$
10:     Set $X_{(k+1)}$ by exploratory move with $X_{(k)}$ using Algorithm 3
11: until MaxIter
12: return $X_{(k)}$
```

E. Memetic DE and Hill Climber (MDE+HC)

Hill Climber was coupled with Algorithm 1 as local search operator as well. This method consists of random perturbations of the variables in a vector, if a new vector generated is better than the original vector, it is replaced. The process is repeated until a maximum number of iterations $maxIter$ is reached. It is the only user-defined parameter. The complete process is shown in Algorithm 5, where Xs is the initial vector, k is the number of iteration, D is the vector dimension, i.e., the number of decision variables in the vector. Finally p is the position within the vector (selected randomly) to be modified by a random value $\sigma \in [0, 1]$.

Algorithm 5 Hill Climber Local search

```
1: Set $X_{(k)} \leftarrow Xs$
2: repeat
3:     Select $\sigma \leftarrow$ random(0, 1)
4:     Select $p \leftarrow$ random(1, $D$)
5:     Set $U \leftarrow X_{(k)}$ and $V \leftarrow X_{(k)}$
6:     Calculate $U \leftarrow X_{(k),p} + \sigma$ and $V \leftarrow X_{(k),p} - \sigma$
7:     Set $X_{(k+1)} \leftarrow$ getBest($X_{(k)}$,$U$,$V$)
8: until MaxIter
9: return $X_{(k)}$
```

F. Constraint-Handling

The MAs implemented in this work adopted the ε-constrained method proposed in [6] which transforms a CNOP into an unconstrained optimization problem. The constraint violation ϕ of a given vector $X_{G,k}$ is computed as the sum of the amounts of all violated constraints, see Eq. (9).

$$\phi(X_{G,k}) = \sum_{i=1}^{m} max(0, g_i(X_{G,k})) + \sum_{j=1}^{p} max(0, |h_j(X_{G,k})| - \delta) \tag{9}$$

The ε tolerance allows the so-called $\leq_\varepsilon$ comparison between two vectors by using only their fitness function values, see Eq. (10).

$$X_{G,k} \leq_\varepsilon X_{G,l} \Leftrightarrow \begin{cases} f(X_{G,k}) \leq f(X_{G,l}) & \text{if } \phi(X_{G,k}), \phi(X_{G,l}) \leq \varepsilon \\ f(X_{G,k}) \leq f(X_{G,l}) & \text{if } \phi(X_{G,k}) = \phi(X_{G,l}) \\ \phi(X_{G,k}) \leq \phi(X_{G,l}) & \text{,otherwise} \end{cases} \tag{10}$$

The ε level is dynamically decreased at each generation as indicated in Eq. (11).

$$\varepsilon(G) = \begin{cases} \varepsilon(0)(1 - \frac{G}{Gc})^{cp} & , 0 < G < Gc \\ 0 & , G \geq Gc \end{cases} \tag{11}$$

where cp is a user-defined parameter to control the reduction speed of the ε tolerance, G is the current generation number and Gc is the generation number when the ε value is set to zero. The initial ε value (i.e., $\varepsilon(0)$) is the constraint violation of the θth solution in the initial population, see Eq. (12).

$$\varepsilon(0) = \phi(X_\theta) \tag{12}$$

G. Tolerance Value for Equality Constraints

In this work, the equality constraints are transformed into inequality constraints using a δ tolerance which decreases at each iteration as indicated in Eq. 13 until a suitable value is reached.

$$\delta(G+1) = \frac{\delta(G)}{dec} \tag{13}$$

where dec is a user-defined parameter which determines how fast the δ value decreases over time and $\delta(0)$ is also a user-defined parameter.

V. Local Search Performance Measure

The improvement Index (*I.I.*) proposed by Barkat et al. in [28] is adapted in this work to measure the local search performance. *I.I.* indicates the rate of fitness improvement made by a particular local search operator. The value of *I.I.* is restricted in the range $-1 \leq I.I. \leq +1$, the values outside these ranges are bounded to the limits permitted. When the local search operator starts with an infeasible vector and becomes feasible after the application, the *I.I.* gets the value +1. If the situation is the opposite, i.e., from feasible to infeasible then *I.I.* is assigned -1. However, if the vector remains feasible before and after the local search operator, *I.I.* is calculated by Eq. (14)

$$I.I. = \frac{f(X)_{before} - f(X)_{after}}{f(X)_{before}} \tag{14}$$

where $f(X)_{before}$ and $f(X)_{after}$ are the fitness value of the vector X before and after the application of the local search operator respectively. In the same way, if the local search operator starts with an infeasible vector and still results an infeasible one, *I.I.* is obtained by Eq. (15)

$$I.I. = \frac{\phi(X)_{before} - \phi(X)_{after}}{\phi(X)_{before}} \tag{15}$$

where $\phi(X)_{before}$ and $\phi(X)_{after}$ are the constraint violation of the vector X before and after the application of the local search operator, respectively. A positive value of *I.I.* indicates that fitness was improved by the local search operator. On the other hand, a negative value indicates a fitness decrease.

VI. EXPERIMENTS AND RESULTS

The experiments are divided in two phases: (1) The MAs mentioned in Sections IV-C, IV-D and IV-E are tested on 18 benchmark problems, with different search space dimensionality (10 and 30 dimensions), used in the special session on "Single Objective Constrained Real-Parameter Optimization" in CEC'2010 [29], to analyze the performance of each local search using the Improvement Index (*I.I*) measure, and (2) the final results obtained by each MA are analyzed and are also compared against those obtained by the εDEga [18], the most competitive approach in the aforementioned special session. The parameter values used for each algorithm are described in Table I. The dimensionalities used in the test problems were $D = 10$ and $D = 30$, as defined in the special session. Likewise, the maximum number of fitness evaluations *maxFEs* (200,000 and 600,000 respectively). The parameters marked with (*) in Table I were fine-tuned by using the IRACE tool [30] with four representative test problems as a training set for the parameter tuning processes, the parameters suggested in [18] and [27] were used as the starting point for such process.

The results of the first experiment (analyzing the performance of local search using the Improvement Index (*I.I*) measure) are presented in Figures 1 and 2. In all cases the 95%-confidence Wilcoxon test showed significant differences with the exception of MDE-HJ and MDE-HC in 10D problems. The results suggest that MDE+HC has a better overall average performance in most problems with respect to MDE+NM (10D and 30D) and MDE+HJ (only in 30D). Only in test problem C13 for 10D, MDE+HC had a negative *I.I.* average performance. On the other hand, MDE+NM obtained positive I.I. values for 10D in only four test problems (C04, C05, C06 and C12) and was clearly outperformed by MDE+HC. It is important to remark MDE+NM's clear improvement from 10D to 30D (most negative I.I. values in 10D and most positive I.I.values in 30D). In contrast, MDE+HJ shows an opposite behavior, i.e. most positive I.I. values in 10D but most negative I.I. values in 30D.

TABLE I. PARAMETERS VALUES. (*) INDICATES THE VALUES OBTAINED USING IRACE.

Algorithm	**Parameter**	**Value**	
		10 D	30 D
DE	*maxFEs*	$2.0E+05$	$6.0E+05$
	Pmax	20	50
	**Cr*	0.7890	0.8830
	**F*	0.8587	0.9989
	T	D	
Nealder-Mead	*β	0.7683	
	*γ	1.2030	
	maxIter	D×20	
Hooke-Jeeves	*Δ	0.6986	
	*α	1.0001	
	maxIter	D×2	
Hill Climber	*maxIter*	D×50	
Epsilon-Constraint	θ	0.2	
	cp	46	
	Gc	1600	
Equality Constraint Tolerance	δ	1.8E+100	
	dec	1.1002	

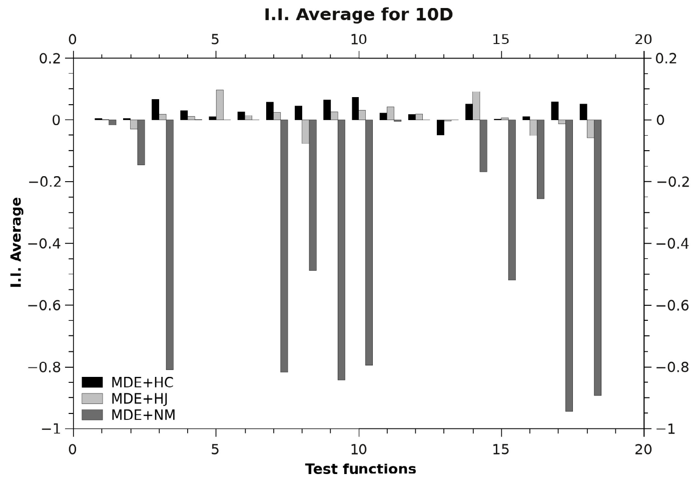

Fig. 1. Average of improvement index (*I.I*) measure for 10D problems

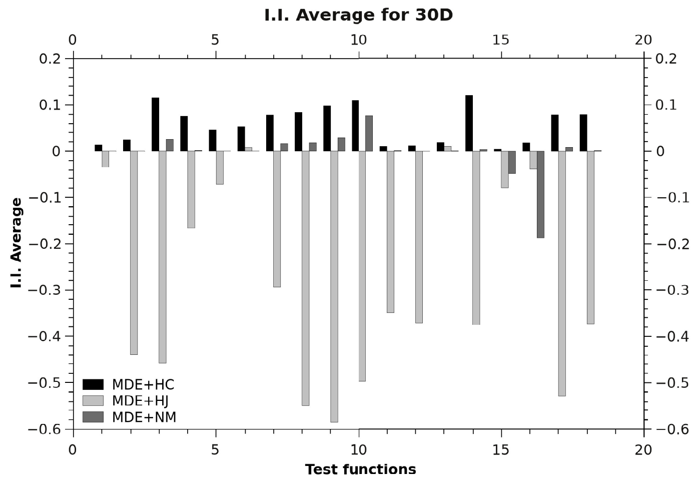

Fig. 2. Average of improvement index (*I.I*) measure for 30D problems

The results of the second experiment are summarized in Table III, where the best and average function values for each MA, besides the standard deviation values, are shown for 10D and 30D. The aforementioned results suggest that MDE+NM, MDE+HJ and MDE+HC are able to find feasible solutions for all test problems in 10D and 30D.

The results of the Wilcoxon test applied to the best and average overall results between εDEga vs MDE+NM, εDEga vs MDE+HJ and εDEga vs MDE+HC are presented in Table II. It can be noted that no significant differences were obtained, i.e. the performance of the three MAs with local search operators based on direct search methods is similar to that observed by a MA with gradient based local search. This finding, besides the one that feasible solutions were found by the three MAs in all test problems, mean that the algorithm coordination proposed in this research and mentioned in Section IV-A works favorably to solve CNOPs.

Regarding comparisons in particular test problems, MDE+NM reaches the optimal function values obtained by εDEga for 10D in nine test problems (C01, C03, C07, C08, C09, C10, C13, C14 and C16). Furthermore, in functions C02, C04, C05, C06, C11, and C17 MDE+NM outperforms εDEga. For 30D, in seven test problems (C02, C04, C05, C11, C12, C17 and C18) MDE+NM gets better solutions than εDEga. Regarding MDE+HJ, it was able to outperform εDEga for 10D in seven test problems (C02, C04, C05, C06, C11, C12 and C17), and in C03, C07, C08, C09 C10, C13, C14 and C16 the optimal functions values are reached. For 30D, MDE+HJ outperform εDEga in ten test problems (C02, C03, C04, C05, C06, C10, C11, C12, C17 and C18). On the other hand, MDE+HC was able to outperform εDEga in eight test problems (C02, C04, C05, C06, C11, C12, C17 and C18) for 10D and 30D, adding C03 and C10 to the latter. However, unlike MDE+NM and MDE+HJ, MDE+HC reaches only two optimal function values for 10D (C03 and C09).

Finally, based on the 95%-confidence Wilcoxon test, no significant differences among the three MAs studied in this paper were obained.

TABLE II. 95%-CONFIDENCE STATISTICAL TEST RESULTS BETWEEN εDEGA AND EACH MA COMPARED FOR 10D AND 30D TEST PROBLEMS. (Y) MEANS SIGNIFICANT DIFFERENCE AND (N) MEANS NO SIGNIFICANT DIFFERENCE.

Comparison		Wilcoxon Test Results	
$p \leq 0.05$.		10 D	30 D
	Best	N	N
εDEga vs MDE+NM	Average	N	N
	Best	N	N
εDEga vs MDE+HJ	Average	N	N
	Best	N	N
εDEga vs MDE+HC	Average	N	N

Comparing the results of experiments 1 and 2, it can be noted that MDE+HC had the best performance based on *I.I.* for 10D and 30D test problems (Figures 1 and 2). However, the final results in Table III show that MDE+HC provided a similar performance with respect to MDE+NM and MDE+HJ.

On the other hand, MDE+NM had the worst performance in 10D problems based on the I.I. measure (see Figure 1), but obtained very competitive final results in the same 10D test problems (Table III). MDE+NM improved its performance for 30D based on the I.I. measure values (see Figure 2), but such values were not better than those presented by MDE+HC. Nevertheless, the final results obtained by MDE+NM in 30D (Table III) were similar than those obtained by MDE+HC.

Finally, MDE+HJ, which obtained similar results (based on the Wilcoxon test) with respect to MDE+HC based in the I.I. values for 10D in Figure 1, obtained similar final results in such dimension in Table III. A different behavior was observed in 30D (Figure 2 and Table III), where most negative I.I. values were obtained by MDE+HJ compared with the most positive I.I. values by MDE+HC, but similar final results were obtained by MDE+HJ and MDE+HC.

The results of both experiments imply that the positive effect of a local search operator in a constrained search space can not be only measured by the isolated improvement of a single solution. A possible way to deal with this issue is considering such improvement but with respect to the improvement of the whole population.

VII. Conclusions and Future Work

This paper analyzed the relationship between the performance of the local search operator within a Memetic Algorithm and its final results in CNOPs by adapting an improvement index measure, which indicates the rate of fitness improvement made by the local search operator. To perform this analysis, adaptations of Nealder-Mead, Hooke-Jeeves and Hill Climber algorithms were used as local search operators separately within a Memetic DE-based structure, where the best solution in the population was used to exploit promising areas in the search space by the aforementioned local search operators. The ε-constrained method was used as a constraint-handling technique. The algorithms solved eighteen benchmark problems in 10D and 30D (thirty six total test problems). Two experiments were carried out, one centered on measuring the performance of the local search operators and other focused on discussing the final results reached by each MA. The results obtained confirmed that the algorithm coordination proposed in this work is suitable to get competitive MAs to solve CNOPs with local search operators based on direct methods. The results also suggested that a poor value of the improvement index measure does not necessarily reflects on also poor final results obtained by the MA in a constrained search space. Therefore, other aspects such as the improvement of the local search operator with respect to the improvement of the global search must be considered and analyzed. This is precisely the initial topic of future research.

Acknowledgment

The authors acknowledge support from MinCyT bi-lateral project MX1103 and CONACyT bi-lateral project No. 164626. The first author acknowledges support from CONACYT and the University of Veracruz to pursue graduate studies.

References

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[2] J. Kennedy and R. C. Eberhart, *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[3] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, pp. 173–194, 2011.

TABLE III. FUNCTION VALUES ACHIEVED BY MDE+NM, MDE+HJ AND MDE+HC FOR 10D-30D AND COMPARISON AGAINST εDEGA. BOLDFACE REMARKS THOSE BETTER RESULTS AND ITALIC TEXT REMARKS THOSE SIMILAR RESULTS.

Function	MA	10D			30D		
		Best	**Avg**	**Std**	**Best**	**Avg**	**Std**
	εDEga	*-7.473104E-01*	**-7.470402E-01**	**1.323339E-03**	**-8.218255E-01**	**-8.208687E-01**	**7.103893E-04**
C01	MDE+NM	*-7.473104E-01*	-7.376742E-01	1.749557E-02	-8.218000E-01	-8.054422E-01	1.333897E-02
	MDE+HJ	-7.473103E-01	-7.421135E-01	6.315751E-03	-8.217591E-01	-7.998397E-01	2.413851E-02
	MDE+HC	-7.473102E-01	-7.340776E-01	1.360141E-02	-8.178500E-01	-7.815536E-01	2.848675E-02
	εDEga	-2.277702E+00	-2.258870E+00	2.389779E-02	-2.169248E+00	-2.151424E+00	1.197582E-02
C02	MDE+NM	*-3.166209E+00*	-7.253218E-01	2.475302E+00	-3.001695E+00	1.490355E-01	2.901242E+00
	MDE+HJ	*-3.166209E+00*	-3.140613E+00	6.333234E-02	-2.178478E+00	-2.071487E+00	8.594076E-02
	MDE+HC	*-3.166209E+00*	**-3.165129E+00**	**1.947029E-03**	**-3.157959E+00**	**-3.155341E+00**	**1.843985E-03**
	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	2.867347E+01	**2.883785E+01**	**8.047159E-01**
C03	MDE+NM	*0.000000E+00*	4.783895E-01	1.295485E+00	4.018871E+01	1.055579E+03	1.219142E+03
	MDE+HJ	*0.000000E+00*	1.265642E+00	3.983805E+00	2.338906E+01	8.851698E+02	1.734344E+03
	MDE+HC	*0.000000E+00*	5.545954E+01	1.862923E+02	**2.867076E-02**	7.440057E+01	1.639822E+02
	εDEga	-9.992345E-06	-9.918452E-06	1.546730E-07	4.698111E-03	8.162973E-03	3.067785E-03
C04	MDE+NM	-4.441070E+01	-4.441070E+01	**7.105427E-15**	*-4.086475E+01*	*-4.086475E+01*	*2.842171E-14*
	MDE+HJ	**-5.162324E+01**	**-4.501714E+01**	1.574523E+00	*-4.086475E+01*	*-4.086475E+01*	1.852869E-14
	MDE+HC	-4.441070E+01	-4.441070E+01	7.519677E-15	*-4.086475E+01*	*-4.086475E+01*	*2.842171E-14*
	εDEga	-4.836106E+02	-4.836106E+02	3.890350E-13	-4.531307E+02	-4.495460E+02	2.899105E+00
C05	MDE+NM	-6.074970E+02	-6.074970E+02	*2.273737E-13*	-6.038261E+02	*-6.038261E+02*	*3.410605E-13*
	MDE+HJ	**-6.187782E+02**	**-6.087846E+02**	3.260816E+00	**-6.046462E+02**	-6.038602E+02	1.638842E-01
	MDE+HC	-6.074970E+02	-6.074970E+02	*2.273737E-13*	-6.038261E+02	*-6.038261E+02*	*3.410605E-13*
	εDEga	-5.786581E+02	-5.786528E+02	3.627169E-03	-5.285750E+02	-5.279068E+02	4.748378E-01
C06	MDE+NM	-5.908091E+02	-5.908091E+02	*2.273737E-13*	*-5.908091E+02*	*-5.908091E+02*	2.501110E-13
	MDE+HJ	**-5.965922E+02**	**-5.914438E+02**	1.404629E+00	*-5.908091E+02*	*-5.908091E+02*	*2.273737E-13*
	MDE+HC	-5.908091E+02	-5.908091E+02	*2.273737E-13*	*-5.908091E+02*	*-5.908091E+02*	*2.273737E-13*
	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	**1.147112E-15**	**2.603632E-15**	**1.233430E-15**
C07	MDE+NM	*0.000000E+00*	6.378527E-01	1.461504E+00	1.104374E+01	7.450512E+01	4.632270E+01
	MDE+HJ	*0.000000E+00*	3.543983E+01	1.153287E+02	1.958496E+01	1.319716E+02	2.140775E+02
	MDE+HC	2.335028E-27	1.140278E+01	3.160441E+01	5.484654E-04	3.613771E+01	6.136745E+01
	εDEga	*0.000000E+00*	**6.727528E+00**	**5.560648E+00**	**2.518693E-14**	**7.831464E-14**	**4.855177E-14**
C08	MDE+NM	*0.000000E+00*	9.743229E+01	2.368199E+02	2.054448E+01	3.295516E+02	4.599177E+02
	MDE+HJ	*0.000000E+00*	1.872247E+02	4.925612E+02	2.580141E+01	2.406614E+03	3.473454E+03
	MDE+HC	2.886723E-26	2.083554E+02	4.002769E+02	1.031884E-03	7.015442E+02	2.623504E+03
	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	**2.770665E-16**	**1.072140E+01**	**2.821923E+01**
C09	MDE+NM	*0.000000E+00*	3.189263E-01	1.081532E+00	3.076869E+01	1.300074E+05	5.463461E+05
	MDE+HJ	*0.000000E+00*	5.538516E+03	2.675761E+04	2.645223E+01	7.286739E+02	9.047144E+02
	MDE+HC	*0.000000E+00*	2.894271E+01	8.247018E+01	2.391410E-02	2.178574E+02	3.038142E+02
	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	3.252002E+01	**3.326175E+01**	**4.545577E-01**
C10	MDE+NM	*0.000000E+00*	7.973158E-01	1.594632E+00	3.297857E+01	1.319775E+04	4.578728E+04
	MDE+HJ	*0.000000E+00*	5.444295E+02	2.593653E+03	2.352233E+01	1.151545E+03	3.730467E+03
	MDE+HC	1.910937E-26	3.539641E+01	1.016156E+02	**1.229880E-02**	1.024342E+02	1.922225E+02
	εDEga	-1.522713E-03	-1.522713E-03	**6.341035E-11**	-3.268462E-04	-2.863882E-04	**2.707605E-05**
C11	MDE+NM	-7.164915E+01	-5.657112E+01	1.730193E+01	-6.043522E+01	**-5.458203E+01**	3.540663E+00
	MDE+HJ	-6.579135E+01	-5.585565E+01	5.458492E+00	**-6.044491E+01**	-5.048541E+01	4.155694E+00
	MDE+HC	**-7.205104E+01**	**-6.013852E+01**	5.316331E+00	-5.535454E+01	-5.016513E+01	3.314065E+00
	εDEga	-5.700899E+02	-3.367349E+02	**1.782166E+02**	-1.991453E-01	3.562330E+02	**2.889253E+02**
C12	MDE+NM	-8.108057E+03	**-7.932093E+03**	2.062272E+02	**-2.680487E+04**	**-2.575254E+04**	4.882107E+02
	MDE+HJ	**-8.158429E+03**	-7.781861E+03	3.168890E+02	-2.595765E+04	-2.423713E+04	8.386515E+02
	MDE+HC	-8.108057E+03	-7.683042E+03	3.521029E+02	-2.620216E+04	-2.425797E+04	1.106711E+03
	εDEga	*-6.842937E+01*	**-6.842936E+01**	**1.025960E-06**	**-6.642473E+01**	**-6.535310E+01**	**5.733005E-01**
C13	MDE+NM	*-6.842937E+01*	-6.365608E+01	2.530104E+00	-6.582004E+01	-6.406239E+01	1.114729E+00
	MDE+HJ	*-6.842937E+01*	-6.222556E+01	3.593505E+00	-6.431489E+01	-5.933518E+01	2.580831E+00
	MDE+HC	-6.842936E+01	-6.391025E+01	3.071028E+00	-6.432779E+01	-6.059491E+01	2.539741E+00
	εDEga	*0.000000E+00*	0.000000E+00	**0.000000E+00**	**5.015863E-14**	**3.089407E-13**	**5.608409E-13**
C14	MDE+NM	*0.000000E+00*	**4.411768E+06**	1.092644E+07	4.444056E+04	1.661360E+09	4.741723E+09
	MDE+HJ	*0.000000E+00*	6.598579E+06	1.407676E+07	2.305974E+01	1.018126E+04	2.453777E+04
	MDE+HC	7.187323E-11	1.375070E+11	5.419730E+11	2.797877E-03	5.547401E+07	2.677654E+08
	εDEga	**0.000000E+00**	1.798978E-01	**8.813156E-01**	**2.160345E+01**	**2.160376E+01**	**1.104834E-04**
C15	MDE+NM	1.922937E+12	**8.456202E+13**	7.011401E+13	5.919372E+13	1.932704E+14	6.761978E+13
	MDE+HJ	8.967169E+12	8.884936E+13	7.827173E+13	1.200730E+14	2.396448E+14	6.652184E+13
	MDE+HC	4.017477E+12	8.161327E+13	5.158264E+13	1.269401E+14	1.989160E+14	4.724277E+13
	εDEga	*0.000000E+00*	3.702054E-01	3.710479E-01	**0.000000E+00**	**2.168404E-21**	**1.062297E-20**
C16	MDE+NM	*0.000000E+00*	**1.808242E-01**	3.367572E-01	1.549673E-05	5.951875E-02	2.012325E-01
	MDE+HJ	*0.000000E+00*	8.004045E-02	1.695592E-01	1.461790E-06	1.351913E-02	5.601275E-02
	MDE+HC	9.864672E-03	4.208551E-02	**2.854565E-02**	1.687228E-11	3.036562E-02	4.586211E-02
	εDEga	1.463180E-17	1.249561E-01	1.937197E-01	2.165719E-01	6.326487E+00	4.986691E+00
C17	MDE+NM	*0.000000E+00*	**4.719779E-29**	**1.640253E-28**	8.102202E-02	1.881183E+01	2.806569E+01
	MDE+HJ	*0.000000E+00*	9.972161E-02	3.219155E-01	2.330202E-02	3.317397E-01	3.520637E-01
	MDE+HC	1.669513E-27	1.847996E-08	6.937622E-08	**1.132108E-06**	**4.067197E-03**	**8.070527E-03**
	εDEga	3.731439E-20	**9.678765E-19**	**1.811234E-18**	1.226054E+00	8.754569E+01	1.664753E+02
C18	MDE+NM	1.104539E+02	4.661089E+03	5.214531E+03	1.047617E-02	3.358028E+02	6.557928E+02
	MDE+HJ	2.088774E-07	1.083765E+02	1.802146E+02	1.769504E-02	**6.777942E-01**	**1.014288E+00**
	MDE+HC	**8.293069E-22**	9.905441E-01	4.852655E+00	**7.578104E-04**	1.083834E+00	2.901766E+00

[4] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review." *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[5] F. Neri and V. Tirronen, "On memetic differential evolution frameworks: A study of advantages and limitations in hybridization," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, June 2008, pp. 2135–2142.

[6] T. Takahama, S. Sakai, and N. Iwane, "Constrained optimization by the constrained hybrid algorithm of particle swarm optimization and genetic algorithm," in *AI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, S. Zhang and R. Jarvis, Eds. Springer Berlin Heidelberg, 2005, vol. 3809, pp. 389–400.

[7] T. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 3, pp. 284 –294, sep 2000.

[8] S. Muelas, A. La Torre, and J.-M. Peña, "A memetic differential evolution algorithm for continuous optimization," in *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, ser. ISDA '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1080–1084.

[9] A. Mandal, A. Das, P. Mukherjee, S. Das, and P. Suganthan, "Modified differential evolution with local search algorithm for real world optimization," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, june 2011, pp. 1565 –1572.

[10] F. Solis and R. Wets, "Minimization by random search techniques," *Math. Oper. Res.*, vol. 6, no. 1, pp. 19–30, 1981.

[11] S. Hernández, G. Leguizamón, and E. Mezura-Montes, "Hibridación de evolución diferencial utilizando hill climbing para resolver problemas de optimización con restricciones," in *XVIII Congreso Argentino de Ciencias de la Computación*, Oct 2012, pp. 1–10.

[12] S. Hernandez, G. Leguizamon, and E. Mezura-Montes, "A hybrid version of differential evolution with two differential mutation operators applied by stages," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, pp. 2895–2901.

[13] S. Dominguez-Isidro, E. Mezura-Montes, and G. Leguizamon, "Memetic differential evolution for constrained numerical optimization problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, pp. 2996–3003.

[14] C. Zhang, J. Chen, and B. Xin, "Distributed memetic differential evolution with the synergy of lamarckian and baldwinian learning," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2947–2959, May 2013.

[15] A. P. Piotrowski, "Adaptive memetic differential evolution with global and local neighborhood-based mutation operators," *Inf. Sci.*, vol. 241, pp. 164–194, Aug. 2013.

[16] M. Vakil-Baghmisheh and M. Ahandani, "A differential memetic algorithm," *Artificial Intelligence Review*, vol. 41, no. 1, pp. 129–146, 2014.

[17] T. Takahama and S. Sakai, "Constrained optimization by the constrained differential evolution with gradient-based mutation and feasible elites," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 0-0 2006, pp. 1 –8.

[18] ——, "Constrained optimization by the epsilon constrained differential evolution with an archive and gradient-based mutation," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, july 2010, pp. 1 –9.

[19] B. Liu, H. Ma, X. Zhang, and Y. Zhou, "A memetic co-evolutionary differential evolution algorithm for constrained optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, sept. 2007, pp. 2996 –3002.

[20] A. Menchaca-Mendez and C. A. Coello Coello, "A new proposal to hybridize the nelder-mead method to a differential evolution algorithm for constrained optimization," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, May, pp. 2598–2605.

[21] M. H. Fuqing Zhao and W. Ma, "A memetic differential evolution algorithm with adaptive mutation operator," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, no. 19, pp. 3687–3691, 2012.

[22] M. Pescador Rojas and C. A. Coello Coello, "A memetic algorithm with simplex crossover for solving constrained optimization problems," in *World Automation Congress (WAC), 2012*, june 2012, pp. 1 –6.

[23] Y. Z. Xiuqin Pan and X. Xu, "Adaptive differential evolution with local search for solving large-scale optimization problems," *Journal of Information and Computational Science*, vol. 9, no. 2, pp. 489–496, 2012.

[24] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: An empirical study," *Inf. Sci.*, vol. 180, no. 22, pp. 4223–4262, Nov. 2010.

[25] F. Neri, J. Toivanen, G. Cascella, and Y.-S. Ong, "An adaptive multimeme algorithm for designing hiv multidrug therapies," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 4, no. 2, pp. 264–278, April 2007.

[26] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[27] K. Deb, *Optimization for Engineering Design Algorithms and Examples*, fisrt edition ed. Prentice-Hall of India, 1995.

[28] A. Ullah, R. Sarker, D. Cornforth, and C. Lokan, "An agent-based memetic algorithm (ama) for solving constrained optimazation problems," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, Sept 2007, pp. 999–1006.

[29] R. Mallipeddi and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2010.

[30] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011. [Online]. Available: http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2011-004.pdf