Interval Differential Evolution Using Structural Information of Global Optimization Problems*

Mariane R. Sponchiado Cassenote[0000−0003−0489−223X], Guilherme A. Derenievicz[0000−0002−3970−1766], and Fabiano Silva[0000−0001−5453−6175]

Informatics Department, Federal University of Paraná, Curitiba, Brazil
{mrscassenote,gaderenievicz,fabiano}@inf.ufpr.br

Abstract. Differential Evolution (DE) algorithms are a promising strategy to Numerical Constrained Global Optimization Problems (NCOP). Most DE recent variants are applied to black-box optimization problems, where the analytical structure of the NCOP instance is unknown. In this paper we present an Interval Differential Evolution (InDE) algorithm that explores the structural information of the problem. The instance structure is represented by a hypergraph Epiphytic decomposition, where the variables are intervals. InDE algorithm is based on several strategies used in state-of-the-art DE implementations. Based on strutural information, our approach extracts a subset of variables of the instance that are critical to the search process. The DE population individuals encode only this subset of variables. The other variables of the instance are valuated by a linear cost constraint propagation over the hypergraph structure. Our experiments show that the use of structural information associated with interval local consistency techniques significantly improves the performance of DE algorithm.

Keywords: Structural Decomposition · Differential Evolution · Global Optimization.

1 Introduction

A *Numerical Constrained Global Optimization Problem* (NCOP) consists of finding an assignment of values to a set of variables $\mathcal{V} = \{x_1, \ldots, x_D\}$ that minimizes an objective function $f : \mathbb{R}^D \mapsto \mathbb{R}$ subject to a set of m constraints of the form $g_i(x_{i_1}, \ldots, x_{i_k}) \leq 0$ called *constraint network*, where $\{x_{i_1}, \ldots, x_{i_k}\} \subseteq \mathcal{V}$ is the *scope* of the i-th constraint, $1 \leq i \leq m$, and D is the *dimension* of the instance.

Differential Evolution (DE) [12] has become one of the most used evolutionary algorithms to tackle NCOPs due to its performance in recent competitions of the IEEE Congress on Evolutionary Computing (CEC), winning the last editions of the constrained real-parameter optimization track. In CEC competitions, the state-of-the-art algorithms are evaluated in the context of black-box optimization, where the analytical structure of the NCOP instance is unknown.

* This work was supported by CAPES – Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil.

On the other hand, interval branch and pruning algorithms have been widely used in the last few decades to rigorously solve NCOPs [1, 6, 7]. These algorithms combine local consistency techniques with a complete investigation of the search space. In many approaches, the local consistency is based on the analytical structure of the NCOP instance [5]. In this context, the *Relaxed Global Optimization* (OGRe) solver, proposed in [3], is an interval method that explores the structural information represented by a hypergraph. The algorithm uses the constraint hypergraph to determine the amount of local consistency that guarantees a solution to be found efficiently. This is ensured by the existence of a particular decomposition of the hypergraph, called Epiphytic decomposition, that extracts a subset of variables of the instance that are critical to the search method. The main issue of the solver is that the search is performed over a relaxed version of the instance, where a tolerance $\varepsilon > 0$ is considered to satisfy some constraints of the problem (with ε set empirically). Another issue is the high computational cost of the branch and pruning approach.

In this paper, we propose an *Interval DE* (InDE) algorithm that uses the Epiphytic decomposition as a source of structural information of the NCOP instance. Such decomposition allows the DE to perform its operations only on a subset of the instance variables, whilst others are valuated by constraint propagation. At each fitness evaluation, a local consistency procedure allows the pruning of unfeasible values from the intervals assigned to that subset of variables. Differently from OGRe, the proposed algorithm solves the original instance by dynamically defining the ε tolerance during the search. The processing time is limited by the maximum number of fitness evaluations, given as input.

The proposed InDE algorithm is based on several strategies used in state-of-the-art algorithms of the CEC competitions, like IUDE [15] and LSHADE44 [10]. We evaluate our approach in an experimental set of 80 instances from the COCONUT Benchmark [11]. Experiments show that the use of structural information associated with interval consistency techniques significantly improves the performance of the DE algorithm. In addition, we find solutions with better approximations than the OGRe solver.

The remainder of this paper is organized as follows: section 2 contains the main concepts about interval algorithms for global optimization, and the approach used in OGRe [3]. Section 3 presents the classical DE algorithm. Section 4 introduces our InDE approach. Experiments and discussions are presented in section 5, and section 6 concludes the paper.

2 Interval Algorithms for Global Optimization

Interval arithmetic was proposed in the 60's as a formal approach to analyze rounding errors on computational systems [9]. A *closed interval* $X = [\underline{x}, \overline{x}]$ is the set of real numbers $X = \{x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x}\}$, where $\underline{x}, \overline{x} \in \mathbb{R}$ are the *endpoints* of X. The *width* of an interval X is $\omega(X) = \overline{x} - \underline{x}$ and its *midpoint* is $\mu(X) = (\underline{x} + \overline{x})/2$. An interval can be defined by its width and midpoint as follows: $X = [\mu(X) - \omega(X)/2,\ \mu(X) + \omega(X)/2]$. A *box* is a tuple of intervals

$(X_1, X_2, \ldots, X_n)$. Given two intervals X and Y the *interval extension* of any binary operator $\circ$ well defined in $\mathbb{R}$ is defined by:

$$X \circ Y = \{x \circ y \mid x \in X,\ y \in Y \text{ and } x \circ y \text{ is defined in } \mathbb{R}\}.$$

The set $X \circ Y$ can be an interval, the empty set or a disconnected set of real numbers (a *multi-interval*). It is possible to compute $X \circ Y$ for all algebraic and the most common transcendental functions only by analyzing the endpoints of X and Y [9, 6], e.g., $[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$, $[\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] = [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}]$ etc.

Since the late eighties, interval arithmetic has been used to deal with numerical constrained problems in a rigorous way by combining consistency techniques from the constraint programming community with *Branch and Bound* (B&B) algorithms [1, 6, 7]. A central work in this context is the Numerica modeling language [16], whose core algorithm forms the basis of most current interval solvers. Interval methods compute a set of atomic boxes that contains an optimum solution of the NCOP instance. Generally, such methods are composed by three main phases that are repeatedly applied until a solution is found:

- **Pruning phase**: contractors that guarantee some level of local consistency are applied, removing from variables' domain values that surely do not constitute a feasible solution of the problem. On numeric context, these contractors are generally implemented using interval arithmetic;
- **Local search**: cheap search methods are used to detect if the current box contains a feasible solution of the instance that can be used as an upper bound for the optimum solution. The Newton method is an example of a local search method that can be applied in this phase;
- **Branching phase**: a variable is chosen and its domain is bisected to continue the search in a recursive fashion. Good heuristics for choosing the variable and the side of its domain to be searched first are essential for the performance of the method.

In the pruning phase, contractors of Generalized Arc-Consistency (GAC) and its relaxed forms Hull Consistency and Box Consistency are generally used. GAC [8] guarantees that any valuation of a single variable can be extended to others variables whilst satisfying a constraint. On ternary numerical constraints of the form $x_1 = x_2 \circ_1 x_3$ this property is ensured by (1), where X_i is the domain of x_i, and $\circ_2$ and $\circ_3$ are the inverse operations of $\circ_1$ that hold the condition $(x_1 = x_2 \circ_1 x_3) \Leftrightarrow (x_2 = x_1 \circ_2 x_3) \Leftrightarrow (x_3 = x_1 \circ_3 x_2)$:

$$(X_1 \subseteq X_2 \circ_1 X_3) \wedge (X_2 \subseteq X_1 \circ_2 X_3) \wedge (X_3 \subseteq X_1 \circ_3 X_2). \tag{1}$$

If a ternary constraint is not GAC, this propriety can be easily achieved by computing the intersection of each relevant domain with the respective *projection function*:

$$\texttt{GAC\_contractor}(x_1 = x_2 \circ_1 x_3) := \begin{cases} X_1 \leftarrow X_1 \cap (X_2 \circ_1 X_3) \\ X_2 \leftarrow X_2 \cap (X_1 \circ_2 X_3) \\ X_3 \leftarrow X_3 \cap (X_1 \circ_3 X_2). \end{cases} \tag{2}$$

Strictly, the box (X_1, X_2, X_3) generated by (2) is not complete due to the finite precision of machine numbers. Furthermore, domains may be disconnected sets of real numbers (multi-intervals). Generally, in order to apply the GAC contractor on k-ary constraints a procedure of decomposition that transforms the constraint into an equivalent set of ternary constraints is used [4]. For example, with an auxiliary variable z_1 the constraint $x_1(x_2 - x_1) = 0$ is decomposed into a network of two new constraints: $x_2 - x_1 = z_1$ and $x_1 z_1 = 0$. Contractor (2) is then repeatedly applied to both constraints until a fixed box is obtained or a maximum number of steps is reached.

In the constraint programming context, a *constraint hypergraph* $\mathcal{H} = (\mathcal{V}, E)$ is a structural representation of a constraint network where $\mathcal{V}$ is a set of vertices representing variables and E is a set of edges representing scopes of constraints. In [3] this structural representation is extended to NCOP instances by encoding the objective function $f(x_1, \ldots, x_D)$ as a new constraint $z = f(x_1, \ldots, x_D)$ on the network, where z is called the *root* variable of the network and its domain Z equals the image of f under the box $(X_1, \ldots, X_D)$. The authors found a relationship between the constraint hypergraph and the amount of consistency that guarantees a backtrack-free solution, i.e., a search that solves the NCOP instance without encountering any conflict. This relationship is ensured by the existence of a particular decomposition of the constraint hypergraph called Epiphytic decomposition.

An *Epiphytic decomposition* [3] of the constraint hypergraph $\mathcal{H} = (\mathcal{V}, E)$ according to $z \in \mathcal{V}$ is a triple $(\mathcal{A}, \Omega, t)$, where $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is an ordered set of disjointed hypergraphs $\mathcal{A}_i = (V_i, E_i)$ obtained by removing from $\mathcal{H}$ a set of edges Ω, and $t : \Omega \mapsto V_\Omega$ is a function that associates each edge $e_i \in \Omega$ with one vertex $t(e_i) \in e_i$ such that: (i) $\mathcal{A}_1$ is a rooted tree in z (i.e., connected and Berge-acyclic); (ii) $\forall \mathcal{A}_{i>1} \in \mathcal{A}$ there is at most one $e_i \in \Omega$ such that $t(e_i) \in V_i$ and $\mathcal{A}_i$ is a rooted tree in $t(e_i)$ (or in any other vertex if such edge e_i does not exist); and (iii) if $t(e_i) \in V_i$, then $e_i \setminus \{t(e_i)\} \subseteq \bigcup_{j=1}^{i-1} V_j$.

Figure 1 shows an Epiphytic decomposition $(\mathcal{A}, \Omega, t)$ according to v_1 of the Rosenbrock function $f(x, y) = 100(x - y^2)^2 + (1 - y)^2$ encoded as a ternary network, where $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$, $\mathcal{A}_1 = (\{v_1, \ldots v_{13}\}, \{e_1, \ldots, e_6\})$, $\mathcal{A}_2 = (\{v_{14}\}, \emptyset)$, $\Omega = \{e_7\}$ and $t(e_7) = v_{14}$.

It was showed that an Epiphytic decomposed NCOP instance can be solved in a backtrack-free manner if the network satisfies a combination of GAC and Relational Arc Consistent (RAC) [3]. Such consistencies enable a complete instantiation of the network from the initial assignmet $\langle z = \min Z \rangle$ that minimizes the objective function. RAC is a local consistency proposed in [2] that guarantees any consistent assignment to all variables except one x_j, in the scope of a constraint, to be extended to x_j whilst satisfying the constraint. A constraint of the form $x_1 = x_2 \circ_1 x_3$ is RAC w.r.t. x_1 if $X_1 \supseteq X_2 \circ_1 X_3$.

More specifically, [3] showed it suffices that only the constraints represented by edges $e_i \in \Omega$ are RAC w.r.t. the variable represented by $t(e_i)$, while the other constraints are GAC. An Epiphytic decomposition can be found in linear time w.r.t. the number of instance's operators, however, not all instances

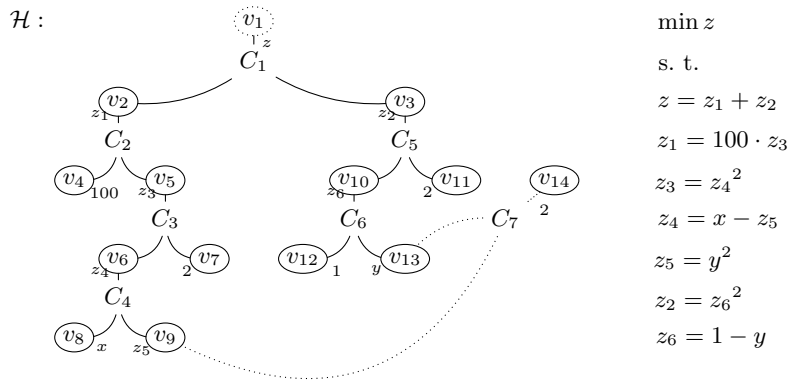

Fig. 1. An Epiphytic decomposition according to v_1 of the ternary encoded Rosenbrock function. Labels below vertices indicate the variables they represent.

have such decomposition. Besides that, it was showed that all instances from an experimental set of COCONUT Benchmark [11] have Epiphytic decompositions.

While the GAC contractor shrinks the domain of variables, an RAC contractor must tighten the constraint on $e_i \setminus \{x_j\}$. If such constraint does not exist it must be added to the network, changing the structure of the hypergraph and its Epiphytic decomposition. To avoid that, [3] proposed a domain filtering algorithm that achieves a relaxed form of RAC. The proposed algorithm forms the basis of a global optimization solver called OGRe (Relaxed Global Optimization)[1] that is a variation of the usual interval B&B for global optimization.

OGRe considers a tolerance ε on constraints represented by the Ω set; therefore, it considers a relaxed version of the original instance. On the pruning phase, the GAC contractor (2) is applied to shrink variables' domain. As a local search strategy, OGRe attempt the entire instantiation of the network in a backtrack-free manner, starting by the initial valuation $\langle z = \min Z \rangle$. If some constraint in Ω is not satisfied with tolerance ε its variables must be narrowed to achieve a better approximation of RAC. Thus, on the branching phase, a variable from the non ε-feasible constraint in Ω is chosen and its domain is bisected to continue the search in a recursive fashion.

3 Differential Evolution

Differential Evolution (DE) was introduced by Storn and Price [12] using a floating-point encoding evolutionary algorithm for global optimization over continuous spaces. The classical DE consists of an evolutionary loop over generations of a population `p` of individuals. An individual is an assignment of values to all variables of the instance, represented by a vector $\mathtt{x} = (a_1, a_2, \ldots, a_D)$ where $a_i \in X_i$ is a value from the domain of the variable x_i, $1 \leq i \leq D$. A popula-

[1] https://gitlab.c3sl.ufpr.br/gaderenievicz/OGRe

tion $\mathtt{p}$ is a set of NP individuals. We denote by $\mathtt{p}_g$ the population on the g-th generation, $0 \leq g \leq G$.

DE follows the general procedure of an evolutionary algorithm. During each generation, the operators of mutation, crossover and selection are performed on the population until a termination condition is satisfied, like a fixed maximum number of fitness evaluations ($MaxFEs$). The initial population ($\mathtt{p}_0$) is randomly generated according to a uniform distribution over $X_1 \times \cdots \times X_D$. In the mutation phase of each generation, a mutation operator is applied to generate a *mutant vector* $\mathtt{v}_i$ for each individual $\mathtt{x}_i$ (called *target vector*) of the population. In the classical DE, several mutation operators have been proposed; for example, DE/rand/1 is described as follows:

$$\mathtt{v}_i = \mathtt{r}_1 + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{3}$$

where $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$ are three mutually distinct individuals randomly selected from the population $\mathtt{p}_g$, and F is the scaling factor. Other popular mutation operators are the DE/current-to-$p$best/1 (4) and the DE/current-to-rand/1 (5):

$$\mathtt{v}_i = \mathtt{x}_i + F \cdot (\mathtt{r}_{\text{pbest}} - \mathtt{x}_i) + F \cdot (\mathtt{r}_1 - \mathtt{r}_2), \tag{4}$$

$$\mathtt{v}_i = \mathtt{x}_i + s \cdot (\mathtt{r}_1 - \mathtt{x}_i) + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{5}$$

where s is a uniformly distributed random number between 0 and 1, p is a value in $[1, NP]$, and $\mathtt{r}_{\text{pbest}}$ is an individual randomly chosen from the p best individuals of the current population.

In the classical DE, after the mutation step a crossover operation is applied to each pair of the target vector $\mathtt{x}_i$ and its corresponding mutant vector $\mathtt{v}_i$ to generate a new individual $\mathtt{u}_i$, called *trial vector*. The basic version of binomial crossover is defined as follows:

$$\mathtt{u}_{ij} = \begin{cases} \mathtt{v}_{ij}, & \text{if } rand_{ij} \leq CR \text{ or } j = j_{rand} \\ \mathtt{x}_{ij}, & \text{otherwise,} \end{cases}$$

where $\mathtt{u}_{ij}$, $\mathtt{v}_{ij}$ and $\mathtt{x}_{ij}$ are the j-th element of the vectors $\mathtt{u}_i$, $\mathtt{v}_i$ and $\mathtt{x}_i$, respectively. The value $rand_{ij}$ is a uniformly distributed random number between 0 and 1, CR is the crossover rate and $j_{rand} \in \{1, \ldots, D\}$ is a randomly chosen index which ensures that the trial individual gets at least one component from the mutant vector. Finally, a selection operator is performed on $\mathtt{x}_i$ and $\mathtt{u}_i$ and the best one according to a fitness evaluation function is selected as a target vector of the next generation.

4 The Proposed Approach

In this section we describe the proposed InDE algorithm. Our approach uses a new representation for the individuals: instead of each individual be an assignment of real values to all variables of the NCOP instance, only the variables present in constraints of the Ω set of an Epiphytic decomposition are considered. Moreover, the representation is based on intervals, i.e., an individual is an

assignment of intervals to variables (a box). The population is a set of boxes that covers parts of the search space, instead of just points in this space as the classical DE. This allows to use the GAC contractor (2) to prune unfeasible solutions. OGRe's local search is used to compute the fitness of each individual. The proposed approach uses interval adaptations of the main features of DE algorithms presented in the last CEC competitions on constrained real-parameter optimization, especially IUDE [15] and LSHADE44 [10], first places on CEC 2018 and CEC 2017, respectively. The components of InDE are discussed below.

4.1 Interval Population

An interval individual is a vector of intervals $\mathtt{X} = (A_1, A_2, \ldots, A_D)$, where $A_i \subseteq X_i$ is an interval. A population $\mathtt{p}$ is a set of NP interval individuals and $\mathtt{p}_g$ denotes the interval population on the g-th generation.

OGRe uses a multi-interval representation of variables' domain for a better use of the GAC contractor. A multi-interval can be represented by an union of intervals and a multi-interval box is a short way of describing a combinatorial of intervals. For example, the multi-interval box $([-2,-1] \cup [1,2], [0,2])$ represents both the boxes $([-2,-1],[0,2])$ and $([1,2],[0,2])$. In our approach we split multi-intervals in a combinatorial representation.

The initial population is generated by the top level branching tree of OGRe's B&B. Each node of the search tree contains a consistent multi-interval box. Each multi-interval box is splitted into a set of interval boxes (individuals) that are added to the initial population. The nodes are iteratively processed until the number of individuals is NP. This strategy guarantees that the initial population covers all search space. Then, the individuals' fitness are evaluated using OGRe's backtrack-free local search (section 4.3). As this procedure is subject to numerical rounding errors, it is possible that some individuals are considered inconsistent. In this case, substitute individuals are randomly generated from the initial consistent box by varying the endpoints of an interval.

Inspired by IUDE, the current population is divided into two sub-populations A and B of size $NP/2$. The current population of NP size is sorted at the beginning of each generation w.r.t. their individuals' fitness. This ensures that the sub-population A will be composed by the best $NP/2$ individuals according to the ε constrained method described in section 4.3. We use these sub-populations to control the application of the mutation operations. In sub-population A we apply a pool of three mutation strategies, promoting the exploitation of the fitter solutions that can speed up the convergence. In order to better allocate the computational resources, in sub-population B an adaptive selection scheme is used to apply only one mutation operator.

DE traditionally employs the one-to-one comparison between target vectors and trial vectors to select the individuals for the next generation. In order to explore promising directions of the search, we maintain an additional memory with the trial vectors that were not selected for the next generation. When a multi-interval box is splitted into interval individuals, only the one with best fitness

value is compared to the target vector, others are added to the additional memory. Periodically, the current population (comprising the two sub-populations) and the additional memory are merged and sorted. Then, the new population is composed by the NP best individuals, and the additional memory is emptied.

We use the Linear Population Size Reduction scheme proposed in LSHADE [14] to linearly reduce the population. In this scheme, NP^{max} is the population size at the beginning of the search and NP^{min} is the population size at the end. At each generation the new population size is calculated and if it is different from the current size, the worst individuals are deleted from the population until it has the appropriate size.

4.2 Interval Operations

Like in IUDE, a pool of three mutation operators consisting of DE/rand/1, DE/current-to-$p$best/1 and DE/current-to-rand/1 is used. In the top sub-population A, each one of the three strategies is used to generate new trial vectors, employing three fitness evaluations for each target vector. Since the sub-population A contains the $NP/2$ best members of the current population, we adopted the same modified ranking-based parent selection used in IUDE. Thus, in the mutation operation the base vector and the terminal vector are selected from the sub-population A.

Since an interval is defined by its width (ω) and midpoint (μ), mutant operators can be applied over midpoints as on real values representation. However, they must be extended to deal with interval widths. Equations 6, 7 and 8 are the result of applying interval arithmetic to the classical mutation operators.

The interval version of DE/rand/1 operator (3) combines $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$ to generate a mutant vector $\mathtt{v}_i$. The j-th element of $\mathtt{v}_i$ is defined by:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{r}_{1j}) + F \cdot (\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j}))\,, \tag{6}$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{r}_{1j}) + F \cdot (\omega(\mathtt{r}_{1j}) \cdot (\omega(\mathtt{r}_{2j})/\omega(\mathtt{r}_{3j})) - \omega(\mathtt{r}_{1j}))\,.$$

The other two mutation operators are defined in the same manner. The interval version of DE/current-to-$p$best/1 (4) combines the target vector $\mathtt{x}_i$ with two randomly selected individuals $\mathtt{r}_1$ and $\mathtt{r}_2$, and with $\mathtt{r}_{\text{pbest}}$, one of the p best individuals of the population. The resulting mutant vector is defined by:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + F \cdot (\mu(\mathtt{r}_{\text{pbest}j}) - \mu(\mathtt{x}_{ij})) + F \cdot (\mu(\mathtt{r}_{1j}) - \mu(\mathtt{r}_{2j}))\,, \tag{7}$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{x}_{ij}) + F \cdot (\omega(\mathtt{r}_{\text{pbest}j}) - \omega(\mathtt{x}_{ij})) + F \cdot (\omega(\mathtt{x}_{ij}) \cdot (\omega(\mathtt{r}_{1j})/\omega(\mathtt{r}_{2j})) - \omega(\mathtt{x}_{ij}))\,.$$

The interval version of DE/current-to-rand/1 (5) combines the target vector $\mathtt{x}_i$ with three randomly selected individuals $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$. The resulting mutant vector is defined by:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + s \cdot (\mu(\mathtt{r}_{1j}) - \mu(\mathtt{x}_{ij})) + F \cdot (\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j}))\,, \tag{8}$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{x}_{ij}) + s \cdot (\omega(\mathtt{r}_{1j}) - \omega(\mathtt{x}_{ij})) + F \cdot (\omega(\mathtt{x}_{ij}) \cdot (\omega(\mathtt{r}_{2j})/\omega(\mathtt{r}_{3j})) - \omega(\mathtt{x}_{ij}))\,,$$

where s is a random number between 0 and 1. This strategy does not use crossover operator, i.e., trial vector $\mathtt{u}_i$ is a copy of mutant vector $\mathtt{v}_i$.

As in IUDE, in the top sub-population A, the three trial vectors generated corresponding to each target vector are compared among themselves, and the trial vector generation strategy with the best trial vector scores a win. At every generation, the success rate of each strategy is evaluated over the period of previous L generations, where L is the learning period. For example, the success rate (SR) of strategy 1 on the learning period $L = 25$ is given by $SR_1 = NW_1/(NW_1 + NW_2 + NW_3)$, where NW_i is the number of wins of the strategy i in the previous 25 generations. In the bottom sub-population B, the probability of employing a trial vector generation strategy is equal to its recent success rate on the top sub-population A over the previous L generations.

The setting of values for the F and CR parameters is problem dependent and may change according to the region of the search space being visited. For this reason, we use the parameter adaptation scheme applied in LSHADE44. Thus, it is employed a pair of memories M_F and M_{CR} for the strategies DE/rand/1 and DE/current-to-$p$best/1. As DE/current-to-rand/1 does not use crossover, we only use one memory M_F. It is important to note that only successful trial vectors in the sub-population A are used in the adaptation of parameters, since only in this sub-population the three strategies are used for each individual.

4.3 Fitness Evaluations

To compute the fitness value of an individual we use OGRe's pruning and local search strategies. First, the GAC contractor (2) shrinks the intervals by removing inconsistent values. Then, we try the instantiation of the entire network in a backtrack-free fashion starting by the initial valuation $\langle z = \min Z \rangle$. Unlike OGRe, we allow the constraints in the Ω set of the Epiphytic decomposition to be instantiated regardless of the tolerance required to satisfy them. The sum of all these tolerances makes the constraint violation value $\phi(\mathtt{x})$ of the individual (if the contractor entails an empty interval we define $\phi(\mathtt{x}) = \infty$), while the $f(\mathtt{x})$ value is simply defined by $\min Z$. The consistent individual obtained from the GAC contractor may contain multi-intervals. Thus, we split this multi-interval box into a set of interval individuals, adding to the population the one with the best fitness value (if it is better than the target vector) and putting in the additional memory all the other generated interval individuals. It is worth noting that unlike usual DE methods, a new individual is not defined only by the mutation and crossover operations, but also by the GAC contractor that shrinks its interval values.

We use the ε constrained method [13] to compare two individuals, similarly to IUDE and LSHADE44. In this method, the fitness value of an individual $\mathtt{x}$ is composed by two values: the objective function value $f(\mathtt{x})$ and the constraint violation value $\phi(\mathtt{x})$. The ε comparisons are defined basically as a lexicographic order in which $\phi(\mathtt{x})$ precedes $f(\mathtt{x})$, i.e., the feasibility is more important than the minimization of the objective function. This precedence is adjusted by the parameter ε. Given two individuals $\mathtt{x}_1$ and $\mathtt{x}_2$ with fitness $(f(\mathtt{x}_1), \phi(\mathtt{x}_1))$ and

$(f(\mathtt{x}_2), \phi(\mathtt{x}_2))$, respectively, $\mathtt{x}_1$ is better than $\mathtt{x}_2$ iff the following conditions are satisfied:

$$(f(\mathtt{x}_1), \phi(\mathtt{x}_1)) <_\varepsilon (f(\mathtt{x}_2), \phi(\mathtt{x}_2)) \Leftrightarrow \begin{cases} f(\mathtt{x}_1) < f(\mathtt{x}_2), & \text{if } \phi(\mathtt{x}_1) \leq \varepsilon, \phi(\mathtt{x}_2) \leq \varepsilon \\ f(\mathtt{x}_1) < f(\mathtt{x}_2), & \text{if } \phi(\mathtt{x}_1) = \phi(\mathtt{x}_2) \\ \phi(\mathtt{x}_1) < \phi(\mathtt{x}_2), & \text{otherwise.} \end{cases}$$

The ε level declines as the generation increases:

$$\varepsilon = \begin{cases} \varepsilon_0 \cdot (1 - \frac{g}{T})^{cp}, & \text{if } 0 < g < T \\ 0, & \text{if } g \geq T \end{cases}, \text{with } cp = \frac{-\log\varepsilon_0 + \lambda}{\log(1 - T)},$$

where ε_0 is the constraint violation of the top θ-th individual in the initial population. The ε level is updated at each generation g until the number of generations exceeds T, from this point the ε level is set to 0 to prefer solutions with minimum constraint violation.

5 Experimental Results

In order to verify the performance of our approach, 80 functions were chosen from the COCONUT Benchmark [11] and tested in 25 independent runs. These functions were selected according to an increasing number of variables and constraints. The maximal number of fitness evaluations was set to $MaxFES = 20000 \times |V_\Omega|$, where $|V_\Omega|$ is the number of variables on the Ω set. Initial population size was defined as $NP^{max} = 12 \times |V_\Omega|$ and the minimal size was $NP^{min} = 6$. The DE/current-to-$p$best/1 operator used $p = 20\%$ of the population. The length of historic memory was $H = 5$. The additional memory and the current population $\mathtt{p}_g$ were merged and sorted every 50 generations and the learning period L was set to 25 generations. Parameters of ε level were $\theta = 0.5$ and $T = 0.75 \times G$, where G is the maximum number of generations. GAC contractor was applied a maximum of 1000 iterations. We chose the exponential version of the classical crossover operator because it outperformed the binomial one in most cases. The timeout was set to $7200s$.

Our experiments considered the OGRe solver, the proposed InDE and a DE variant that uses the same strategies, but does not use interval representation neither structural decomposition. In this case, the variables are represented as real values and the NCOP instance is handled as black-box optimization. As the optimal solution of all instances are known, we used the absolute distance from these values as a performance measure (absolute error). Only solutions that respect all constraints of the instance (feasible) were considered. The maximum tolerance used in OGRe was $\varepsilon = 10^{-1}$.

Figure 2 shows the number of instances solved with a given maximum absolute error. The lines show the best, the median and the worst solutions of 25 runs of InDE and the real parameter DE. For OGRe we only show the best solution of each instance among several parameter settings [3]. Instances with absolute error greater than 10^3 were disregarded.

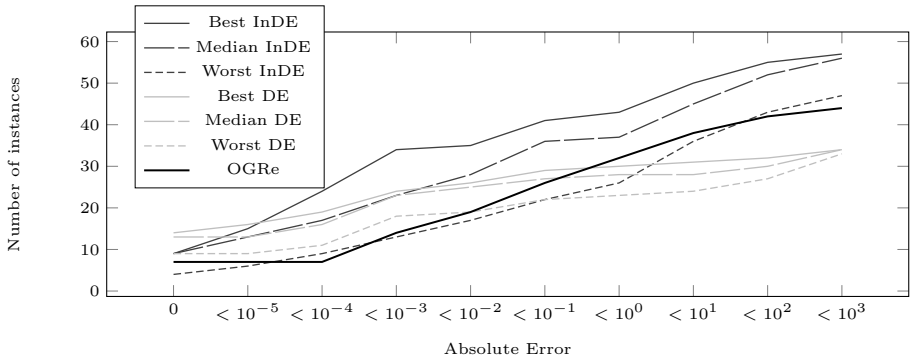

Fig. 2. Accumulated absolute error of the solutions found.

Our InDE approach found feasible solutions for 71.25% of the 80 instances, with 11.25% of optimal solutions and 42.5% with an better approximation than 10^{-3}. Although the real-parameter DE found more optimal solutions than the other approaches, it found feasible solutions for only 42.5% of instances. In general, given a maximum absolute error (*maxAE*), in our experiments the InDE found a solution far from the optimal by at most *maxAE* for a greater number of instances than OGRe and real-parameter DE. It is worth noting that the InDE's search outperformed OGRe's B&B using the same local consistence procedure. The average time of the 25 executions of the 80 instances, including the time-outs, was 1891.23s for the InDE, 0.19s for the DE and 2258.31s for the OGRe. The processing times of InDE and OGRe are significantly higher due to the local consistency process.

6 Conclusion

In this work, an Interval Differential Evolution (InDE) algorithm was proposed to solve constrained optimization problems. InDE combines the main strategies of state-of-the-art algorithms of the CEC competitions on real-parameter optimization [15, 10] with the Epiphytic decomposition approach [3], that allows the structural exploration of the NCOP instances. With this decomposition InDE executes its operators only on a subset of variables, while the others are valuated by a linear propagation through the network of constraints.

The performance of the proposed InDE on the functions of COCONUT Benchmark [11] reveals that the algorithm is capable of finding feasible solutions to more problems. In addition, the comparative analysis with real-parameter black-box DE showed that the exploration of structural information of the instances considerably improves the performance of the search method. Thus, this paper shows that the use of structural decomposition in optimization metaheuristics is a promising investigation direction.

Some future works include extending the InDE to optimize unconstrained problems by removing the ε constrained component. Another aspect to be analyzed is the possibility of reducing the amount of InDE parameters, making them self-adaptive. In addition, our experimental evaluation can be extended by the use of other contractors and different structural decompositions of instances.

References

1. Araya, I., Reyes, V.: Interval branch-and-bound algorithms for optimization and constraint satisfaction: a survey and prospects. J. Glob. Optim. **65**(4), 837–866 (2016)
2. Dechter, R., van Beek, P.: Local and global relational consistency. Theoretical Computer Science **173**(1), 283 – 308 (1997)
3. Derenievicz, G.A., Silva, F.: Epiphytic trees: Relational consistency applied to global optimization problems. In: van Hoeve, W.J. (ed.) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. pp. 153–169. Springer International Publishing, Cham (2018)
4. Faltings, B., Gelle, E.M.: Local consistency for ternary numeric constraints. In: 15th International Joint Conference on Artificial Intelligence. pp. 392–397 (1997)
5. Freuder, E.C.: A sufficient condition for backtrack-free search. Journal of the ACM **29**(1), 24–32 (Jan 1982)
6. Hansen, E., Walster, G.W.: Global optimization using interval analysis. Monographs and textbooks in pure and applied mathematics, Marcel Dekker, New York (2004)
7. Kearfott, R.B.: An interval branch and bound algorithm for bound constrained optimization problems. J. Glob. Optim. **2**(3), 259–280 (1992)
8. Mackworth, A.K.: On reading sketch maps. In: 5th International Joint Conference on Artificial Intelligence. pp. 598–606. Morgan Kaufmann, San Francisco (1977)
9. Moore, R.E.: Interval analysis. Prentice-Hall Englewood Cliffs, N.J (1966)
10. Poláková, R.: L-shade with competing strategies applied to constrained optimization. In: 2017 IEEE congress on evolutionary computation (CEC). pp. 1683–1689. IEEE (2017)
11. Shcherbina, O., Neumaier, A., Sam-Haroud, D., Vu, X.H., Nguyen, T.V.: Benchmarking Global Optimization and Constraint Satisfaction Codes, pp. 211–222. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
12. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
13. Takahama, T., Sakai, S.: Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In: IEEE congress on evolutionary computation. pp. 1–9. IEEE (2010)
14. Tanabe, R., Fukunaga, A.S.: Improving the search performance of shade using linear population size reduction. In: 2014 IEEE congress on evolutionary computation (CEC). pp. 1658–1665. IEEE (2014)
15. Trivedi, A., Sanyal, K., Verma, P., Srinivasan, D.: A unified differential evolution algorithm for constrained optimization problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 1231–1238. IEEE (2017)
16. Van Hentenryck, P.: Numerica: A modeling language for global optimization. In: 15th International Joint Conference on Artificial Intelligence. pp. 1642–1647. Morgan Kaufmann, San Francisco (1997)