I2DE: Improved Interval Differential Evolution for Numerical Constrained Global Optimization

Mariane R. S. 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,guilherme,fabiano`}`@inf.ufpr.br`

Abstract. Several hybrid approaches have been proposed to solve numerical constrained optimization problems. In this paper we present an Improved Interval Differential Evolution (I2DE) that uses structural information of the instance during the optimization process. We extend the math operations supported by a multi-interval core implementation that allows pruning infeasible solutions by using local consistency techniques and a backtrack-free local search. Furthermore, we propose a reformulation of interval evolutionary mutation strategies. A comprehensive experimental analysis is conducted over COCONUT and CEC2018 competition benchmarks and indicates that the hybridization between metaheuristics and constraint programming significantly improves the quality of the solutions. The experimental evaluation shows that our black-box version of I2DE outperformed several state-of-the-art solvers.

Keywords: Global Optimization · Differential Evolution · Interval Methods.

1 Introduction

In the last few decades, interval based solvers have been used to tackle *Numerical Constrained Global Optimization Problems* (NCOP) in a rigorously way [1]. In general, such methods are composed by a complete investigation of the search space, using the structure of the problem and consistency techniques from the constraint programming field to prune infeasible solutions. Despite the great progress of interval techniques, such methods remain impractical in instances with a large number of dimensions.

On the other hand, Differential Evolution (DE) has become one of the most used evolutionary algorithms to deal with large global optimization problems, due to its performance and simplicity [3, 4, 20]. As opposed to interval algorithms, DE based solvers are fast, but do not guarantee the global optimality. Also, DE uses the instance as a black-box model, where its structure is unknown.

A promising strategy to handle with NCOPs is the hybrid approach, where different methods are combined [26]. In this context, the solver *Interval Differential Evolution* (InDE) was proposed [5]. This method integrates the usual DE approach with an interval solver called *Relaxed Global Optimization* (OGRe) [8].

This solver uses a structural decomposition of the NCOP instance to identify the amount of local consistency that guarantees a backtrack-free optimization. Although this is a strong theoretical result, achieving this amount of local consistency can be an intractable problem. Therefore, OGRe uses an *Interval Branch & Bound* (IB&B) method to tackle a relaxed form of the instance. Thus, the solution found by OGRe may be infeasible on the original NCOP instance. Moreover, the computational cost of the IB&B is prohibitive.

The InDE solver extended usual DE operators to the interval context, using OGRe's core to select a subset of variables on which the search process will occur, whilst the others are valuated by constraint propagation. In addition, local consistency techniques are applied to prune infeasible solutions. By combining theses techniques, InDE outperformed OGRe [5].

In this work, we present the *Improved Interval Differential Evolution* (I2DE) solver. We extend InDE and the operators supported by the OGRe's multi-interval core, which allows us to tackle a greater diversity of benchmark functions and real-world problems. Moreover, we present improved versions of three interval evolutionary mutation operations and incorporate several heuristics of state-of-the-art solvers. An extensive experimental analysis performed over the COCONUT [19] and CEC2018 Competition on Constrained Real-Parameter Optimization [27] benchmarks reveals that our I2DE significantly outperforms InDE, OGRe and a black-box version of I2DE, which suggests that hybridization between metaheuristics and structural decomposition is a promising research direction. Furthermore, our black-box solver outperformed several state-of-the-art metaheuristic solvers.

The remaining of this paper is organized as follows: Section 2 contains background definitions of interval methods and metaheuristics. Section 3 provides details of I2DE's features and the main improvements compared to InDE. The experimental analysis is presented in Section 4 and Section 5 concludes this work.

2 Background

A *Numerical Constrained Global Optimization Problem* (NCOP) consists of finding an assignment of values to a set of variables $V = \{x_1, x_2, \ldots, x_D\}$ that minimizes an objective function $f : \mathbb{R}^D \mapsto \mathbb{R}$ subject to a *constraint network* (CN) $\mathcal{N} = (V, \mathcal{D}, \mathcal{C})$, where $\mathcal{D}$ is the domain set of V and $\mathcal{C}$ is a set of constraints of the form $g_i(x_{i_1}, \ldots, x_{i_k}) \leq 0$. Domain sets can be represented by intervals or multi-intervals. Given a closed interval $X = [\underline{x}, \overline{x}]$, we call $\underline{x}, \overline{x} \in \mathbb{R}$ the *endpoints* of X; $\omega(X) = \overline{x} - \underline{x}$ denotes the *width* of X and $\mu(X) = (\underline{x} + \overline{x})/2$ denotes its *midpoint*. A closed interval can be defined by its width and midpoint as follows: $X = [\mu(X) - \omega(X)/2,\ \mu(X) + \omega(X)/2]$. A *multi-interval* $\mathcal{X} = \langle X_1, X_2, \ldots, X_k \rangle$ is an ordered set of disjointed intervals, where $i < j \implies \overline{x}_i < \underline{x}_j$. In this case, the domain set of a CN is a (multi-) interval *box* $(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_D)$.

In this paper, we tackle NCOPs which constraints can be decomposed into a set of ternary constraints[1] $x = y \circ z$ or $x = \diamond y$, where $\circ$ ($\diamond$) is a well-defined binary

[1] A constraint is said to be *ternary* if it involves at most three variables.

(unary) operation. Therefore, auxiliary variables are included in the network. For example, the constraint $x + 2 \cdot \sin(y)^2 \leq 0.5$ is encoded as $\{c_1 = x + a_1,\ \ a_1 = 2 \cdot a_2,\ \ a_2 = {a_3}^2,\ \ a_3 = \sin(y)\}$, where $A_i = (-\infty, +\infty)$ is the domain set of the auxiliary variable a_i and $C_1 = (-\infty, 0.5]$ is the interval constant that represents the original constraint's relation.

2.1 Interval Analysis Applied to Global Optimization

Interval Analysis is a method of numerical analysis introduced by Moore [16]. Given intervals X, Y and Z the *interval extension* of any binary (unary) operation $\circ$ ($\diamond$) well defined in $\mathbb{R}$ is defined by:

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

The sets $X \circ Y$ and $\diamond Z$ can be intervals, multi-intervals or empty sets. It is possible to compute $X \circ Y$ or $\diamond Z$ for all algebraic and the common transcendental functions only by analyzing the endpoints of X and Y or Z [16, 10], e.g., $[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$, $[\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] = [\min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, \max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}]$, $2^{[\underline{z}, \overline{z}]} = [2^{\underline{z}}, 2^{\overline{z}}]$, etc. Given multi-intervals $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$, the operation $\mathcal{X} \circ \mathcal{Y}$ is the tightest multi-interval that contains $\{X \circ Y \mid X \in \mathcal{X}, Y \in \mathcal{Y}\}$, and $\diamond \mathcal{Z}$ is the tightest multi-interval that contains $\{\diamond Z \mid Z \in \mathcal{Z}\}$.

In the constraint programming field, a constraint is locally consistent if it satisfies some specific property within a given box. For instance, a ternary constraint $C : x = y \circ_1 z$ is *Generalized Arc-Consistent* (GAC) w.r.t. a box (X, Y, Z) iff $X \subseteq Y \circ_1 Z$, $Y \subseteq X \circ_2 Z$ and $Z \subseteq X \circ_3 Y$, where $\circ_2$ and $\circ_3$ are the inverse operations of $\circ_1$ that hold the condition $(x = y \circ_1 z) \iff (y = x \circ_2 z) \iff (z = x \circ_1 y)$. In other words, if C is GAC then given any value for one of its variables, one can extend such valuation for its remaining variables whilst satisfying C. This notion of local consistency was proposed by [15].

It is well known that acyclic[2] CN can be solved in a backtrack-free manner if GAC is achieved [6]. Such result can be extended to NCOP by encoding the objective function $f(x)$ as a new constraint $y = f(x)$; after enforcing GAC (by removing inconsistent values from the current box) we instantiate $y = \min Y$ and propagate this valuation over the entire network without encountering any conflicts [8]. However, GAC is not enough when the network is not acyclic.

Another notion of local consistency is *Relational Arc-Consistent* (RAC) [7]. A ternary constraint $C : x = y \circ_1 z$ is RAC w.r.t. a box (X, Y, Z) iff $X \supseteq Y \circ_1 Z$, $Y \supseteq X \circ_2 Z$ and $Z \supseteq X \circ_3 Y$, i.e., given any value for two variables of C, one can extend such valuation for its remaining variable whilst satisfying C.

In [8] it was proposed a decomposition of CNs that relates the amount of consistency necessary to ensure a backtrack-free solution. An *epiphytic decomposition* of a CN is a tuple $(\mathcal{A}, \Omega, t)$, where $\mathcal{A}$ is an ordered set of acyclic networks

[2] The structure of a CN can be represented by a hypergraph which vertices are the variables and for each constraint there is a hyperedge connecting its respective vertices. Therefore, a CN is acyclic if its hypergraph is Berge-acyclic [2].

obtained by removing from the CN a set of constraints Ω and $t : \Omega \mapsto V_\Omega$ is a function that associates each constraint $C \in \Omega$ with one of its variables $t(C)$ satisfying the following: if $t(C)$ belongs to the network N_i then (i) the remaining variables of C belongs to previous networks $N_{j<i}$; and (ii) there is no other constraint $C' \in \Omega$ such that $t(C')$ belongs to N_i. If the CN encodes a NCOP instance, the variable $y = f(c)$ that represents the objective function must be in the first acyclic network of $\mathcal{A}$.

It was shown that if a NCOP $\mathcal{P}$ encoded as a ternary CN is GAC and the constraints in the set Ω of its epiphytic decomposition are RAC, then $\mathcal{P}$ can be solved in a backtrack-free fashion. The proposed OGRe [8] and InDE [5] solvers are based on this relation and attempt to achieve the relational arc-consistency of the CN as a form of optimization. However, enforcing RAC is in general intractable. The approach proposed in OGRe approximates RAC by using an *Interval Branch and Pruning* scheme. If a constraint $(C : x = y \circ z) \in \Omega$ is not RAC under the current box, the domain of some variable in this constraint (excluding $t(C)$) is bisected (*branch*) and GAC is enforced on both sub-problems (*pruning*) before a new verification of the RAC property of Ω constraints. The algorithm continues in a recursive fashion. Therefore, although the ternary decomposition of the original NCOP instance increases the number of variables, not all of them are considered in the *branch* process, but only those of Ω constraints.

In each branch of the search tree, a backtrack-free local search occurs. First, the variable representing the objective function is instantiated with its minimum value in the current box. Next, the valuation is propagated over all the network, following the ordering $\mathcal{A}$ of the epiphytic decomposition. Note that acyclic networks of $\mathcal{A}$ are feasible if GAC was enforced, but the constraints in Ω may be infeasible. There are two main parameters that control OGRe's search procedure: the tolerance ε_Ω allowed in each Ω constraint, and the minimum granularity Δ which intervals can be bisected.

OGRe's approach is a variation of usual *Interval Branch and Bound* (IB&B) methods [1] that have been used in the last few decades to rigorously solve NCOP. Interval methods compute a set of atomic boxes that contains an optimum solution of the NCOP instance. Due to elevated computational cost, these algorithms remain inefficient in instances with many dimensions. On the other hand, a solution found by OGRe may be inexact (w.r.t. objective function cost or constraint violation), because RAC is approximated by a tolerance ε_Ω in Ω constraints. Besides that, find an acceptable value for ε_Ω is not trivial.

The multi-interval core proposed in OGRe is also used in InDE, including the local search procedure, GAC contractor, epiphytic decomposition of the CN, and multi-interval operations $+$, $-$, $*$, $/$, $\wedge$ and $\sqrt{}$.

2.2 Differential Evolution

Storn and Price [20] proposed Differential Evolution (DE) as an Evolutionary Algorithm that combines the coordinates of existing solutions with a particular probability to generate new candidate solutions. The classical DE consists of a loop of G generations over a population of NP 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, $1 \leq i \leq D$. The fitness evaluation of an individual is responsible for determinate its quality throughout the evolutionary process.

The initial population is randomly generated according to a uniform distribution over $X_1 \times \cdots \times X_D$. 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 ($MaxFEs$).

In the mutation phase, an operator is applied to generate a *mutant vector* $\mathtt{v}_i$ for each individual $\mathtt{x}_i$ (called *target vector*) of the population. The most popular mutation operator is DE/rand/1 and is defined by:

$$\mathtt{v}_i = \mathtt{r}_1 + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{1}$$

where F is the scaling factor and $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$ are three mutually distinct individuals randomly selected from the population. Other popular mutation operators are used in this work, such as DE/current-to-rand/1 (Eq.2) [12] and DE/current-to-$p$best/1 (Eq.3) [29]:

$$\mathtt{v}_i = \mathtt{x}_i + s \cdot (\mathtt{r}_1 - \mathtt{x}_i) + F \cdot (\mathtt{r}_2 - \mathtt{r}_3), \tag{2}$$

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

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.

A crossover search operator is applied on the mutant individual $\mathtt{v}_i$ to produce a new solution $\mathtt{u}_i$, called *trial vector*, given a probability defined by the crossover rate CR. In the exponential crossover used in this work, we first choose an integer $d \in [1, D]$ to be the starting point of the target vector for crossover. We also determine another integer value l chosen from $[1, D]$ with probability $\mathsf{P}(l \geq L) = CR^{L-1}$ for any $L > 0$ which denotes how many consecutive decision variables are selected from the mutant vector starting at d position. Then, the offspring is generated as follows:

$$\mathtt{u}_{ij} = \begin{cases} \mathtt{v}_{ij}, & \text{if } j \in \{m_D(d), m_D(d+1), \ldots, m_D(d+l-1)\}, \\ \mathtt{x}_{ij}, & \text{otherwise,} \end{cases}$$

where $m_D(n) = 1 + ((n-1) \mod D)$ allows to iterate cyclically through the 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 chosen for the next generation.

3 Improved Interval Differential Evolution

In this section we describe details of our approach, called I2DE, and emphasize the improvements and differences from the work presented in [5] which introduced the InDE solver, an Interval Differential Evolution approach that uses structural information for global optimization. One of the main contributions of our work is the extension of operations supported by OGRe's multi-interval core. Previously, it was only possible to tackle instances restricted to operations $+$, $-$, $*$, $/$, $\wedge$ and $\sqrt{}$. We extended the multi-interval implementation for handling the following additional operations: *log*, *exp*, *sin*, *cos*, *tan*, *abs*, *sign*, *max* and *min*. Considering the new operations, we implemented a new procedure to identify the tolerance ε_{Ω} needed to approximate RAC. For example, given a constraint $x = y \circ z$, this tolerance is the maximum distance between any value of the multi-interval $\mathcal{Y} \circ \mathcal{Z}$ to its closest value in the multi-interval $\mathcal{X}$. However, the operator $\circ$ may be not well-defined within the box $(\mathcal{Y}, \mathcal{Z})$. In this case, we compute the distance using the inverse operation of $\circ$ that results in the tightest multi-interval. For instance, we can not compute the distance for the constraint $x = y/z$ within the box $Z = [0, 0]$, but we can do it for the inverse $y = x * z$.

In I2DE and InDE, 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 as in classical metaheuristics. In the optimization process, only the variables of constraints in the Ω set of an epiphytic decomposition are considered. This allows to apply local consistency techniques to prune infeasible solutions. OGRe's local search is used to compute the real parameter instantiation of the variables. However, unlike in [5], we use the original instance modeling to evaluate the individual's fitness value, instead of the ternary encoded CN. We propose improved formulations of three DE interval mutation operators. Interval adaptations of the main features of state-of-the-art solvers are implemented. Some details of I2DE's components are discussed below.

3.1 Interval Population

In the same way as in [5], the I2DE's initial population is generated by the top level branching tree of OGRe's IB&B. This strategy guarantees that the initial population covers all the search space. Although each branch of OGRe's IB&B is composed by a GAC multi-interval box, we iteratively split multi-intervals to obtain a set of interval boxes (individuals) that are added to the initial population until the number of individuals is NP. If numerical rounding errors make an individual to be considered inconsistent by OGRe's backtrack-free local search, this individual is replaced by a randomly generated individual from the initial domain of the instance.

The scheme in which the current population is subdivided between the sub-populations A and B of size $NP/2$ was mantained. The entire population is kept sorted, which allows A to contain the best individuals. In order to promote exploitation of the fitter solutions and accelerate the convergence process, we apply a pool of three strategies to each individual of this sub-population. The best one of the three trial individuals is compared to the parent individual in the selection operation. The other individuals are added to an archive, while in [5] they were discarded. In sub-population B, an adaptive scheme is used to choose the mutation strategy to be applied, as proposed in [24]. At the end of each generation, all individuals are sorted and divided between the sub-populations.

In InDE, when a multi-interval box is split into interval individuals, only the best fitting one is compared to the target vector, and the others are added to the archive. Similarly, individuals who are not selected for the next generation are also added to this archive with a maximum size of 100,000 individuals. When the archive is full, or every 50 generations, the current population and the archive are merged, the *NP* best individuals go to the new population, and the archive is emptied. Our tests revealed that this process is computationally expensive and contributes very little to the maintenance of diversity in the population. So I2DE only maintain individuals who are not selected for the next generation in a $NP \times 2.6$ archive, according to the strategy proposed in [29] and used in many state-of-art DE solvers [3, 13, 18].

Finally, we kept the linear population size reduction scheme proposed in [22] and applied in [5]. 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 new size. When the population size reduces, the archive size is proportionally reduced.

3.2 Interval Operations

In IUDE [24] it was proposed the use of three mutation operators: DE/rand/1, DE/current-to-*p*best/1 and DE/current-to-rand/1. Just like in [5], we use this operators pool in the sub-population A in order to intensify the local search around the best individuals in the current population. The three generated trial vectors are compared among themselves and the mutation 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 25 generations. In the bottom sub-population B, the probability of employing a mutation strategy is equal to its recent success rate on the top sub-population A. Additionally, base and terminal vectors are selected from the top sub-population.

Since an interval is defined by its width (ω) and midpoint (μ), mutation operators can be applied over midpoints and extended to deal with interval widths. In [5], it was introduced interval versions of the three mutation operators mentioned above. However, our tests revealed that in practice numerical rounding errors may result in intervals with negative widths. Consequently, with the application of local consistency, the box is considered inconsistent and discarded.

In order to optimize the convergence process, we reformulated the mutation strategies proposed in InDE [5]. The interval version of DE/rand/1 combines $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$ to generate the mutant vector $\mathtt{v}_i$. The j-th element of $\mathtt{v}_i$ is defined by:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{r}_{1j}) + F_i \cdot (\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j}))\,, \tag{4}$$
$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{r}_{1j}) \cdot \left(1 + F_i \cdot \left(\frac{\omega(\mathtt{r}_{2j})}{\omega(\mathtt{r}_{3j})} - 1\right)\right).$$

This formulation reduces numerical errors and avoid inconsistent intervals.

The other two mutation operators are defined in a similar way. The interval version of DE/current-to-rand/1 combines the target vector $\mathtt{x}_i$ with three randomly selected individuals $\mathtt{r}_1$, $\mathtt{r}_2$ and $\mathtt{r}_3$. The mutant vector is defined by:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + s_i \cdot (\mu(\mathtt{r}_{1j}) - \mu(\mathtt{x}_{ij})) + F_i \cdot (\mu(\mathtt{r}_{2j}) - \mu(\mathtt{r}_{3j}))\,, \qquad (5)$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{x}_{ij}) \cdot \left(1 + s_i \cdot \left(\frac{\omega(\mathtt{r}_{1j})}{\omega(\mathtt{r}_{ij})} - 1\right)\right) \cdot \left(1 + F_i \cdot \left(\frac{\omega(\mathtt{r}_{2j})}{\omega(\mathtt{r}_{3j})} - 1\right)\right),$$

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

Finally, the interval version of DE/current-to-$p$best/1 with archive uses the coordinates $\mathtt{x}_i$ and $\mathtt{r}_1$ in the same way as in Eq. 5, while $\mathtt{r}_2$ is randomly chosen from the union of the current population and the archive, and $\mathtt{r}_{pbest}$ is selected among the p best individuals in the population:

$$\mu(\mathtt{v}_{ij}) = \mu(\mathtt{x}_{ij}) + F_i \cdot (\mu(\mathtt{r}_{\text{pbest}j}) - \mu(\mathtt{x}_{ij})) + F_i \cdot (\mu(\mathtt{r}_{1j}) - \mu(\mathtt{r}_{2j}))\,, \qquad (6)$$

$$\omega(\mathtt{v}_{ij}) = \omega(\mathtt{x}_{ij}) \cdot \left(1 + F_i \cdot \left(\frac{\omega(\mathtt{r}_{\text{pbest}j})}{\omega(\mathtt{r}_{ij})} - 1\right)\right) \cdot \left(1 + F_i \cdot \left(\frac{\omega(\mathtt{r}_{1j})}{\omega(\mathtt{r}_{2j})} - 1\right)\right).$$

Whereas in [5] the value of p remained fixed throughout the evolutionary process, we adopted the strategy proposed in [4]. After each generation g, the p value in the next generation $g + 1$ is computed as follows:

$$p = \left(\frac{p_{max} - p_{min}}{MaxFEs}\right) \cdot nfes + p_{min}, \qquad (7)$$

where p_{min} and p_{max} are, respectively, the minimum and maximum values of p, $nfes$ is the current number of fitness evaluations and $MaxFEs$ is the maximum number of fitness evaluations.

Additionally, in [5] if the width $\omega(\mathtt{u}_{ij})$ is greater than the width of the base individual $\omega(\mathtt{r}_{1j})$, it is updated by $\omega(\mathtt{u}_{ij}) := \omega(\mathtt{r}_{1j})$. Considering that this mechanism can decrease the population diversity by forcing the reduction of intervals, it is not used in I2DE.

It is known that the settings of values for the F and CR parameters is instance dependent and may change according to the region of the search space being visited. To make these parameters self-adaptive, InDE employed a scheme introduced in [22] that uses a pair of memory values $\langle M_{CR}, M_F\rangle$ for each mutation operator in order to store the settings that were successful in the last generations. All the memory pairs are stored in a vector of H positions. At each generation one of the positions is circularly updated based on the weighted Lehmer mean of the fitness differences between offspring and their parents.

In I2DE we incorporated some features proposed in [3, 4]. The first $H - 1$ positions of the vector are initialized with $\langle 0.8, 0.3\rangle$. The last position is always set to $\langle 0.9, 0.9\rangle$ and remains unchanged during the evolutionary process. At each generation one of the first $H - 1$ positions are circularly updated based on the Euclidean distance between the coordinates of offspring and their parents. Also, very low values of CR and very high values of F are not allowed in early stages of the search. It is important to note that only successful trial vectors in A are used in the adaptation of parameters, since only in this sub-population the three strategies are used for each individual.

3.3 Fitness Evaluations

To estimate the quality of the boxes that represent individuals in the population, we use OGRe's pruning and local search strategy. First, we enforce GAC on individuals from the initial population or resulting from the operators mentioned in Section 3.2. Then, we try the instantiation of the entire CN in a backtrack-free fashion starting by the initial valuation $y = \min Y$, where Y is the interval domain of the objective function $f(\mathtt{x})$. We allow the constraints in the Ω set of the epiphytic decomposition to be instantiated regardless of the tolerance $_{\Omega}$ required to satisfy them. In [5], the sum of all these tolerances is considered the *constraint violation* value $\phi(\mathtt{x})$ of the individual, while its *cost* is $f(\mathtt{x}) = \min Y$.

In this paper we assume that the instantiation of real parameters described above provides suitable reference points for evaluating interval individuals. However, to obtain the real values of $f(\mathtt{x})$ and $\phi(\mathtt{x})$, we only consider the instantiation of variables belonging to the original modeling of the instance, not the approximation given by the ternary CN employed in [5]. This allows us to compare results with recent black-box solvers that use the original instance modeling.

The consistent individuals obtained by the GAC contractor may contain multi-intervals. In [5], such multi-interval box was split 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 saving all other generated interval individuals in the additional archive. As commented in the Section 3.1, our tests revealed that this mechanism is computationally expensive and does not contribute to maintaining the population diversity. So, instead of splitting the multi-intervals and adding them to the archive, we use the *interval hull* of the box (the smallest interval box that contains all the multi-intervals) and consider it as only one individual which is compared with its parent in the selection operation.

In order to compare two individuals, we apply the widely used ε constrained method [21]. The ε comparisons are defined as a lexicographic order in which $\phi(\mathtt{x})$ precedes $f(\mathtt{x})$. This precedence is adjusted by the parameter ε that is updated at each generation until the number of generations exceeds a predefined threshold. From this point the ε level is set to 0 to prefer solutions with minimum constraint violation.

4 Experimental Results

Our experimental evaluation considers four solvers: the improved approach proposed in this paper (I2DE); an implementation that uses the same DE interval operations and additional archive of the original InDE [5]; a black-box version of the I2DE (BBDE); and OGRe [8]. The aim of use the features of original InDE is to measure the impact of the modifications incorporated in our approach. The black-box version does not use interval representation neither structural decomposition as the other three white-box solvers. This comparison aims to evaluate the impact of using the instance structure in optimization process. In turn, the comparative analysis with OGRe is intended to provide a baseline with a search method that employs IB&B, local consistency and constraint propagation over the constraint network. It is important to note that local consistency and backtrack-free local search techniques are the same in I2DE, InDE and OGRe.

To ensure a fair comparison, the three DE solvers employ the same heuristics coming from state-of-the-art solvers for adapting and adjusting parameters, as well as the same number of fitness evaluations. Furthermore, the input parameters of the solvers were empirically defined after a series of preliminary experiments. It was set to $MaxFEs = 20000 \times D$. The maximal and minimal population size was defined as $NP_{max} = 15 \times D$ and $NP_{min} = 6$, respectively, where D is the number of variables on the original instance. Note that in [5], $MaxFEs$ and NP were defined in relation to $|V_{\Omega}|$, which resulted in a much larger budget and population. The DE/current-to-$p$best/1 operator used $p_{max} = 0.25$ and $p_{min} = p_{max}/2$. The length of historical memory was $H = 5$. Parameters of ε level were $\theta = 0.7$, $cp = 4$ and $T = 0.85 \times G$, where G is the maximum number of generations. We used the exponential version of the classical crossover operator. GAC contractor was applied to a maximum of 1000 iterations. The timeout for OGRe was $10000s$. Since OGRe parameters ε_{Ω} and Δ are not trivially configured, and to have a similar methodology of execution to DEs, we ran 25 different configurations for each instance using $\varepsilon_{\Omega} \in \{10^e \mid e = -5, \ldots, 1\}$ and $\Delta \in \{10^e \mid e = -6, \ldots, 0\}$, $\Delta \leq \varepsilon_{\Omega}$.

Although most experimental evaluations of metaheuristic-based solvers use CEC competition benchmarks, their instances are only available in source code to be used as black-box evaluation functions. Therefore, they do not provide the necessary formalization for use in solvers that exploit structural information of the NCOP instance, such as I2DE. For the main experimental evaluation we used the COCONUT Benchmark [19] due to its wide use in numerical optimization research and because it contains the AMPL (A Mathematical Programming Language) description necessary to explore the structure of the instances in a white-box approach.

However, to provide a baseline with some state-of-the-art solvers[3], we compared our black-box version of I2DE (BBDE) with solvers from CEC2018 Competition on Constrained Real-Parameter Optimization [27] in the 28 proposed benchmark functions with $D = \{10, 30, 50, 100\}$, totalizing 112 instances. The results of the CEC2018 solvers were obtained from the competition records[4]. BBDE used the same competition protocol, with 25 runs for instance.

The rank of Table 1 considers the CEC2018 [27] methodology based on mean and median score values on each instance, in which the best solver obtains the lowest total score. The results indicate that our black-box version is highly competitive with some of the most popular state-of-the-art solvers.

To investigate the performance of I2DE, experiments were conducted on 155 optimization problems from the COCONUT Benchmark [19] with different num-

[3] LSHADE-IEpsilon [9], εMAgES [11], LSHADE44 [17], UDE [23], IUDE [24], LSHADE+IDE [25] and CAL-SHADE [28].

[4] https://www3.ntu.edu.sg/home/EPNSugan/index_files/CEC2018/CEC2018.htm, last accessed 18 Jun 2021.

Table 1. Comparison of BBDE with solvers from CEC2018.

solver	mean	median	total	rank	solver	mean	median	total	rank
BBDE	**381**	**380**	**761**	**1st**	LSHADE44	462	485	947	5th
IUDE	445	398	843	2nd	UDE	521	481	1002	6th
εMAgES	439	409	848	3rd	LSHADE+IDE	529	548	1077	7th
LSHADE-IEpsilon	445	449	894	4th	CAL-SHADE	647	575	1222	8th

ber of equality and inequality constraints and up to 75 dimensions. Note that it was only possible to tackle a larger number of functions than in [5] because our approach extends the implementation of OGRe's multi-interval core to other math operations, as commented in Section 3.

The first aspect to be analyzed is the impact of using structural decomposition. Transforming the original instance modeling into a ternary CN usually involves adding an auxiliary variable for each occurrence of math operators. From this, the variables of constraints in the Ω set, V_Ω, whose valuation is critical for the search process, are extracted. As noted in Section 3.2, I2DE and InDE only apply its operators on these variables, while the other valuations are assigned through OGRe's backtrack-free local search. Figure 1 illustrates the number of variables (and operators) from the original modeling, the ternary representation and the V_Ω size in the 155 instances. Although V_Ω presents a considerable reduction in number of variables compared to the ternary representation, it is evident that the search space is still larger than in the original instance modeling.

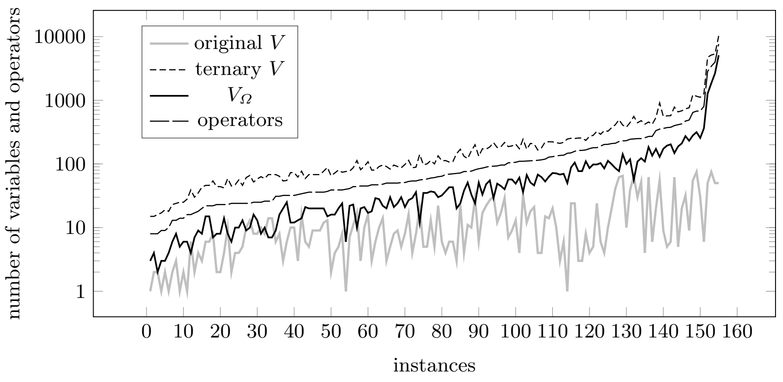

Fig. 1. Number of variables in original V, ternary V, V_Ω and operators per instance.

The solvers ran on a computer with Intel Xeon E5-4627v2 3.30GHz processor, 256GB of RAM and CentOS Linux 7. The average time of the 25 executions of the 155 instances was 7372.04s for I2DE, 5031.19s for InDE, 4.63s for BBDE and 5619.52s for OGRe. The processing times of I2DE, InDE and OGRe are significantly higher due to the local consistency process. Moreover, the execution time of InDE is smaller compared to I2DE because many candidate solutions resulting from InDE interval operations considered inconsistent are immediately discarded, without going through the local consistency process.

I2DE found feasible solutions on 119 instances, against 107 in InDE, 115 in BBDE and 64 in OGRe. Feasible solutions with value at most 10^{-8} greater than the optimal value are considered as optimal solutions. So, I2DE found 40 optimal solutions, while InDE, BBDE and OGRe found 35, 35 and 5, respectively.

In order to perform a comparative analysis among I2DE, InDE, BBDE and OGRe we used two methodologies. The first was the one applied in CEC2018 [27], the same used in Table 1. The second was proposed in CEC2020 Competition on Non-Convex Constrained Optimization Problems from the Real-World [14] and involves the objective function and constraints violation of the best, the mean, and the median solutions. The values are normalized and adjusted to calculate the performance measure. The total score is a weighted composition of best, mean and median with weights 50%, 30% and 20%, respectively.

To estimate the feasibility of the four solvers, we compared the average constraint violation values of the 25 runs for the 155 benchmark instances. As the pre-analysis rejected the null hypothesis that the distribution of violation values of each instance is normal, we chose a non-parametric test. The Friedman test with post-hoc Nemenyi test with significance level of 0.05 was applied to the violation results of each instance, comparing four sets of 25 values each time. The null hypothesis that the four solvers perform equally was rejected in all instances with p-value close to 10^{-6}. Based on the Nemenyi test rank on each instance, we counted the number of instances where each solver was better, equal or worse than the others. If the rank distance of two solvers was less than the critical distance of the test, we considered the solvers perform equally in the instance. Table 2 shows in its violation rows the number of instances where I2DE was better, equal or worse than the other three solvers. In all three cases, I2DE obtained more solutions with violation values smaller than the other solvers.

A second statistical test was conducted over the normalized and adjusted objective function value of best, mean and median solution of the CEC2020 methodology. The value of each solver in an instance is a composition of objective function value and the average constraint violations of the obtained solutions. As the pre-analysis rejected the null hypothesis that the distribution of each set of values is normal, we chose a non-parametric test. We applied the Wilcoxon signed rank test with significance level of 0.05 to compare the solvers on each set of values: best, mean and median. Table 2 shows the test results and the number of instances where I2DE was better, equal or worse than the other solver. Only in two cases the null hypothesis, that the two solvers perform equally, was not rejected: decision $\approx$. In the other cases, I2DE perform better: decision $+$.

According to both CEC2018 and CEC2020 ranking methodologies, the best solver will obtain the lowest total score value. Table 3 shows that the I2DE outperforms the other three approaches in both ranking methodologies. In comparison to InDE, the results point to a significant performance improvement due to the reformulation of interval mutation strategies and the maintenance

Table 2. Statistical tests results and comparison of I2DE with InDE, BBDE and OGRe based on best, mean, median and violation scores over 155 COCONUT instances.

solvers	criteria	better	equal	worse	p-value	decision
I2DE vs. InDE	best	66	60	29	0.0002	+
	mean	89	42	24	10^{-8}	+
	median	85	46	24	10^{-6}	+
	violation	39	110	6	$\sim 10^{-6}$	+
I2DE vs. BBDE	best	60	36	59	0.9204	$\approx$
	mean	64	33	58	0.0002	+
	median	61	39	55	0.9987	$\approx$
	violation	47	83	25	$\sim 10^{-6}$	+
I2DE vs. OGRe	best	124	10	21	10^{-16}	+
	mean	131	9	15	10^{-24}	+
	median	127	7	21	10^{-20}	+
	violation	108	40	7	$\sim 10^{-6}$	+

Table 3. Rank of I2DE, InDE, BBDE and OGRe according to CEC2018 and CEC2020 methodologies on COCONUT Benchmark.

	CEC2018				CEC2020				
solver	mean	median	total	rank	best	mean	median	total	rank
I2DE	**249**	**229**	**478**	**1st**	**0.2153**	**0.0811**	**0.1348**	**0.1589**	**1st**
InDE	330	309	639	3rd	0.3068	0.1361	0.2345	0.2411	2nd
BBDE	317	320	637	2nd	0.2975	0.2340	0.3049	0.2799	3rd
OGRe	575	543	1118	4th	0.7306	0.8123	0.7314	0.7553	4th

of population diversity with the additional archive of solutions. Furthermore, the comparative analysis with BBDE and OGRe suggests that the hybridization between metaheuristics, constraint programming and structural decomposition is a promising research direction.

5 Conclusion

In this work we proposed the I2DE, an improved version of Interval Differential Evolution that uses structural information to solve global optimization problems. From exploration of the epiphytic decomposition, it becomes possible to concentrate the search process only on a subset of variables that have critical valuation, while all the others are instantiated by propagation through the constraint hypergraph. Additionally, our search is enhanced by a local consistency process that prunes values that certainly do not constitute the optimal solution.

One of the main contributions of our approach is the extension of the multi-interval core of OGRe. This allows us to tackle a greater diversity of benchmark functions and real-world problems that otherwise could not be adequately represented. We also proposed a reformulation of three DE interval mutation operations and incorporated heuristics from several state-of-the-art solvers that contributed to improve the performance of our approach.

The experimental analysis of the proposed I2DE on the 155 functions selected from the COCONUT Benchmark [19] showed that the reformulation of the interval mutation strategies and of the additional archive significantly improved the performance of the search method. Furthermore, the results obtained in comparison with OGRe and BBDE reveal that, although the exploration of the instance's structural information increases the size of the search space and the processing time, it considerably improves the quality of the solutions found. Considering that BBDE outperformed several state-of-the-art solvers and was overcome by I2DE, we shown that the use of structural instance information in the context of metaheuristics is a promising research direction.

Some future work includes the implementation of other contractors that help to efficiently prune the intervals without loss of solutions. In addition, we intend to develop a hybrid cooperative approach that joins our interval metaheuristics with exact methods that also use this representation for the solutions.

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. Berge, C.: Graphs and Hypergraphs. Elsevier Science Ltd., Oxford, UK, UK (1985)
3. Brest, J., Maučec, M.S., Bošković, B.: iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 1188–1195. IEEE (2016)
4. Brest, J., Maučec, M.S., Bošković, B.: Single objective real-parameter optimization: Algorithm jSO. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 1311–1318. IEEE (2017)
5. Cassenote, M.R.S., Derenievicz, G.A., Silva, F.: Interval Differential Evolution using structural information of global optimization problems. In: EPIA Portuguese Conference on Artificial Intelligence. pp. 724–736. Springer (2019)
6. Cohen, D., Jeavons, P.: The power of propagation: when gac is enough. Constraints **22**, 3–23 (2016)
7. Dechter, R., van Beek, P.: Local and global relational consistency. In: Montanari, U., Rossi, F. (eds.) Principles and Practice of Constraint Programming — CP '95. pp. 240–257. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
8. 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)
9. Fan, Z., Fang, Y., Li, W., Yuan, Y., Wang, Z., Bian, X.: LSHADE44 with an improved ε constraint-handling method for solving constrained single-objective optimization problems. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2018)
10. Hansen, E., Walster, G.W.: Global optimization using interval analysis. Monographs and textbooks in pure and applied mathematics, New York (2004)
11. Hellwig, M., Beyer, H.G.: A matrix adaptation evolution strategy for constrained real-parameter optimization. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2018)

12. Iorio, A.W., Li, X.: Solving rotated multi-objective optimization problems using Differential Evolution. In: Australasian joint conference on artificial intelligence. pp. 861–872. Springer (2004)
13. Jou, Y.C., Wang, S.Y., Yeh, J.F., Chiang, T.C.: Multi-population modified L-SHADE for single objective bound constrained optimization. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2020)
14. Kumar, A., Wu, G., Ali, M.Z., Mallipeddi, R., Suganthan, P.N., Das, S.: A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm and Evolutionary Computation **56**, 100693 (2020)
15. Mackworth, A.K.: On reading sketch maps. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence, IJCAI 1977. pp. 598–606. MIT, Cambridge, MA (1977)
16. Moore, R.E.: Interval analysis. Prentice-Hall Englewood Cliffs, N.J (1966)
17. Poláková, R.: L-SHADE with competing strategies applied to constrained optimization. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 1683–1689. IEEE (2017)
18. Sallam, K.M., Elsayed, S.M., Chakrabortty, R.K., Ryan, M.J.: Improved multi-operator Differential Evolution algorithm for solving unconstrained problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2020)
19. 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)
20. 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)
21. Takahama, T., Sakai, S.: Constrained optimization by the ε constrained Differential Evolution with an archive and gradient-based mutation. In: 2010 IEEE Congress on Evolutionary computation. pp. 1–9. IEEE (2010)
22. 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)
23. 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)
24. Trivedi, A., Srinivasan, D., Biswas, N.: An improved unified Differential Evolution algorithm for constrained optimization problems. IEEE (2018)
25. Tvrdík, J., Poláková, R.: A simple framework for constrained problems with application of L-SHADE44 and IDE. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 1436–1443. IEEE (2017)
26. Vanaret, C., Gotteland, J.B., Durand, N., Alliot, J.M.: Preventing premature convergence and proving the optimality in evolutionary algorithms. In: International Conference on Artificial Evolution. pp. 29–40. Springer (2013)
27. Wu, G., Mallipeddi, R., Suganthan, P.: Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. Tech. rep. (2017)
28. Zamuda, A.: Adaptive constraint handling and success history Differential Evolution for CEC 2017 constrained real-parameter optimization. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 2443–2450. IEEE (2017)
29. Zhang, J., Sanderson, A.C.: JADE: adaptive Differential Evolution with optional external archive. IEEE Trans. on Evolutionary Computation **13**(5), 945–958 (2009)