Hybridization of Interval CP and Evolutionary Algorithms for Optimizing Difficult Problems (CP 2015)

Charlie Vanaret, Jean-Baptiste Gotteland, Nicolas Durand, Jean-Marc Alliot

Hybridization of Interval CP and Evolutionary Algorithms for Optimizing Difficult Problems

C. Vanaret[1], J-B. Gotteland[2], N. Durand[2], and J-M. Alliot[1]

[1] Institut de Recherche en Informatique de Toulouse
2 rue Charles Camichel, 31000 Toulouse, France
charlie.vanaret@enseeiht.fr, jean-marc.alliot@irit.fr
[2] Ecole Nationale de l'Aviation Civile
7 avenue Edouard Belin, 31055 Toulouse Cedex 04, France
gottelan@recherche.enac.fr, durand@recherche.enac.fr

Abstract. The only rigorous approaches for achieving a numerical proof of optimality in global optimization are interval-based methods that interleave branching of the search-space and pruning of the subdomains that cannot contain an optimal solution. State-of-the-art solvers generally *integrate* local optimization algorithms to compute a good upper bound of the global minimum over each subspace. In this document, we propose a *cooperative* framework in which interval methods cooperate with evolutionary algorithms. The latter are stochastic algorithms in which a population of candidate solutions iteratively evolves in the search-space to reach satisfactory solutions.
Within our cooperative solver Charibde, the evolutionary algorithm and the interval-based algorithm run in parallel and exchange bounds, solutions and search-space in an advanced manner via message passing. A comparison of Charibde with state-of-the-art interval-based solvers (GlobSol, IBBA, Ibex) and NLP solvers (Couenne, BARON) on a benchmark of difficult COCONUT problems shows that Charibde is highly competitive against non-rigorous solvers and converges faster than rigorous solvers by an order of magnitude.

1 Motivation

We consider n-dimensional continuous constrained optimization problems over a hyperrectangular domain $\boldsymbol{D} = D_1 \times \ldots \times D_n$:

$$\begin{aligned} (\mathcal{P}) \quad & \min_{\boldsymbol{x} \in \boldsymbol{D} \subset \mathbb{R}^n} & & f(\boldsymbol{x}) \\ & s.t. & g_i(\boldsymbol{x}) \leq 0, & \quad i \in \{1, \ldots, m\} \\ & & h_j(\boldsymbol{x}) = 0, & \quad j \in \{1, \ldots, p\} \end{aligned} \tag{1}$$

When f, g_i and h_j are non-convex, the problem may have multiple local minima. Such difficult problems are generally solved using generic exhaustive branch and bound (BB) methods. The objective function and the constraints are bounded on disjoint subspaces by enclosure methods. By keeping track of the best known upper bound of the global minimum, subspaces that cannot contain a global minimizer are discarded (pruned).

Several authors proposed hybrid approaches in which a BB algorithm cooperates with another technique to enhance the pruning of the search-space. Hybrid algorithms may be classified into two categories [24]: *integrative* approaches, in which one of the two methods replaces a particular operator of the other method, and *cooperative* methods, in which the methods are independent and are run sequentially or in parallel. Previous works include

- integrative approaches: [34] integrates a stochastic genetic algorithm (GA) within an interval BB. The GA provides the direction along which a box is partitioned, and an individual is generated within each subbox. At each generation, the best evaluation updates the best known upper bound of the global minimum. In [9], the crossover operator is replaced by a BB that determines the best individual among the offspring.
- cooperative approaches: [28] sequentially combines an interval BB and a GA. The interval BB generates a list $\mathcal{L}$ of remaining small boxes. The GA's population is initialized by generating a single individual within each box of $\mathcal{L}$. [12] (BB and memetic algorithm) and [7] (beam search and memetic algorithm) describe similar parallel strategies: the BB identifies promising regions that are then explored by the metaheuristic. [1] hybridizes a GA and an interval BB. The two independent algorithms exchange upper bounds and solutions through shared memory. New optimal results are presented for the rotated Michalewicz ($n = 12$) and Griewank functions ($n = 6$).

In this communication, we build upon the cooperative scheme of [1]. The efficiency and reliability of their solver remain very limited; it is not competitive against state-of-the-art solvers. Their interval techniques are naive and may lose solutions, while the GA may send evaluations subject to roundoff errors. We propose to hybridize a stochastic differential evolution algorithm (close to a GA), described in Section 2, and a deterministic interval branch and contract algorithm, described in Section 3. Our hybrid solver Charibde is presented in Section 4. Experimental results (Section 5) show that Charibde is highly competitive against state-of-the-art solvers.

2 Differential Evolution

Differential evolution (DE) [29] is among the simplest and most efficient metaheuristics for continuous problems. It combines the coordinates of existing individuals (candidate solutions) with a given probability to generate new individuals. Initially devised for continuous unconstrained problems, DE was extended to mixed problems and constrained problems [23].

Let NP denote the size of the population, $W > 0$ the amplitude factor and $CR \in [0, 1]$ the crossover rate. At each generation (iteration), NP new individuals are generated: for each individual $\boldsymbol{x} = (x_1, \dots, x_n)$, three other individuals $\boldsymbol{u} = (u_1, \dots, u_n)$ (called *base individual*), $\boldsymbol{v} = (v_1, \dots, v_n)$ and $\boldsymbol{w} = (w_1, \dots, w_n)$, all different and different from $\boldsymbol{x}$, are randomly picked in the population. The coordinates y_i of the new individual $\boldsymbol{y} = (y_1, \ldots, y_n)$ are computed according to

$$y_i = \begin{cases} u_i + W \times (v_i - w_i) & \text{if } i = R \text{ or } r_i < CR \\ x_i & \text{otherwise} \end{cases} \tag{2}$$

where r_i is picked in $[0, 1]$ with uniform probability. The index R, picked in $\{1, \ldots, n\}$ with uniform probability for each $\boldsymbol{x}$, ensures that at least a coordinate of $\boldsymbol{y}$ differs from that of $\boldsymbol{x}$. $\boldsymbol{y}$ replaces $\boldsymbol{x}$ in the population if it is "better" than $\boldsymbol{x}$ (in unconstrained optimization, $\boldsymbol{y}$ is better than $\boldsymbol{x}$ if it improves the objective function).

Figure 1 depicts a two-dimensional crossover between individuals $\boldsymbol{x}$, $\boldsymbol{u}$ (base individual), $\boldsymbol{v}$ and $\boldsymbol{w}$. The contour lines of the objective function are shown in grey. The difference $\boldsymbol{v} - \boldsymbol{w}$, scaled by W, yields the direction (an approximation of the direction opposite the gradient) along which $\boldsymbol{u}$ is translated to yield $\boldsymbol{y}$.

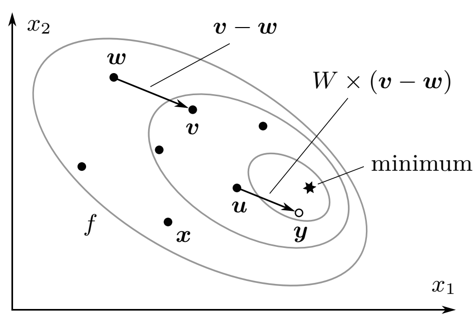


Fig. 1: Crossover of the differential evolution

3 Reliable Computations

The only reliable approaches for achieving a numerical proof of optimality in global optimization are interval-based methods that interleave branching of the search-space and pruning of the subdomains that cannot contain an optimal solution.

Reliable (or rigorous) methods provide bounds on the global minimum, even in the presence of roundoff errors. Section 3.1 introduces interval arithmetic, an extension of real arithmetic. Reliable global optimization is detailed in Section 3.2, and interval contractors are mentioned in Section 3.3.

3.1 Interval Arithmetic

An *interval* X with floating-point bounds defines the set $\{x \in \mathbb{R} \mid \underline{X} \leq x \leq \overline{X}\}$. $\mathbb{IR}$ denotes the set of all intervals. The *width* of X is $w(X) = \overline{X} - \underline{X}$. $m(X) =$

$\frac{\underline{X}+\overline{X}}{2}$ is the *midpoint* of X. A *box* $\boldsymbol{X}$ is a Cartesian product of intervals. The width of a box is the maximum width of its components. The *convex hull* $\square(X, Y)$ of X and Y is the smallest interval enclosing X and Y.

Interval arithmetic [19] extends real arithmetic to intervals. Interval arithmetic implemented on a machine must be *rounded outward* (the left bound is rounded toward $-\infty$, the right bound toward $+\infty$) to guarantee conservative properties. The interval counterparts of binary operations and elementary functions produce the smallest interval containing the image. The conservative properties of interval arithmetic allow to build interval extensions (Definition 1) of functions that may be expressed as a finite composition of elementary functions.

Definition 1 (Interval extension). *Let* $f : \mathbb{R}^n \to \mathbb{R}$. $F : \mathbb{IR}^n \to \mathbb{IR}$ *is an* interval extension *(or inclusion function) of* f *iff*

$$\begin{aligned} \forall \boldsymbol{X} \in \mathbb{IR}^n, \quad & f(\boldsymbol{X}) := \{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}\} \subset F(\boldsymbol{X}) \\ \forall \boldsymbol{X} \in \mathbb{IR}^n, \forall \boldsymbol{Y} \in \mathbb{IR}^n, \quad & \boldsymbol{X} \subset \boldsymbol{Y} \Rightarrow F(\boldsymbol{X}) \subset F(\boldsymbol{Y}) \end{aligned} \tag{3}$$

Interval extensions with various sharpnesses may be defined (Example 1). The *natural interval extension* F_N replaces the variables with their domains and the elementary functions with their interval counterparts. The *Taylor interval extension* F_T is based on the Taylor expansion at point $\boldsymbol{c} \in \boldsymbol{X}$.

Example 1 (Interval extensions). Let $f(x) = x^2 - x$, $X = [-2, 0.5]$ and $c = -1 \in X$. Then

- $F_N(X) = X^2 - X = [-2, 0.5]^2 - [-2, 0.5] = [0, 4] - [-2, 0.5] = [-0.5, 6]$;
- $F_T(X, c) = 2 + (2X - 1)(X + 1) = 2 + [-5, 0][-1, 1.5] = [-5.5, 7]$.

Example 1 shows that interval arithmetic often overestimates the range of a real-valued function. This is due to the *dependency problem*, an inherent behavior of interval arithmetic. Dependency decorrelates multiple occurrences of the same variable in an analytical expression (Example 2).

Example 2 (Dependency). Let $X = [-5, 5]$. Then

$$\begin{aligned} X - X = [-10, 10] &= \{x_1 - x_2 \mid x_1 \in X, x_2 \in X\} \\ &\supset \{x - x \mid x \in X\} = \{0\} \end{aligned} \tag{4}$$

Interval extensions (F_N, F_T) have different convergence orders, that is the overestimation decreases at different speeds with the width of the interval.

3.2 Global Optimization

The conservative properties of interval arithmetic allow to compute a rigorous enclosure of the range of a function over a box. The first branch and bound algorithms for continuous optimization based on interval arithmetic were devised in the 1970s [20][27], then refined during the following years [14]: the search-space is partitioned into subspaces (boxes that may share faces). The objective function and the constraints are evaluated on each subspace. The subspaces that cannot contain a global minimizer are discarded and are not further explored.

To overcome the pessimistic enclosures of interval arithmetic, interval branch and bound algorithms have recently been endowed with filtering algorithms (Section 3.3) that narrow the bounds of the boxes without loss of solutions. Stemming from the Interval Analysis and Interval Constraint Programming communities, filtering (or contraction) algorithms discard values from the domains by enforcing local (each constraint individually) or global (all constraints simultaneously) consistencies. The resulting methods, called interval branch and contract (IBC) algorithms, interleave steps of contraction and steps of bisection.

3.3 Interval Contractors

State-of-the-art contractors (contraction algorithms) include HC4 [6], Box [32], Mohc [2], 3B [17], CID [31] and X-Newton [3]. Only HC4 and X-Newton are used in this communication.

HC4Revise is a two-phase algorithm that exploits the syntax tree of a constraint to contract each occurrence of the variables. The first phase (evaluation) evaluates each node (elementary function) using interval arithmetic. The second phase (propagation) uses projection functions to inverse each elementary function. HC4Revise is generally used as the revise procedure (subcontractor) of HC4, an AC3-like propagation loop.

X-Newton computes an outer linear relaxation of the objective function and the constraints, then computes a lower bound of the initial problem using LP techniques (e.g. the simplex algorithm). $2n$ additional calls may contract the domains of the variables.

4 Charibde: a Cooperative Approach

4.1 Hybridization of Stochastic and Deterministic Techniques

Our hybrid algorithm *Charibde* (Cooperative Hybrid Algorithm using Reliable Interval-Based methods and Differential Evolution), written in OCaml [16], combines a stochastic DE and a deterministic IBC for non-convex constrained optimization. Although it embeds a stochastic component, Charibde is a *fully rigorous* solver.

Previous Work Preliminary results of a basic version of Charibde were published in 2013 [33] on classical multimodal problems (7 bound-constrained and 4 inequality-constrained problems) widely used in the Evolutionary Computation community. We showed that Charibde benefitted from the start of convergence of the DE algorithm, and completed the proof of convergence faster than a standard IBC algorithm. We provided new optimal results for 3 problems (Rana, Eggholder and Michalewicz).

Contributions In this communication, we present *two contributions*:

1. we devised a new cooperative exploration strategy MaxDist that
   – selects boxes to be explored in a novel manner;
   – periodically reduces DE's domain;
   – restarts the population within the new (smaller) domain.
   An example illustrates the evolution of the domain without loss of solutions;
2. we assess the performance of Charibde against state-of-the-art rigorous (GlobSol, IBBA, Ibex) and non-rigorous (Couenne, BARON) solvers on a benchmark of difficult problems.

Cooperative Scheme Two independent parallel processes exchange bounds, solutions and search-space via MPI message passing (Figure 2).

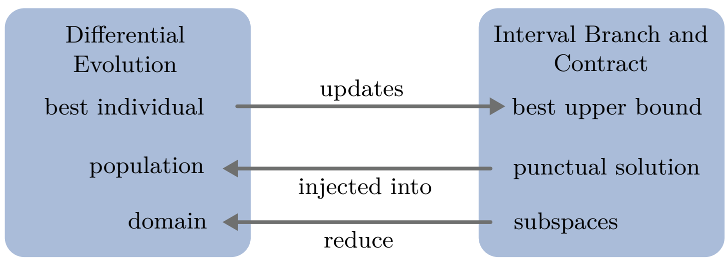


Fig. 2: Cooperative scheme of Charibde

The cooperation boils down to three main steps:

1. whenever the DE improves its best evaluation, the best individual and its evaluation are sent to the IBC to update the best known upper bound $\tilde{f}$;
2. whenever the IBC finds a better punctual solution (e.g. the center of a box), it is injected into DE's population;
3. the exploration strategy MaxDist periodically reduces the search-space of DE, then regenerates the population in the new search-space.

Sections 4.2 and 4.3 detail the particular implementations of the DE (Algorithm 1) and the IBC (Algorithm 2) within Charibde.

4.2 Differential Evolution

Population-based metaheuristics, in particular DE, are endowed with mechanisms that help escape local minima. They are quite naturally recommended to solve difficult multimodal problems for which traditional methods struggle to converge. They are also capable of generating feasible solutions without any a priori knowledge of the topology. DE has proven greatly beneficial for improving the best known upper bound $\tilde{f}$, a task for which standard branch and bound algorithms are not intrinsically intended.

Algorithm 1 Charibde: Differential Evolution

```
function DifferentialEvolution(f: objective function, C: system of constraints, D: search-space, NP: size of population, W: amplitude factor, CR: crossover rate)
    P ← initial population, randomy generated in D
    f̃ ← +∞
    repeat
        (x, f_x) ← MPI_ReceiveIBC()
        Insert x into P
        f̃ ← f_x
        Generate temporary population P′ by crossover
        P ← P′
        (x_best, f_best) ← BestIndividual(P)
        if f_best < f̃ then
            f̃ ← f_best
            MPI_SendIBC(x_best, f_best)
        end if
    until termination criterion is met
    return best individual of P
end function
```

Algorithm 2 Charibde: Interval Branch and Contract

```
function IntervalBranchAndContract(F: objective function, C: system of constraints, D: search-space, ε: tolerance)
    f̃ ← +∞                                            ▷ best known upper bound
    Q ← {D}                                            ▷ priority queue
    while Q ≠ ∅ do
        (x_DE, f_DE) ← MPI_ReceiveDE()
        f̃ ← min(f̃, f_DE)
        Extract a box X from Q
        Contract X w.r.t. constraints                  ▷ Algorithm 3
        if X cannot be discarded then
            if overline(F(m(X))) < f̃ then              ▷ midpoint test
                f̃ ← overline(F(m(X)))                  ▷ update best upper bound
                MPI_SendDE(m(X), overline(F(m(X))))
            end if
            Split X into {X_1, X_2}
            Insert {X_1, X_2} into Q
        end if
    end while
    return (f̃, x̃)
end function
```

Base Individual In the standard DE strategy, all the current individuals have the same probability to be selected as the base individual $\boldsymbol{u}$. We opted for an alternative strategy [23] that guarantees that all individuals of the population play this role once and only once at each generation: the index of the base individual is obtained by summing the index of the individual $\boldsymbol{x}$ and an offset in $\{1, \dots, NP-1\}$, drawn with uniform probability.

Bound Constraints When a coordinate y_i of $\boldsymbol{y}$ (computed during the crossover) exceeds the bounds of the component D_i of the domain $\boldsymbol{D}$, the bounce-back method [23] replaces y_i with a valid coordinate y'_i that lies between the base coordinate u_i and the violated bound:

$$y'_i = \begin{cases} u_i + \omega(\overline{D_i} - u_i) & \text{if } y_i > \overline{D_i} \\ u_i + \omega(\underline{D_i} - u_i) & \text{if } y_i < \underline{D_i} \end{cases} \tag{5}$$

where ω is drawn in $[0, 1]$ with uniform probability.

Constraint Handling The extension of evolutionary algorithms to constrained optimization has been addressed by numerous authors. We implemented the direct constraint handling [23] that assigns to each individual a vector of evaluations (objective function and constraints), and selects the new individual $\boldsymbol{y}$ (see Section 2) based upon simple rules:

- $\boldsymbol{x}$ and $\boldsymbol{y}$ are feasible and $\boldsymbol{y}$ has a lower or equal objective value than $\boldsymbol{x}$;
- $\boldsymbol{y}$ is feasible and $\boldsymbol{x}$ is not;
- $\boldsymbol{x}$ and $\boldsymbol{y}$ are infeasible, and $\boldsymbol{y}$ does not violate any constraint more than $\boldsymbol{x}$.

Rigorous Feasibility Numerous NLP solvers tolerate a slight violation (relaxation) of the inequality constraints (e.g. $g \leq 10^{-6}$ instead of $g \leq 0$). The evaluation of a "pseudo-feasible" solution $\boldsymbol{x}$ (that satisfies such relaxed constraints) is not a rigorous upper bound of the global minimum; roundoff errors may even lead to absurd conclusions: $f(\boldsymbol{x})$ may be lower than the global minimum, and (or) $\boldsymbol{x}$ may be very distant from actual feasible solutions in the search-space.

To ensure that an individual $\boldsymbol{x}$ is numerically feasible (i.e. that the evaluations of the constraints are non-positive), we evaluate the constraints g_i using interval arithmetic. $\boldsymbol{x}$ is considered as feasible when the interval evaluations $G_i(\boldsymbol{x})$ are non-positive, that is $\forall i \in \{1, \dots, m\}, \overline{G_i(\boldsymbol{x})} \leq 0$.

Rigorous Objective Function When $\boldsymbol{x}$ is a feasible point, the evaluation $f(\boldsymbol{x})$ may be subject to roundoff errors; the only reliable upper bound of the global minimum available is $\overline{F(\boldsymbol{x})}$ (the right bound of the interval evaluation). However, evaluating the individuals using only interval arithmetic is much costlier than cheap floating-point arithmetic.

An efficient in-between solution consists in systematically computing the floating-point evaluations $f(\boldsymbol{x})$, and computing the interval evaluation $F(\boldsymbol{x})$ whenever the best known "round to nearest" evaluation is improved. $\overline{F(\boldsymbol{x})}$ is then compared to the best known reliable upper bound $\tilde{f}$: if $\tilde{f}$ is improved, $\overline{F(\boldsymbol{x})}$ is sent to the IBC. This implementation greatly reduces the cost of evaluations, while ensuring that all the values sent to the IBC are rigorous.

4.3 Interval Branch and Contract

Branching aims at refining the computation of lower bounds of the functions using interval arithmetic. Two strategies may be found in the early literature:

- the variable with the largest domain is bisected;
- the variables are bisected one after the other in a round-robin scheme.

More recently, the Smear heuristic [10] has emerged as a competitive alternative to the two standard strategies. The variable x_i for which the interval quantity $\frac{\partial F}{\partial x_i}(\boldsymbol{X})(X_i - x_i)$ is the largest is bisected.

Charibde's *main contractor* is detailed in Algorithm 3. We exploit the contracted nodes of HC4Revise to compute partial derivatives via automatic differentiation [26]. HC4Revise is a revise procedures within a quasi-fixed point with tolerance $\eta \in [0, 1]$: the propagation loop stops when the box $\boldsymbol{X}$ is not sufficiently contracted, i.e. when the size of $\boldsymbol{X}$ becomes larger than a fraction ηw_0 of the initial size w_0. Most contractors include an evaluation phase that yields a lower bound of the problem on the current box. Charibde thus computes several lower bounds (natural, Taylor, LP) as long as the box is not discarded. Charibde calls ocaml-glpk [18], an OCaml binding for GLPK (GNU Linear Programming Kit). Since the solution of the linear program is computed using floating-point arithmetic, it may be subject to roundoff errors. A cheap postprocessing step [21] computes a rigorous bound on the optimal solution of the linear program, thus providing a rigorous lower bound of the initial problem.

Algorithm 3 Charibde: contractor for constrained optimization

```
function CONTRACTION(in-out X: box, F: objective function, in-out f̃: best upper bound, in-out C: system of constraints)
    lb ← −∞                                              ▷ lower bound
    repeat
        w0 ← w(X)                                        ▷ initial size
        F_X ← HC4REVISE(F(X) ≤ f̃)                        ▷ evaluation of f/contraction
        lb ← F_X (lower bound)                           ▷ lower bound by natural form
        G ← ∇F(X)                                        ▷ gradient by AD
        lb ← max(lb, SECONDORDER(X, F, f̃, G))            ▷ second-order form
        C ← HC4(X, C, η)                                 ▷ quasi-fixed point with tolerance η
        if use linearization then
            lb ← max(lb, LINEARIZATION(X, F, f̃, G, C))   ▷ simplex or X-Newton
        end if
    until X = ∅ ou w(X) > η w0
    return lb
end function
```

When the problem is subject to *equality constraints* h_j ($j \in \{1, \ldots, p\}$), IBBA [22], Ibex [30] and Charibde handle a relaxed problem where each equality constraint $h_j(\boldsymbol{x}) = 0$ ($j \in \{1, \dots, p\}$) is replaced by two inequalities:

$$-\varepsilon_= \leq h_j(\boldsymbol{x}) \leq \varepsilon_= \tag{6}$$

$\varepsilon_=$ may be chosen arbitrarily small.

4.4 MaxDist: a New Exploration Strategy

The boxes that cannot be discarded are stored in a priority queue $\mathcal{Q}$ to be processed at a later stage. The order in which the boxes are extracted determines the exploration strategy of the search-space ("best-first", "largest first", "depth-first"). Numerical tests suggest that

- the "best-first" strategy is rarely relevant because of the overestimated range (due to the dependency problem);
- the "largest first" strategy does not give advantage to promising regions;
- the "depth-first" strategy tends to quickly explore the neighborhood of local minima, but struggles to escape from them.

We propose a new exploration strategy called MaxDist. It consists in extracting from $\mathcal{Q}$ the box that is *the farthest from the current solution* $\tilde{\boldsymbol{x}}$. The underlying ideas are to

- explore the neighborhood of the global minimizer (a tedious task when the objective function is flat in this neighborhood) only when the best possible upper bound is available;
- explore regions of the search-space that are hardly accessible by the DE algorithm.

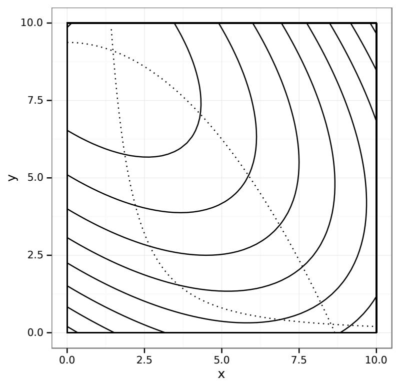
The distance between a point $\boldsymbol{x}$ and a box $\boldsymbol{X}$ is the sum of the distances between each coordinate x_i and the closest bound of X_i. Note that MaxDist is an adaptive heuristic: whenever the best known solution $\tilde{\boldsymbol{x}}$ is updated, $\mathcal{Q}$ is reordered according to the new priority of the boxes.

Preliminary results (not presented here) suggest that MaxDist is competitive with standard strategies. However, the most interesting observation lies in the behavior of $\mathcal{Q}$: when using MaxDist, the maximum size of $\mathcal{Q}$ (the maximum number of boxes simultaneously stored in $\mathcal{Q}$) remains remarkably low (a few dozens compared to several thousands for standard strategies). This offers promising perspectives for the cooperation between DE and IBC: the remaining boxes of the IBC may be exploited in the DE to avoid exploring regions of the search-space that have already been proven infeasible or suboptimal.

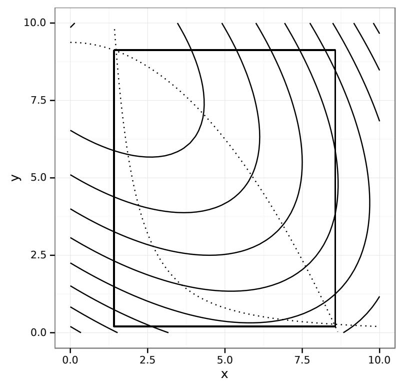
The following numerical example illustrates how the remaining boxes are exploited to reduce DE's domain through the generations. Let

$$\begin{aligned} \min_{(x,y)\in(X,Y)} \quad & -\frac{(x+y-10)^2}{30} - \frac{(x-y+10)^2}{120} \\ \text{s.t.} \quad & \frac{20}{x^2} - y \leq 0 \\ & x^2 + 8y - 75 \leq 0 \end{aligned} \tag{7}$$

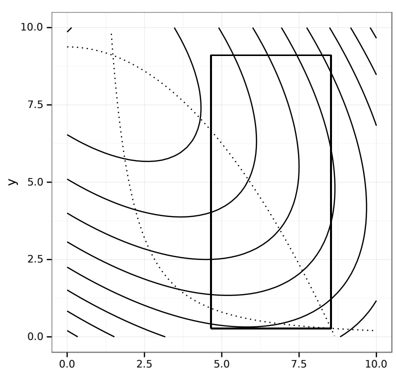
be a constrained optimization problem defined on the box $X \times Y = [0, 10] \times [0, 10]$ (Figure 3a). The dotted curves represent the frontiers of the two inequality constraints, and the contour lines of the objective function are shown in solid dark. The feasible region is the banana-shaped set, and the global minimizer is located in its lower right corner.



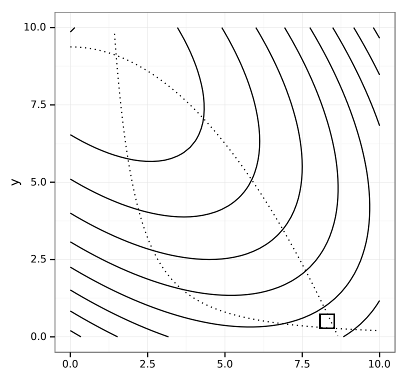
(a) Initial domain



(b) Contracted initial domain



(c) Convex hull at generation 10



(d) Convex hull at generation 20

Fig. 3: Evolution of DE's domain with the number of generations

The initial domain of DE (which corresponds to the initial box in the IBC) is first contracted with respect to the constraints of the problem. The initial population of DE is then generated within this contracted domain, thus avoiding obvious infeasible regions of the search-space. This approach is similar to that of [11]. Figure 3b depicts the contraction (the black rectangle) of the initial domain with respect to the constraints (sequentially by HC4Revise): $X \times Y = [1.4142, 8.5674] \times [0.2, 9.125]$.

Periodically, we compute the convex hull $\square(\mathcal{Q})$ of the remaining boxes of $\mathcal{Q}$ and replace DE's domain with $\square(\mathcal{Q})$. Note that

1. the convex hull (linear complexity) may be computed at low cost, because the size of $\mathcal{Q}$ remains small when using MaxDist;
2. by construction, MaxDist handles boxes on the rim of the remaining domain (the boxes of $\mathcal{Q}$), which boosts the reduction of the convex hull.

Figures 3c and 3d represent the *convex hull* $\square(\mathcal{Q})$ of the remaining subboxes in the IBC, respectively after 10 and 20 DE generations. The population is then randomly regenerated within the new contracted domain $\square(\mathcal{Q})$. The convex hull operation progressively eliminates local minima and infeasible regions. The global minimum eventually found by Charibde with precision $\varepsilon = 10^{-8}$ is $\tilde{f} = f(8.532424, 0.274717) = -2.825296148$; both constraints are active.

5 Experimental Results

Currently, GlobSol [15], IBBA [22] and Ibex [8] are among the most efficient solvers in rigorous constrained optimization. They share a common skeleton of interval branch and bound algorithm, but differ in the acceleration techniques. GlobSol uses the reformulation-linearization technique (RLT), that introduces new auxiliary variables for each intermediary operation. IBBA calls a contractor similar to HC4Revise, and computes a relaxation of the system of constraints using affine arithmetic. Ibex is dedicated to both numerical CSPs and constrained optimization; it embeds most of the aforementioned contractors (HC4, 3B, Mohc, CID, X-Newton). Couenne [5] and BARON [25] are state-of-the-art NLP solvers. They are based on a non-rigorous spatial branch and bound algorithms, in which the objective function and the constraints are over- and underestimated by convex relaxations. Although they perform an exhaustive exploration of the search-space, they cannot guarantee a given precision on the value of the optimum.

All five solvers and Charibde are compared on a subset of 11 COCONUT constrained problems (Table 1), extracted by Araya [3] for their difficulty: ex2_1_7, ex2_1_9, ex6_2_6, ex6_2_8, ex6_2_9, ex6_2_11, ex6_2_12, ex7_2_3, ex7_3_5, ex14_1_7 and ex14_2_7. Because of numerical instabilities of the ocaml-glpk LP library ("assert failure"), the results of the problems ex6_1_1, ex6_1_3 and ex_6_2_10 are not presented. The second and third columns give respectively the number of variables n and the number of constraints m. The fourth (resp. fifth) column specifies the type of the objective function (resp. the constraints): L is linear, Q is quadratic and NL is nonlinear. The logsize of the domain $\boldsymbol{D}$ (sixth column) is $\log(\prod_{i=1}^{n}(\overline{D_i} - \underline{D_i}))$.

The comparison of CPU times (in seconds) for solvers GlobSol, IBBA, Ibex, Couenne, BARON and Charibde on the benchmark of 11 problems is detailed in Table 2. Mean times and standard deviations (in brackets) are given for Charibde over 100 runs. The numerical precision on the objective function $\varepsilon = 10^{-8}$ and the tolerance for equality constaints $\varepsilon_= = 10^{-8}$ were identical for all solvers. TO (timeout) indicates that a solver could not solve a problem within one hour. The results of GlobSol (proprietary piece of software) were not available for all problems; only those mentioned in [22] are presented. The results of IBBA were also taken from [22]. The results of Ibex were taken from [3]: only the best strategy (simplex, X-NewIter or X-Newton) for each benchmark problem is presented. Couenne and BARON (only the commercial version of the code is available) were run on the NEOS server [13].

Table 1: Description of difficult COCONUT problems

Problem	n	m	Type f	g_i, h_j	Domain logsize
ex2_1_7	20	10	Q	L	$+\infty$
ex2_1_9	10	1	Q	L	$+\infty$
ex6_2_6	3	1	NL	L	$-3 \cdot 10^{-6}$
ex6_2_8	3	1	NL	L	$-3 \cdot 10^{-6}$
ex6_2_9	4	2	NL	L	-2.77
ex6_2_11	3	1	NL	L	$-3 \cdot 10^{-6}$
ex6_2_12	4	2	NL	L	-2.77
ex7_2_3	8	6	L	NL	61.90
ex7_3_5	13	15	L	NL	$+\infty$
ex14_1_7	10	17	L	NL	23.03
ex14_2_7	6	9	L	NL	$+\infty$

Table 2: Comparison of convergence times (in seconds) between GlobSol, IBBA, Ibex, Charibde (mean and standard deviation over 100 runs), Couenne and BARON on difficult constrained problems

	Rigorous				Non rigorous	
Problem	GlobSol	IBBA	Ibex	Charibde	Couenne	BARON
ex2_1_7		16.7	**7.74**	34.9 (13.3)	476	16.23
ex2_1_9		154	9.07	35.9 (0.29)	**3.01**	3.58
ex6_2_6	306	1575	136	**3.3** (0.41)	TO	5.7
ex6_2_8	204	458	59.3	2.9 (0.37)	TO	TO
ex6_2_9	463	523	25.2	**2.7** (0.03)	TO	TO
ex6_2_11	273	140	7.51	**1.96** (0.06)	TO	TO
ex6_2_12	196	112	22.2	**8.8** (0.17)	TO	TO
ex7_2_3		TO	544	**1.9** (0.30)	TO	TO
ex7_3_5		TO	28.91	**4.5** (0.09)	TO	4.95
ex14_1_7		TO	406	4.2 (0.13)	13.86	**0.56**
ex14_2_7		TO	66.39	0.2 (0.04)	**0.01**	0.02
Sum	> 1442	TO	1312.32	**101.26**	TO	TO

Charibde was run on an Intel Xeon E31270 @ 3.40GHz x 8 with 7.8 GB of RAM. BARON and Couenne were run on 2 Intel Xeon X5660 @ 2.8GHz x 12 with 64 GB of RAM. IBBA and Ibex were run on similar processors (Intel x86, 3GHz). The difference in CPU time between computers is about 10% [4], which makes the comparison quite fair.

The hyperparameters of Charibde for the benchmark problems are given in Table 3; NP is the population size, and η is the quasi-fixed point ratio. The amplitude $W = 0.7$, the crossover rate $CR = 0.9$ and the MaxDist strategy are common to all problems. Tuning the hyperparameters is generally problem-dependent, and requires structural knowledge about the problem: the population size NP may be set according to the dimension and the number of local minima, the crossover rate CR is related to the separability of the problem, and the techniques based on linear relaxation have little influence for problems with few constraints, but are cheap when the constraints are linear.

Table 3: Hyperparameters of Charibde for the benchmark problems

Problem	NP	Bisections	Fixed-point ratio (η)	LP	X-Newton
ex2_1_7	20	RR	0.9	✓	✓
ex2_1_9	100	RR	0.8	✓	
ex6_2_6	30	Smear	0	✓	
ex6_2_8	30	Smear	0	✓	
ex6_2_9	70	Smear	0		
ex6_2_11	35	Smear	0		
ex6_2_12	35	RR	0	✓	
ex7_2_3	40	Largest	0	✓	✓
ex7_3_5	30	RR	0	✓	
ex14_1_7	40	RR	0	✓	
ex14_2_7	40	RR	0	✓	

Charibde outperforms Ibex on 9 out of 11 problems, IBBA on 10 out of 11 problems and GlobSol on all the available problems. The cumulated CPU time shows that Charibde (101.26s) improve the performances of Ibex (1312.32s) by an order of magnitude (ratio: 13) on this benchmark of 11 difficult problems. Charibde also proves highly competitive with *non-rigorous* solvers Couenne and BARON. The latter are faster or have similar CPU times on some of the 11 problems, however they both time out on at least five problems (seven for Couenne, five for BARON). Overall, Charibde seems more robust and solves all the problems of the benchmark, while providing a numerical proof of optimality. Surprisingly, the convergence times do not seem directly related to the dimensions of the instances. They may be explained by the nature of the objective function and constraints (in particular, Charibde seems to struggle when the objective function is quadratic) and the dependency induced by the multiple occurrences of the variables.

Table 4 presents the best upper bounds obtained by Charibde, Couenne and BARON at the end of convergence (precision reached or timeout). Truncated digits on the upper bounds are bounded (e.g. 1.23_7^8 denotes $[1.237, 1.238]$ and -1.23_7^8 denotes $[-1.238, -1.237]$). The incorrect digits of the global minima obtained by Couenne and BARON are underlined. This demonstrates that non-rigorous solvers may be affected by roundoff errors, and may provide solutions that are infeasible or have an objective value *lower than the global minimum* (Couenne on ex2_1_9, BARON on ex2_1_7, ex2_1_9, ex6_2_8, ex6_2_12, ex7_2_3 and ex7_3_5). For the most difficult instance ex7_2_3, Couenne is not capable of finding a feasible solution with a satisfactory evaluation within one hour. It would be very informative to compute the ratio between the size of the feasible domain (the set of all feasible points) and the size of the entire domain. On the other hand, the strategy MaxDist within Charibde greatly contributes to finding an excellent upper bound of the global minimum, which significantly accelerates the interval pruning phase.

Table 4: Best upper bounds obtained by Charibde, Couenne and BARON

Problem	Charibde	Couenne	BARON
ex2_1_7	-4150.41013392_8^9	$-4150.4101\underline{2731}_7^8$	$-4150.4101\underline{6079}_7^8$
ex2_1_9	-0.3750000075	$-0.37500\underline{15}_3^4$	$-0.37500\underline{111}_0^1$
ex6_2_6	-0.00000260_2^3	0.00000071_0^1	-0.00000260_2^3
ex6_2_8	-0.0270063_{49}^{50}	-0.0270063_{49}^{50}	$-0.0270063\underline{7}_0^1$
ex6_2_9	-0.03406618_4^5	-0.034066184	-0.03406619_0^1
ex6_2_11	-0.00000267_2^3	-0.00000267_2^3	-0.00000267_2^3
ex6_2_12	0.2891947_{39}^{40}	0.28919475	$0.28919\underline{169}_8^9$
ex7_2_3	7049.24802052_8^9	10^{50}	$7049.\underline{02029170}_6^7$
ex7_3_5	1.20671699_1^2	1.2068965	$\underline{0.23982448}_7^8$
ex14_1_7	0.0000000_{09}^{10}	0.00000000_0^1	0
ex14_2_7	0.00000000_7^8	0.00000000_0^1	0

6 Conclusion

We proposed a cooperative hybrid solver Charibde, in which a deterministic interval branch and contract cooperates with a stochastic differential evolution algorithm. The two independent algorithms run in parallel and exchange bounds, solutions and search-space in an advanced manner via message passing. The domain of the population-based metaheuristic is periodically reduced by removing local minima and infeasible regions detected by the branch and bound.

A comparison of Charibde with state-of-the-art interval-based solvers (GlobSol, IBBA, Ibex) and NLP solvers (Couenne, BARON) on a benchmark of difficult COCONUT problems shows that Charibde is highly competitive against non-rigorous solvers (while bounding the global minimum) and converges faster than rigorous solvers by an order of magnitude.

References

1. Alliot, J.M., Durand, N., Gianazza, D., Gotteland, J.B.: Finding and proving the optimum: Cooperative stochastic and deterministic search. In: 20th European Conference on Artificial Intelligence (ECAI 2012), August 27-31, 2012, Montpellier, France (2012)
2. Araya, I., Trombettoni, G., Neveu, B.: Exploiting monotonicity in interval constraint propagation. In: Proc. AAAI. pp. 9–14 (2010)
3. Araya, I., Trombettoni, G., Neveu, B.: A contractor based on convex interval taylor. In: Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems, pp. 1–16. Springer (2012)
4. Araya, I., Trombettoni, G., Neveu, B., Chabert, G.: Upper bounding in inner regions for global optimization under inequality constraints. Journal of Global Optimization 60(2), 145–164 (2014)
5. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tighteningtechniques for non-convex minlp. Optimization Methods & Software 24(4-5), 597–634 (2009)
6. Benhamou, F., Goualard, F., Granvilliers, L., Puget, J.F.: Revising hull and box consistency. In: International Conference on Logic Programming. pp. 230–244. MIT press (1999)
7. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11(6), 4135–4151 (2011)
8. Chabert, G., Jaulin, L.: Contractor programming. Artificial Intelligence 173, 1079–1100 (2009)
9. Cotta, C., Troya, J.M.: Embedding branch and bound within evolutionary algorithms. Applied Intelligence 18(2), 137–153 (2003)
10. Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM Journal on Numerical Analysis 34(3), 922–938 (1997)
11. Focacci, F., Laburthe, F., Lodi, A.: Local search and constraint programming. In: Handbook of metaheuristics, pp. 369–403. Springer (2003)
12. Gallardo, J.E., Cotta, C., Fernández, A.J.: On the hybridization of memetic algorithms with branch-and-bound techniques. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 37(1), 77–83 (2007)
13. Gropp, W., Moré, J.: Optimization environments and the NEOS server. Approximation theory and optimization pp. 167–182 (1997)
14. Hansen, E.: Global optimization using interval analysis. Dekker (1992)
15. Kearfott, R.B.: Rigorous global search: continuous problems. Springer (1996)
16. Leroy, X., Doligez, D., Frisch, A., Garrigue, J., Rémy, D., Vouillon, J.: The objective caml system release 3.12. Documentation and userâĂŹs manual. INRIA (2010)
17. Lhomme, O.: Consistency techniques for numeric csps. In: IJCAI. vol. 93, pp. 232–238. Citeseer (1993)
18. Mimram, S.: ocaml-glpk. http://ocaml-glpk.sourceforge.net/ (2004)
19. Moore, R.E.: Interval Analysis. Prentice-Hall (1966)
20. Moore, R.E.: On computing the range of a rational function of n variables over a bounded region. Computing 16(1), 1–15 (1976)
21. Neumaier, A., Shcherbina, O.: Safe bounds in linear and mixed-integer linear programming. Mathematical Programming 99(2), 283–296 (2004)
22. Ninin, J., Hansen, P., Messine, F.: A reliable affine relaxation method for global optimization. Groupe d'études et de recherche en analyse des décisions (2010)

23. Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global Optimization. Natural Computing, Springer-Verlag (2006)
24. Puchinger, J., Raidl, G.R.: Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In: Artificial intelligence and knowledge engineering applications: a bioinspired approach, pp. 41–53. Springer (2005)
25. Sahinidis, N.V.: Baron: A general purpose global optimization software package. Journal of Global Optimization 8(2), 201–205 (1996)
26. Schichl, H., Neumaier, A.: Interval analysis on directed acyclic graphs for global optimization. Journal of Global Optimization 33(4), 541–562 (2005)
27. Skelboe, S.: Computation of rational interval functions. BIT Numerical Mathematics 14(1), 87–95 (1974)
28. Sotiropoulos, D., Stavropoulos, E., Vrahatis, M.: A new hybrid genetic algorithm for global optimization. Nonlinear Analysis: Theory, Methods & Applications 30(7), 4529–4538 (1997)
29. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization pp. 341–359 (1997)
30. Trombettoni, G., Araya, I., Neveu, B., Chabert, G.: Inner regions and interval linearizations for global optimization. In: AAAI (2011)
31. Trombettoni, G., Chabert, G.: Constructive interval disjunction. In: Principles and Practice of Constraint Programming–CP 2007, pp. 635–650. Springer (2007)
32. Van Hentenryck, P.: Numerica: a modeling language for global optimization. MIT press (1997)
33. 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 (EA-2013). pp. 84–94 (2013)
34. Zhang, X., Liu, S.: A new interval-genetic algorithm. In: Natural Computation, 2007. ICNC 2007. Third International Conference on. vol. 4, pp. 193–197. IEEE (2007)