A Fast and Reliable Hybrid Algorithm for Numerical Nonlinear Global Optimization

Charlie Vanaret and **Jean-Baptiste Gotteland** and **Nicolas Durand**
Laboratoire de Mathématiques Appliquées, Informatique, Automatique pour l'Aérien
École Nationale de l'Aviation Civile, Toulouse, France
{vanaret, gottelan, durand}@recherche.enac.fr

Jean-Marc Alliot
Institut de Recherche en Informatique de Toulouse, France
jean-marc.alliot@irit.fr

Abstract

Highly nonlinear and ill-conditioned numerical optimization problems take their toll on the convergence of existing resolution methods. Stochastic methods such as Evolutionary Algorithms carry out an efficient exploration of the search-space at low cost, but get often trapped in local minima and do not prove the optimality of the solution. Deterministic methods such as Interval Branch and Bound algorithms guarantee bounds on the solution, yet struggle to converge within a reasonable time on high-dimensional problems. The contribution of this paper is a hybrid algorithm in which a Differential Evolution algorithm and an Interval Branch and Contract algorithm cooperate. Bounds and solutions are exchanged through shared memory to accelerate the proof of optimality. It prevents premature convergence toward local optima and outperforms both deterministic and stochastic existing approaches. We demonstrate the efficiency of this algorithm on two currently unsolved problems: first by presenting new certified optimal results for the Michalewicz function for up to 75 dimensions and then by proving that the putative minimum of Lennard-Jones clusters of 5 atoms is optimal.

1 Motivation

Evolutionary Computation (EC) algorithms have been widely used by the global optimization community for their ability to handle complex and high-dimensional problems with no assumption on continuity or differentiability. They carry out a fast exploration of the search-space and generally converge toward satisfactory solutions. However, EC may get trapped in local optima and provide suboptimal solutions. Moreover, their convergence remains hard to control due to their stochastic nature. On the other hand, Interval Branch and Bound Algorithms (IBBA) guarantee rigorous bounds on the solutions to nonlinear and nonconvex numerical problems but are limited by their exponential complexity regarding the number of variables and by the dependency problem inherent to Interval Analysis.

Few approaches attempted to hybridize EC algorithms and Interval Branch and Bound algorithms. (Sotiropoulos, Stavropoulos, and Vrahatis 1997) and (Zhang and Liu 2007) devised *integrative* methods that embedded one algorithm within the other: Sotiropoulos et al. used a Branch and Bound algorithm to reduce the domain to a list of ϵ-large subspaces. A Genetic Algorithm (GA) was then initialized within each subspace to improve the upper bound of the global minimum. Zhang and Liu used a Genetic Algorithm within the Branch and Bound algorithm to improve the bounds and the order of the list of remaining subspaces to process. (Alliot et al. 2012) proposed a *cooperative* approach combining the efficiency of a GA and the reliability of IBBA to guarantee the optimality of solutions to highly nonlinear bound-constrained problems. Original optimal results were achieved on benchmark functions, demonstrating the validity of the approach. However, local monotonicity and constraint programming techniques, which exploit the analytical form of the objective function, were left out in the basic formulation of the algorithm. In this paper, we propose an advanced cooperative algorithm in which a Differential Evolution algorithm cooperates with Interval Constraint Programming. It is reliable as it guarantees bounds on the global optimum. New optimal results achieved on two highly multimodal functions – the Michalewicz function and the Lennard-Jones cluster problem – attest the substantial gain in performance. In this study, we consider only bound-constrained numerical minimization problems:

$$\min_{x \in \mathbf{D}} f(x) \tag{1}$$

where $f : (\mathbf{D} = \prod_{i=1}^{n} [l_i, u_i] \subset \mathbb{R}^n) \to \mathbb{R}$ is the objective function to be minimized. We assume that f is differentiable, and that the analytical forms of f and its partial derivatives are available.

The standard Differential Evolution algorithm is presented in section 2, and details about our Interval Branch and Contract Algorithm are given in section 3. The implementation of our hybrid algorithm is detailed in section 4. In section 5, we present new optimal results for the Michalewicz function and we prove that the putative minimum for the Lennard-Jones cluster of 5 atoms is optimal.

2 Differential Evolution

Evolutionary Computations (EC) are stochastic iterative optimization algorithms that usually mimic natural processes. In particular, Evolutionary Algorithms (EA) are

based on the theory of evolution (survival of the fittest) where stochastic operators iteratively improve a population of individuals (candidate solutions) according to an adaptation criterion (the objective function), in order to converge toward satisfactory solutions. Numerous EC techniques have recently emerged: Particle Swarm Optimization (Kennedy and Eberhart 1995), Ant Colony Optimization (Dorigo, Maniezzo, and Colorni 1996), CMA-ES (Hansen and Kern 2004), etc. (Alliot et al. 2012) used a Genetic Algorithm, but did clearly state that this algorithm had been chosen because it was the most familiar to the authors, but that other EC algorithms might be more adapted to the hybridization. In the following, we use a Differential Evolution algorithm as it greatly enhanced the original results.

Differential Evolution (DE) is a simple yet powerful EC algorithm introduced by Storn and Price (1997). It has proved to be particularly efficient on difficult black-box optimization problems. DE combines the coordinates of existing individuals with a particular probability to generate new potential solutions. Unlike Genetic Algorithms, it offers the possibility to move individuals around in only part of the dimensions. Algorithm 1 describes DE for a n-dimensional problem.

Algorithm 1 Differential evolution algorithm (DE)

```
Randomly initialize each individual
Evaluate each individual
while termination criterion is not met do
    for each individual x^(i), i ∈ {1,...,NP} do
        Randomly pick r, x^(i0), x^(i1), x^(i2)
        for each dimension j ∈ {1,...,n} do
            if j = r or rand(0,1) < CR then
                y_j^(i) = x_j^(i0) + W × (x_j^(i1) − x_j^(i2))
            else
                y_j^(i) = x_j^(i)
            end if
        end for
        if f(y^(i)) < f(x^(i)) then
            x^(i) ← y^(i)
        end if
    end for
end while
return the best individual(s) of the population
```

NP is the population size, $W > 0$ is the weighting factor and $CR \in [0, 1]$ is the crossover rate. r is a random index picked in $\{1, \ldots, n\}$ at each generation to ensure that at least $y_r^{(i)}$ differs from $x_r^{(i)}$. The algorithm may be stopped after a fixed number of iterations or when many successive iterations do not produce better results.

3 Interval Branch and Contract algorithm

Interval analysis (IA) was introduced by (Moore 1966) to bound round-off errors due to the use of floating-point arithmetic. A compact interval with floating-point bounds is defined by $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$.

In the following, capital letters represent interval quantities (interval X) and bold letters represent vectors (box $\mathbf{X}$, vector $\mathbf{x}$). $\mathbb{IR}$ is the set of all intervals. The lower (resp. upper) bound of an interval X is noted $\underline{X}$ (resp. $\overline{X}$). A *box* denotes an interval vector. The width of an interval is $w(X) = \overline{X} - \underline{X}$, and the width of a box $(X_1, \ldots, X_n)$ is $\max_{1 \leq i \leq n} w(X_i)$. The midpoint of an interval is $m(X) = \frac{1}{2}(\underline{X} + \overline{X})$ and the midpoint of a box $(X_1, \ldots, X_n)$ is $(m(X_1), \ldots, m(X_n))$. The convex $hull$ of a set of boxes S is the smallest box that contains S.

Interval arithmetic extends to intervals binary operators $(+, -, \times, /)$ and elementary functions (exp, log, cos, etc.). Interval computations are carried out with outward rounding. An *interval extension* F of a real-valued function f guarantees a rigorous enclosure of its range:

$$\forall \mathbf{X} \in \mathbb{IR}^n, f(\mathbf{X}) = \{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}\} \subset F(\mathbf{X}) \qquad (2)$$

The *natural interval extension* F_N is computed by replacing real elementary operations by interval arithmetic operations.

IA generally computes a large overestimation of the image due to the dependency problem: when a variable appears more than once in an expression, each occurrence is treated as a different variable. However, if f is continuous inside a box, its natural interval extension yields the exact range when each variable occurs only once in its expression:

Example 1 *Let $f_1(x) = x^2 - x$ and $X = [0, 2]$. The functions $f_2(x) = x(x - 1)$ and $f_3(x) = (x - \frac{1}{2})^2 - \frac{1}{4}$ are equivalent to f_1. However,*

$$\begin{aligned} F_1(X) &= [0,2]^2 - [0,2] = [0,4] - [0,2] = [-2,4] \\ F_2(X) &= [0,2] \times ([0,2] - 1) = [0,2] \times [-1,1] = [-2,2] \\ F_3(X) &= ([0,2] - \frac{1}{2})^2 - \frac{1}{4} = [-\frac{1}{2}, \frac{3}{2}]^2 - \frac{1}{4} = [-\frac{1}{4}, 2] \end{aligned} \qquad (3)$$

We thus have $F_3(X) \subset F_2(X) \subset F_1(X)$. $F_3(X)$ is the best computable enclosure, i.e. it is the exact range of $f(X)$.

Interval Branch and Bound Algorithms (IBBA) exploit the conservative properties of interval extensions to rigorously bound global optima of numerical optimization problems (Hansen 1992). However, their exponential complexity hinders the speed of convergence on large problems. The method consists in splitting the initial search-space into subspaces (branching) on which an interval extension F of the objective function f is evaluated (bounding). By keeping track of the best upper bound $\tilde{f}$ of the global minimum, boxes that certainly do not contain a global minimizer are discarded. Remaining boxes are stored in a priority queue $\mathcal{L}$ to be iteratively processed until the desired precision on the width of the box (ϵ_x) or its image (ϵ_f) is reached. The process is repeated until $\mathcal{L}$ is empty.

Given $\tilde{f}$ (the best known upper bound of the global minimum), the dynamic constraint $f \leq \tilde{f}$ may be set to reduce the boxes. To prove that a box cannot contain a local minimizer, **i)** if the box has a common bound with the domain, we check the monotonicity of f, or **ii)** we set the derivative of f to zero to find a stationary point. To this end, we make use of **Interval Constraint Programming** (ICP), that aims at solving systems of nonlinear equations and numerical optimization problems. Stemming from IA and ICP communities, filtering (contraction) algorithms (Chabert and Jaulin 2009) narrow the bounds of the variables without loss of solutions. The standard contraction algorithms `HC4` (Benhamou

et al. 1999), `Box` (Van Hentenryck 1997) and `Mohc` (Araya, Trombettoni, and Neveu 2010) compute a propagation loop to narrow the bounds of the variables with revised procedures that handle a single constraint. `HC4Revise` enforces hull consistency and computes the optimal contraction (under continuity assumption) when the constraint contains no multiple occurrences of a variable. The procedure is detailed in Example 2. `BoxNarrow` enforces box consistency and is optimal when the constraint contains multiple occurences of a single variable. `MohcRevise` combines `HC4Revise` and `BoxNarrow` algorithms under monotonicity assumption. It is optimal when the constraint is monotonic with respect to all variables.

Alliot et al. used an IBBA that merely determines whether a box contains a global minimizer. We integrate a contraction step based on `HC4Revise` to narrow the bounds of the variables (algorithm 2). G_i denotes an interval extension of $\frac{\partial f}{\partial x_i}$. The resulting **Interval Branch and Contract Algorithm** (IBCA) is described in Algorithm 3. The implementation of this framework is discussed in the next section.

Algorithm 2 Contraction Step

procedure CONTRACT($\mathbf{X}, \tilde{f}$)
  `HC4Revise`($F(\mathbf{X}) \leq \tilde{f}$)
  for $i \in \{1, \ldots, n\}$ **do**
    if X_i has a common bound with the domain **then**
      Check the sign of $G_i(\mathbf{X})$
    else
      `HC4Revise`($G_i(\mathbf{X}) = 0$)
    end if
  end for
end procedure

Algorithm 3 Interval Branch and Contract Algorithm

$\tilde{f} \leftarrow +\infty$ ▷ best found upper bound
$\mathcal{L} \leftarrow \{\mathbf{X}_0\}$ ▷ priority queue of boxes to process
$\mathcal{S} \leftarrow \{\}$ ▷ list of solutions
repeat
  Extract a box $\mathbf{X}$ from $\mathcal{L}$ ▷ selection rule
  Compute $F(\mathbf{X})$ ▷ bounding rule
  if $\underline{F(\mathbf{X})} \leq \tilde{f} - \epsilon_f$ **then** ▷ cut-off test
    CONTRACT($\mathbf{X}, \tilde{f}$) ▷ filtering algorithms
    $\tilde{f} \leftarrow \min(\overline{F(m(\mathbf{X}))}, \tilde{f})$ ▷ midpoint test
    if $w(\mathbf{X}) > \epsilon_x$ **and** $w(F(\mathbf{X})) > \epsilon_f$ **then**
      Bisect $\mathbf{X}$ into $\mathbf{X}_1$ and $\mathbf{X}_2$ ▷ branching rule
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{X}_1\} \cup \{\mathbf{X}_2\}$
    else
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{X}\}$ ▷ termination rule
    end if
  else
    Discard $\mathbf{X}$
  end if
until $\mathcal{L} = \varnothing$
return $\mathcal{S}$

`HC4Revise` (Example 2) carries out a double exploration of the syntax tree of a constraint to contract each occurrence of a variable. It consists in an evaluation (bottom-up) phase that computes the elementary operation of each node, and a backward (top-down) propagation phase using elementary projection (inverse) functions.

Example 2 *Let $2x = z - y^2$ be an equality constraint, with $x \in [0, 20]$, $y \in [-10, 10]$ and $z \in [0, 16]$. The subexpressions are represented by the nodes $n_1 = 2x$, $n_2 = y^2$ and $n_3 = z - n_2$.*

The evaluation phase (Figure 1) computes $n_1 = 2 \times [0, 20] = [0, 40]$, $n_2 = [-10, 10]^2 = [0, 100]$ and $n_3 = [0, 16] - [0, 100] = [-100, 16]$.

The propagation phase (Figure 2) starts by intersecting n_1 and n_3: $n_1' = n_3' = n_1 \cap n_3 = [0, 40] \cap [-100, 16] = [0, 16]$. The backward propagation using inverse functions yields:

$x' = x \cap \frac{n_1'}{2} = [0, 20] \cap [0, 8] = [0, 8]$,
$z' = z \cap (n_2 + n_3') = [0, 16] \cap ([0, 100] + [0, 16]) = [0, 16]$,
$n_2' = n_2 \cap (z' - n_3') = [0, 100] \cap ([0, 16] - [0, 16]) = [0, 16]$,
$y' = y \cap hull(-\sqrt{n_2'}, \sqrt{n_2'}) = [-10, 10] \cap [-4, 4] = [-4, 4]$.

The initial box $([0, 20], [-10, 10], [0, 16])$ has been reduced to $([0, 8], [-4, 4], [0, 16])$ without loss of solutions.

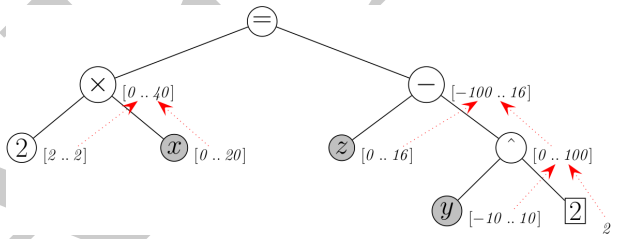

Figure 1: `HC4Revise`: evaluation phase

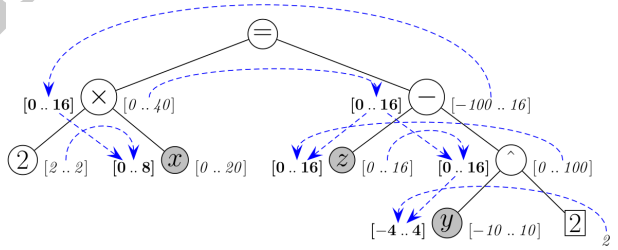

Figure 2: `HC4Revise`: propagation phase

4 Cooperative hybrid algorithm

The original cooperative algorithm (Alliot et al. 2012) combined an EA and an IBBA that ran independently, and cooperated by exchanging information through shared memory (Figure 3) in order to accelerate the convergence. The EA carries out a fast exploration of the search-space and quickly finds satisfactory solutions. The best known evaluation is used to improve the upper bound of the global minimum, and allows the IBBA to prune parts of the search-space more efficiently. A third process periodically projects EA's individuals trapped in local minima onto the closest admissible box of the IBBA.

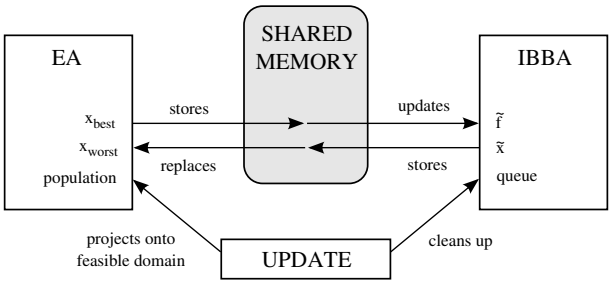

Figure 3: Original cooperative hybrid algorithm

We propose to replace the EA from the original paper by a DE algorithm that has shown a better convergence toward the global optimum. Our IBCA algorithm, integrating an efficient contraction step, replaces the IBBA.

4.1 DE thread

In the basic formulation of the algorithm, i_0, i_1 and i_2 are chosen at random in $\{1, \ldots, NP\}$, all distinct from each other and from i. We use a refinement suggested by (Price, Storn, and Lampinen 2006): the *random offset* method assigns a unique index $i_0 = (i + k) \bmod NP$, where k is an offset randomly chosen in $\{1, \ldots, NP\}$ at each generation. The DE has no termination criterion and stops only when the IBCA has reached convergence.

Two approaches deal with newly generated individuals that do not satisfy boundary constraints:

- a constant or adaptive amount (depending on the number and/or the magnitude of the violations) penalizes the evaluation of an out-of-bounds individual. This approach may slow down the convergence when numerous individuals are likely to violate the boundary constraints.
- the out-of-bounds components of an individual are reinitialized within the admissible bounds. A random reinitialization within the domain may prevent individuals to reach minimizers located near the bounds. We use the *bounce-back method* (Price, Storn, and Lampinen) to replace an out-of-bounds component y_j with a component that lies between $x_j^{(i_0)}$ and the admissible bound:

$$y_j^{(i)} = \begin{cases} x_j^{(i_0)} + rand(0,1)(l_i - x_j^{(i_0)}), & \text{if } y_j < l_j \\ x_j^{(i_0)} + rand(0,1)(u_i - x_j^{(i_0)}), & \text{if } y_j > u_j \end{cases} \quad (4)$$

DE's individuals are evaluated using the real-valued objective function in *round-to-nearest mode*. This value may be lower than the (theoretical) exact evaluation, therefore it should not be used to update $\tilde{f}$ at risk of losing a rigorous enclosure of the global minimum. An evaluation of the interval extension is therefore required whenever the best known evaluation is improved. The upper bound of the image interval computed by IA is stored in the shared memory, which guarantees certified bounds on the solution.

4.2 IBCA thread

A good upper bound provided by the DE is retrieved at each iteration from the shared memory and compared to the current best upper bound $\tilde{f}$. If the latter is improved, it is updated to prune more efficiently parts of the search-space that cannot contain a global minimizer. We use the following steps in our Interval Branch and Contract algorithm:

Selection rule The priority with which boxes are inserted in the queue $\mathcal{L}$ determines the exploration strategy of the search-space: smallest lower bound first, largest box (breadth-first search), stack (depth-first search). Our IBCA uses the location of the current best solution $\tilde{\mathbf{x}}$ to extract from $\mathcal{L}$ the box $\mathbf{X}$ for which the distance to $\tilde{\mathbf{x}}$ is maximum:

$$dist(\mathbf{X}, \tilde{\mathbf{x}}) = \sum_{i=1}^{n} \epsilon_i(X_i, \tilde{x}_i)^2 \quad (5)$$

$$\text{where } \epsilon_i(X_i, \tilde{x}_i) = \begin{cases} \tilde{x}_i - \overline{X_i}, & \text{if } \overline{X_i} < \tilde{x}_i \\ \underline{X_i} - \tilde{x}_i, & \text{if } \tilde{x}_i < \underline{X_i} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

It is worth noticing that the order in which the remaining boxes are processed has no incidence whatsoever on the convergence when the DE thread finds the global optimum, since the rest of the domain is to be exhaustively processed without updating $\tilde{f}$. However, we have observed that our heuristic maintains the lowest queue size S among the aforementioned strategies that limits the cost of insertion and extraction (both in $O(\log S)$) of boxes from the $\mathcal{L}$.

Midpoint test If $\overline{F(m(\mathbf{X}))}$ improves the best found upper bound $\tilde{f}$ of the global minimum, $\tilde{f}$ and $\tilde{x}$ are updated.

Cut-off test If $\tilde{f} < \underline{F(\mathbf{X})} - \epsilon_f$, $\mathbf{X}$ is discarded since it cannot contain a solution that improves $\tilde{f}$ by more than ϵ_f.

Bounding rule Computing a sharp lower bound of $f(\mathbf{X})$ is a crucial issue in Branch and Bound algorithms to determine whether $\mathbf{X}$ may contain a global minimizer. In the following, we introduce standard alternatives to the natural interval extension for bounding the range of a differentiable univariate function f. Let F' be an interval extension of f', X an interval and $x \in X$. The zeroth-order Taylor expansion with Lagrange remainder of $f(x)$ at the point $c \in X$ is $f(x) = f(c) + (x - c)f'(\xi)$. Since $\xi \in X$, $f'(\xi) \in f'(X) \subset F'(X)$. The expansion holds for all $x \in X$, therefore

$$f(X) \subset F(X) = f(c) + (X - c)F'(X) \quad (7)$$

In practice, $f(c)$ is also computed using IA to bound rounding errors. We denote F_T the multivariate **Taylor interval extension**, with $\mathbf{c} \in \mathbf{X}$:

$$F_T(\mathbf{X}) = F(\mathbf{c}) + \sum_{i=1}^{n} (X_i - c_i) G_i(X_1, \ldots, X_n) \quad (8)$$

The quality of inclusion of equation 8 depends on the point $\mathbf{c}$. The **Baumann centered form**, for which Baumann (1988) gave an analytical expression of the optimal center, computes the greatest lower bound of $f(X)$:

$$\underline{F_B(\mathbf{X})} = F(\mathbf{c}_B) + \sum_{i=1}^{n} w(X_i) \frac{L_i U_i}{U_i - L_i} \quad (9)$$

where $\frac{\partial f}{\partial x_i}(\mathbf{X}) \in G_i(\mathbf{X}) = [L_i, U_i]$, $\mathbf{c}_B = (c_1, \ldots, c_n)$ and $c_i = (\underline{X_i}U_i - \overline{X_i}L_i)/(U_i - L_i)$. Note that computing $F(\mathbf{c}_B)$ offers the possibility of updating $\tilde{f}$ (see midpoint test).

The order of approximation k of an interval extension F indicates the speed at which the interval inclusion approaches the exact range of f:

$$w(F(\mathbf{X})) - w(f(\mathbf{X})) = O(w(\mathbf{X})^k) \tag{10}$$

Centered forms have a quadratic order of approximation ($k = 2$), while that of natural interval extensions is only linear ($k = 1$). Consequently, the enclosure $F_B(\mathbf{X})$ of $f(\mathbf{X})$ becomes sharper when the width of $\mathbf{X}$ approaches 0. In practice, it is not known when the Baumann form computes a better inclusion than the natural interval extension. To exploit the quadratic convergence of F_B, we set an arbitrary threshold σ on the width of the boxes, under which the Baumann inclusion is computed in addition to the natural inclusion. Intersecting both inclusions may yield a tighter enclosure of the image:

$$\underline{F(\mathbf{X})} = \begin{cases} \max\left(\underline{F_N(\mathbf{X})}, \underline{F_B(\mathbf{X})}\right) & \text{if } w(\mathbf{X}) < \sigma \\ \underline{F_N(\mathbf{X})} & \text{otherwise} \end{cases} \tag{11}$$

For small boxes, the additional information supplied by the lower bound is generally worth the cost of extra computations (see results in section 5.2).

Termination rule Parameters ϵ_x and ϵ_f determine the desired precision of the solution. $\mathbf{X}$ is stored in the solution list $\mathcal{S}$ when $w(\mathbf{X}) \leq \epsilon_x$ or $w(F(\mathbf{X})) \leq \epsilon_f$. Otherwise, $\mathbf{X}$ is bisected, and the insertion priority is computed for the resulting subboxes $\mathbf{X_1}$ and $\mathbf{X_2}$.

Bisection rule $\mathbf{X}$ is bisected along one dimension after another (round-robin method for each individual box). The two resulting subboxes are inserted in $\mathcal{L}$ to be subsequently processed.

4.3 Update thread

The individuals of the EA are periodically projected within the admissible domain to avoid exploring infeasible parts of the search-space. If an individual $\mathbf{x}$ lies outside the remaining domain, it is randomly reinitialized within the closest box $\mathbf{X}$:

$$x_i = \begin{cases} rand(\underline{X_i}, m(X_i)), & \text{if } x_i < \underline{X_i} \\ rand(m(X_i), \overline{X_i}), & \text{if } \overline{X_i} < x_i \end{cases} \tag{12}$$

5 Experimental results

In this section, we demonstrate the efficiency of our algorithm on two standard, difficult examples:

1. We compute the optima for the Michalewicz function up to 75 variables while the best known results were only computed for up to 50 variables. We then prove the optimality of these solutions, while optimality of the solutions had currently only be proven up to 12 variables. Last, we present an improvement of Adorio formula regarding the putative value of optima of Michalewicz function for all dimensions with an $R^2 = 0.9999999133$.
2. We prove that the currently putative solution to the Lennard-Jones 5 atoms cluster problem is the global optimum, a result which had up to now never been proved.

5.1 Michalewicz function

The Michalewicz function (Michalewicz 1996) is a highly multimodal function ($n!$ local optima) with domain $[0, \pi]^n$.

$$f_n(\mathbf{x}) = -\sum_{i=1}^{n} \sin(x_i) \left[\sin(\frac{ix_i^2}{\pi})\right]^{20} \tag{13}$$

The best found solutions for up to 50 dimensions are given in (Mishra 2006) using a repulsive particle swarm algorithm: $f_{10}^* = -9.6602$, $f_{20}^* = -19.6370$, $f_{30}^* = -29.6309$, $f_{50}^* = -49.6248$. Very few results regarding deterministic methods are available: Alliot et al. proved the optimality of the solution for $n = 12$ with precisions $\epsilon_x = 10^{-3}$ and $\epsilon_f = 10^{-4}$ in 6000s: $f_{12}^* = -11.64957$. Our advanced version of the cooperative algorithm significantly accelerates the proof of optimality: The convergence on the same problem is achieved after 0.03s. The proved minima[1] for $n = 10$ to 75 with parameters are presented in table 1.

n	f_n^*	n	f_n^*
10	-9.66015171564	50	-49.62483231828
20	-19.63701359935	60	-59.62314622857
30	-29.63088385032	70	-69.62222020764
40	-39.62674886468	75	-74.62181118757

Table 1: Proved minima of Michalewicz function

As an example, a comparison between DE alone, IBCA alone and our hybrid algorithm for $n = 20$ illustrates the gain in performance achieved by our approach: The DE algorithm alone converges toward a local optimum -19.6356, the IBCA alone achieves convergence in 64s and the hybrid algorithm in 0.09s.

Table 2 presents the average and maximum CPU times (in seconds) and average number of evaluations (NE) of f, F and its partial derivatives, after 100 executions of the hybrid algorithm for $n = 10$ to 75.

n	Av. time	Max. time	NE f	NE F	NE G_i
10	0.02	0.02	14,516	851	6,480
20	0.1	0.1	26,055	1,549	22,972
30	0.4	0.4	56,851	3,357	74,201
40	1.1	1.4	143,639	7,894	236,079
50	3.1	4	343,237	16,497	659,877
60	14.9	19.6	1,293,570	56,724	2,857,623
70	74.2	116.6	5,009,374	194,080	11,750,245
75	127.0	189.5	9,361,016	358,662	23,395,437

Table 2: Average values after 100 executions

It is claimed in (Adorio 2005) that $f_n^* = -0.966n$. An interpolation of f_n^* over $[10, 75]$ rather suggests that $f_n^* = -0.9995371232n + 0.3486088434$ ($R^2 = 0.9999999133$).

[1] Parameters of the algorithm were: ϵ_x (precision, width of box) $= 10^{-10}$, ϵ_f (precision, width of image) $= 10^{-10}$, σ (Baumann threshold) $= 0$, NP (population size) $= 20$ to 100, W (weighting factor) $= 0.7$, CR (crossover rate) $= 0$.

5.2 Lennard-Jones clusters

The Lennard-Jones potential is a simplified model proposed by (Jones 1924) to describe pairwise interactions between atoms. It is deemed as an accurate model of clusters of noble gas atoms. Given r_{ij} the distance between the centers of atoms i and j, the pairwise potential is defined by

$$v(r_{ij}) = 4\left(\frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}}\right) \tag{14}$$

Finding the most stable configuration of a cluster of k atoms amounts to minimizing:

$$f_n(\mathbf{x}) = \sum_{i<j}^{k} v(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}) \tag{15}$$

where (x_i, y_i, z_i) are the Cartesian coordinates of atom i. A simple way do reduce dependency in equation 14 is to complete the square so that r_{ij} occurs only once (equation 16). Sharper bounds are therefore computed in equation 15.

$$v(r_{ij}) = 4\left(\frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}}\right) = 4\left(\frac{1}{r_{ij}^{6}} - \frac{1}{2}\right)^2 - 1 \tag{16}$$

The number of effective dimensions of the problem can be reduced, as the cluster configuration is invariant under rotation and translation. In order to break part of the symmetries, we use the following boundary constraints and reduce the size of the problem to $n = 3k - 6$ components:

$$\begin{aligned} x_1 = y_1 = z_1 = 0 \\ x_2 \geq 0, y_2 = z_2 = 0 \\ x_3 \geq 0, y_3 \geq 0, z_3 = 0 \\ x_4 \geq 0, y_4 \geq 0, z_4 \geq 0 \end{aligned} \tag{17}$$

In spite of its seeming simplicity, the Lennard-Jones objective function is non-convex and highly combinatorial. Numerical experiments suggest that the number of local minima is $O(e^n)$ (Locatelli and Schoen 2003). For $n \leq 4$, the atoms may be optimally positioned on the vertices of a regular tetrahedron so that the pairwise potential v is minimal. Proof of optimality for $n \geq 5$ however remains an open problem (Vavasis 1994). Numerous authors have found putative global minima using a wide range of approximate techniques (Northby 1987; Hoare and Pal 1971; Leary 1997; Wales and Doye 1997). The best known solutions are available at *http://physchem.ox.ac.uk/˜doye/jon/structures/LJ*.

The best known solution for the **Lennard-Jones cluster of 5 atoms** (9 variables) is -9.103852415708 and the corresponding solution is a triangular bipyramid (Sloane et al. 1995). Our hybrid algorithm *proves the global minimum* $f_5^* = -9.103852415707552$ with domain $(x_i, y_i, z_i) \in [-1.2, 1.2]$. The coordinates of one optimal solution[2] are presented in table 3.

The computation times and function evaluations are given in table 4 after 100 executions. The DE algorithm finds the global optimum f_5^* after 764 iterations (0.11s), but the proof of optimality is only achieved by the IBCA after 1436s.

[2]Parameters of the algorithm were: ϵ_x (precision, width of box) $= 10^{-6}$, ϵ_f (precision, width of image) 10^{-6}, σ (Baumann threshold) $= 10^{-4}$, NP (population size) $= 40$, W (weighting factor) $= 0.7$, CR (crossover rate) $= 0.4$.

Atom	x	y	z
1	0	0	0
2	1.1240936	0	0
3	0.5620468	0.9734936	0
4	0.5620468	0.3244979	0.9129386
5	0.5620468	0.3244979	-0.9129385

Table 3: Coordinates of optimal solution (5 atoms)

Average time (s)	1436s
Maximum time (s)	1800s
Maximal queue size	46
F evaluations (IBCA)	$7,088,758$
∇F evaluations (IBCA)	$78,229,737$
f evaluations (DE)	$483,642,320$
F evaluations (DE)	132

Table 4: Convergence results after 100 executions (5 atoms)

Effect of the lower bound on the pruning step We used the Baumann centered form F_B (see section 4.2, paragraph Bounding rule) to try to compute a greater lower bound of f than that computed by the natural interval extension F_N. The lower bound of F_B was computed for $144,642$ boxes whose width was lower than the threshold σ. All of them improved the lower bound computed by F_N, and $121,333$ boxes (84%) were discarded using the cut-off test. The maximum gap computed is $\underline{F_N(\mathbf{X})} \approx -9.103968$ and $\underline{F_B(\mathbf{X})} \approx -9.103848$. We have $\underline{F_N(\mathbf{X})} < f_5^* < \underline{F_B(\mathbf{X})}$, which proves that the box cannot contain a global minimizer.

6 Conclusion

Extending the basic concept of Alliot et al., we have presented in this paper a new hybrid algorithm in which a stochastic Differential Evolution algorithm (DE) cooperates with a deterministic Interval Branch and Contract Algorithm (IBCA). The DE algorithm quickly finds incumbent solutions that help the IBCA to improve pruning the search-space. Domain reduction performed by the IBCA is used to project DE individuals trapped in local optima into the remaining domain.

We have demonstrated the efficiency of this algorithm on two very difficult and different examples: previously unknown results (for stochastic and deterministic methods) for the multimodal Michalewicz function have been computed, and the optimality for the open Lennard-Jones cluster problem with 5 atoms has been certified. This cooperative algorithm significantly outperforms both stochastic and deterministic existing optimization algorithms on these examples, and is likely to outperform them on other similar problems whose derivatives are numerically computable.

References

Adorio, E. P. 2005. Mvf - multivariate test functions library in c for unconstrained global optimization. Technical report, Department of Mathematics, U.P. Diliman.

Alliot, J.-M.; Durand, N.; Gianazza, D.; and Gotteland, J.-B. 2012. Finding and proving the optimum: Cooperative stochastic and deterministic search. *20th European Conference on Artificial Intelligence*.

Araya, I.; Trombettoni, G.; and Neveu, B. 2010. Exploiting monotonicity in interval constraint propagation. In *AAAI*.

Baumann, E. 1988. Optimal centered forms. *BIT Numerical Mathematics* 28:80–87.

Benhamou, F.; Goualard, F.; Granvilliers, L.; and Puget, J.-F. 1999. Revising hull and box consistency. In *International Conference on Logic Programming*, 230–244. MIT press.

Chabert, G., and Jaulin, L. 2009. Contractor programming. *Artificial Intelligence* 173:1079–1100.

Dorigo, M.; Maniezzo, V.; and Colorni, A. 1996. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics* 26(1):29–41.

Hansen, N., and Kern, S. 2004. Evaluating the cma evolution strategy on multimodal test functions. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, 282–291.

Hansen, E. 1992. *Global optimization using interval analysis*. Dekker.

Hoare, M., and Pal, P. 1971. Physical cluster mechanics: Statics and energy surfaces for monatomic systems. *Advances in Physics* 20(84):161–196.

Jones, J. E. 1924. On the determination of molecular fields. i. from the variation of the viscosity of a gas with temperature. *Proceedings of the Royal Society of London. Series A* 106(738):441–462.

Kennedy, J., and Eberhart, R. 1995. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*.

Leary, R. 1997. Global optima of lennard-jones clusters. *J. of Global Optimization* 11(1):35–53.

Locatelli, M., and Schoen, F. 2003. Efficient algorithms for large scale global optimization: Lennard-jones clusters. *Comput. Optim. Appl.* 26(2):173–190.

Michalewicz, Z. 1996. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag.

Mishra, S. K. 2006. Some new test functions for global optimization and performance of repulsive particle swarm method. Technical report, University Library of Munich, Germany.

Moore, R. E. 1966. *Interval Analysis*. Prentice-Hall.

Northby, J. A. 1987. Structure and binding of lennard-jones clusters: 13 <= n <= 147. *The Journal of Chemical Physics* 87(10):6166–6177.

Price, K.; Storn, R.; and Lampinen, J. 2006. *Differential Evolution - A Practical Approach to Global Optimization*. Natural Computing. Springer-Verlag.

Sloane, N.; Hardin, R.; Duff, T.; and Conway, J. 1995. Minimal-energy clusters of hard spheres. *Discrete & Computational Geometry* 14:237–259.

Sotiropoulos, G. D.; Stavropoulos, C. E.; and Vrahatis, N. M. 1997. A new hybrid genetic algorithm for global optimization. In *Proceedings of second world congress on Nonlinear analysts*, 4529–4538. Elsevier Science Publishers Ltd.

Storn, R., and Price, K. 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 341–359.

Van Hentenryck, P. 1997. Numerica: a modeling language for global optimization. In *Proceedings of the Fifteenth international joint conference on Artifical intelligence - Volume 2*, IJCAI'97, 1642–1647.

Vavasis, S. A. 1994. Open problems. *Journal of Global Optimization* 4:343–344.

Wales, D. J., and Doye, J. P. K. 1997. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A* 101(28):5111–5116.

Zhang, X., and Liu, S. 2007. A new interval-genetic algorithm. *International Conference on Natural Computation* 4:193–197.