ALGORITHMS OF MULTI-OBJECTIVE OPTIMIZATION BASED ON STATISTICAL MODELS OF OBJECTIVES

Gražina Gimbutienė

October 2016

Technical Report MII-DS-09P-16-4
Abstract

This report presents the work done during the third academic year of the PhD study programme at Vilnius University Institute of Mathematics and Informatics. The publications produced throughout the year are included.

Keywords: global optimization, statistical models, non-linear optimization
## Contents

1 Clustering-based statistical global optimization ................................................................. 4
   1.1 Introduction .............................................................................................................. 4
   1.2 The proposed algorithm ....................................................................................... 4
      1.2.1 Statistical partitioning-based global search ....................................................... 4
   1.3 The clustering procedure ....................................................................................... 5
      1.3.1 The local search ............................................................................................... 6
      1.3.2 The optimization algorithm ............................................................................. 7
   1.4 Numerical experiments ......................................................................................... 7
2 Conclusions .................................................................................................................. 8
References ...................................................................................................................... 8
1 Clustering-based statistical global optimization

This is the material presented at the conference *Numerical Computations: Theory and Algorithms (NUMTA 2016)*, held at Pizzo Calabro, Calabria, Italy on 19-25 June 2016.

1.1 Introduction

One of the known approaches to the global optimization problem $\min_{x \in A} f(x)$, where $A = \prod_{i=1}^{d}[a_i, b_i] \subset \mathbb{R}^d$ and $f(x)$ is expensive and black-box, is represented by the class of statistically-justified methods based on the theory of rational decisions under uncertainty. We refer to [ZZ07] for the introduction to this class of methods. A well-known representative of this class is the so-called P-algorithm. Assume the function evaluations $x_i, y_i = f(x_i), i = 1, \ldots, n$, have already been performed, $y_{on} = \min_{i=1,\ldots,n} y_i$ and $x_{on} = \arg \min_{i=1,\ldots,n} f(x_i)$. The P-algorithm maximizes the probability of improvement to select a new site for the function evaluation:

$$x_{n+1} = \arg \max_{x \in A} P(\xi(x) < y_{on} - \epsilon_n | x_i, y_i, i = 1, \ldots, n),$$

where $\xi(x)$ is the unknown value of $f(x)$ at $x \in A$ and $\epsilon_n > 0$ is a parameter sequence, regulating the globality of search, with larger values leading to more ambitious improvement goal and therefore the search that is more global.

Recently a version of hybridization of the idea of the P-algorithm with the rectangular decomposition of the feasible region has been introduced in [CGPv]. We build upon this algorithm to improve its convergence speed.

1.2 The proposed algorithm

1.2.1 Statistical partitioning-based global search

We propose a heuristic improvement over a global optimization algorithm first introduced in [CGPv], which is statistically-justified and is based on a rectangular decomposition of the feasible region. The original algorithm is seeking a solution to the black-box optimization problem $\min_{x \in A} f(x)$, where $A = \prod_{i=1}^{d}[a_i, b_i] \subset \mathbb{R}^d$ is a hyper-rectangle. It possesses an established asymptotic convergence rate and has implementation complexity similar to that of deterministic algorithms of similar purpose. However, the algorithm’s ability to converge after a relatively moderate number of function evaluations needs to be improved.

The algorithm [CGPv] assumes a family of Gaussian random variables $\xi(x), x \in A$, as a statistical model for $f(x)$. It performs an iterative decomposition of $A$ based on an analogue to the probability of improvement in Equation 1 at the center of any hyper-rectangular subset of $A$. It starts by evaluating the $d$-dimensional objective function on the $2^d$ vertices of a hyper-rectangular feasible region, which is scaled to the normalized hyper-rectangle $A = [0, 1]^d$, and initializes the current decomposition of the feasible region $D$ with $A$ itself. Further progress is made by selecting a single rectangle from the
current decomposition that maximizes a statistically-justified criterion and bisecting it by performing function evaluations in the middle of the edges along its longest dimension. The resulting parts replace the selected hyper-rectangle in the decomposition and the process is iterated.

The statistical criterion $\rho(R, \epsilon_n)$, i.e. a computationally efficient analogue to Equation 1, computed for each rectangle $R \in D$, is:

$$\rho(R, \epsilon_n) = \frac{V_R}{(L_R - y_{on} + \epsilon_n)^{d/2}},$$

where $V_R$ denotes the hyper-rectangle volume, $L_R$ is the mean of the function values at the vertices of the hyper-rectangle $R$, $\epsilon_n(v_n) = q \cdot d \cdot (v_n \ln(1/v_n))^{2/d}$, $q = \frac{32}{2}^{2/3} \exp(-1)$, and $v_n = \min_{R \in D} V_R$ is the smallest hyper-rectangle volume in the current decomposition. The hyper-rectangle $R^*$ satisfying $R^* = \max_{R \in D} \rho(R, \epsilon_n)$ is partitioned evaluating $f(x_k), k \in \{1, 2, \ldots, 2^{d-1}\}$, where $x_k$ is the midpoint of the $k$-th edge along the longest dimension.

In the proposed algorithm the global search is organized using the iterative rectangle selection and partitioning according to the algorithm [CGPv], and, additionally, the local refinement phase is introduced periodically.

### 1.3 The clustering procedure

During the operation of the algorithm [CGPv] the function evaluation points tend to cluster around the promising local minimizers. However, the thorough exploration of each cluster might take a significant number of algorithm iterations and slow down the convergence to the global minimum. To overcome this problem we explicitly introduce a clustering and local refinement phase into the algorithm. We employ the CURE algorithm [GRS98] as a clustering procedure, providing the unique vertices of the rectangles in the current decomposition $D$ as its input set $P$. CURE is a hierarchical clustering algorithm that can recognize arbitrarily shaped clusters, is robust in the presence of outliers, and allows for the use of only a subset of the original points to produce the clusters.

Initially the CURE algorithm treats each input point in the point set $P$ as a separate cluster. Each cluster $C$ is represented by a collection $\text{rep}(C)$ of no more than $c$ representative points, built from the well-scattered points of the cluster, that are shrunk towards its mean $\mu$ by a factor $\alpha \in [0, 1]$ such that for $q \in \text{rep}(C)$ it is true that $q = q_0 + \alpha(\mu - q_0)$, where $q_0$ is one of the cluster points. The inter-cluster distance for clusters $W$ and $Z$ is defined as $\min_{p \in \text{rep}(W), q \in \text{rep}(Z)} \text{dist}(p, q)$ for an arbitrary inter-point distance $\text{dist}(\cdot)$. The clusters are arranged in a heap according to their distance to the closest cluster and the currently closest pair of clusters is merged at each step until only $k$ clusters remain. The merged cluster contains the points of both original clusters, the representative points are recomputed and its closest cluster is determined efficiently using a range tree data structure.

The clustering procedure is repeated every time the global search has performed $n =$
$S + m T, m = 0, 1, \ldots$, function evaluations. Since we expect that the measurement clusters are circular, we used $c = 1$ and $\alpha = 1$. The number of clusters was set to $k = 50$ to ensure that the majority of clusters are of a relatively insignificant size as compared to the largest one.

Once the function evaluation clusters have been formed, each rectangle $R \in D$ is assigned to a cluster having a closest representative point to the center of $R$.

### 1.3.1 The local search

After the clusters of function evaluations have been formed, a local direct search optimization algorithm by Hooke and Jeeves [HJ61], as described in [Kel99], is run to discover the corresponding local minima. The algorithm [HJ61] is derivative-free, the function is evaluated on a stencil and the function values determine the search direction. The algorithm uses a set of decreasing scales $h_i, i = 0, \ldots, l$, to perform the search. For each scale, it starts with a base point and takes probes in a set of directions, parallel to the coordinate axes, in search of an improvement over the base point. In case of improvement the base point changes, otherwise the search scale is decreased. If the set of sampling points is bounded, the algorithm always converges after a finite number of function evaluations. Constraining the search space is achieved by prohibiting probes outside the permissible region.

In the algorithm we propose out of the $k$ clusters produced with the CURE procedure at most two are selected for local examination by algorithm [HJ61]: always the largest one, and the cluster containing $x_{on}$, if it differs from the first one. The starting point for the local descent $x_0$ is initialized to the best point of the cluster. Measures are taken to ensure that the local search stays approximately within the boundaries of the cluster $C$. To this end we compute the bounding box of the points in $C$ as $BB_p = \prod_{i=1}^d [l_i^p, u_i^p]$, $l_i^p = \min_{p \in C} p_i$, $u_i^p = \max_{p \in C} p_i$, as well as that of the rectangles assigned to $C$ as $BB_C = \prod_{i=1}^d [l_i^C, u_i^C]$, $l_i^C = \min_{C \in verts(C)} p_i$, $u_i^C = \max_{C \in verts(C)} p_i$, $verts(C) = \bigcup_{R \in \text{rects}(C)} verts(R)$, where $R$ is a rectangle assigned to $C$ and $verts(R)$ are its vertices. Also, the diameter of the points in $C$ is defined as $d_C = \max_{p,q \in C} \text{dist}(p,q)$. The search space is then constrained to the hyper-rectangle $BB_s = \prod_{i=1}^d [l_i^s, u_i^s]$, where $l_i^s = \max(0, \min(l_i^p, u_i^p) - \delta)$, $u_i^s = \min(1, \max(u_i^p, u_i^p) + \delta)$, $\delta = \frac{1}{\log_{1/2} 0.2 d_C}$. The search scales are set to $h_j = \frac{1}{2^j}, j = [\log_{1/2} 0.2 d_C], \ldots, [\log_{1/2} \Delta]$, where $\Delta$ is a parameter corresponding to the required precision. The local search stops when no more improvement over the remaining scales is achieved or the budget of function evaluations $N_{\text{max}}^{\text{local}} = 1000$ is exhausted. All function evaluations performed at this stage are stored separately from those of the global optimization algorithm, corresponding to the vertices of hyper-rectangles.

Once the local optimization is completed, some of the smallest rectangles in the current decomposition $D$ it has crossed are removed from further processing. This is achieved by first determining the bounding box of the function evaluations in the local optimization sequence. Then the rectangles in $D$ having vertices inside this hyper-rectangle are identified. Those that have volume smaller than the average volume in this
Table 1: The maximum number of function evaluations for GKLS classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>$d$</th>
<th>$\Delta$</th>
<th>$r^*$</th>
<th>$\rho^*$</th>
<th>Proposed</th>
<th>Original</th>
<th>DIRECT</th>
<th>Proposed</th>
<th>Original</th>
<th>DIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$10^{-4}$</td>
<td>0.9</td>
<td>0.2</td>
<td>237</td>
<td>237</td>
<td>111</td>
<td>616</td>
<td>1107</td>
<td>1159</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$10^{-4}$</td>
<td>0.9</td>
<td>0.1</td>
<td>788</td>
<td>920</td>
<td>1062</td>
<td>1775</td>
<td>9482</td>
<td>3201</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$10^{-6}$</td>
<td>0.66</td>
<td>0.2</td>
<td>1836</td>
<td>3454</td>
<td>386</td>
<td>4176</td>
<td>12855</td>
<td>12507</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>$10^{-6}$</td>
<td>0.9</td>
<td>0.2</td>
<td>3435</td>
<td>4964</td>
<td>1749</td>
<td>8345</td>
<td>44617</td>
<td>$&gt;10^6(4)$</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>$10^{-6}$</td>
<td>0.66</td>
<td>0.2</td>
<td>12889</td>
<td>12889</td>
<td>4805</td>
<td>51228</td>
<td>131349</td>
<td>$&gt;10^6(4)$</td>
</tr>
</tbody>
</table>

Table 2: The average number of function evaluations for GKLS classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>$d$</th>
<th>$\Delta$</th>
<th>$r^*$</th>
<th>$\rho^*$</th>
<th>Proposed</th>
<th>Original</th>
<th>DIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>$10^{-4}$</td>
<td>0.9</td>
<td>0.2</td>
<td>270.22</td>
<td>293.18</td>
<td>198.89</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>$10^{-4}$</td>
<td>0.9</td>
<td>0.1</td>
<td>842.98</td>
<td>1430.81</td>
<td>1063.78</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$10^{-6}$</td>
<td>0.66</td>
<td>0.2</td>
<td>2062.36</td>
<td>3757.67</td>
<td>1117.7</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>$10^{-6}$</td>
<td>0.9</td>
<td>0.2</td>
<td>3550.29</td>
<td>7538.56</td>
<td>&gt;42322.65</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>$10^{-6}$</td>
<td>0.66</td>
<td>0.2</td>
<td>15401.6</td>
<td>19199.11</td>
<td>&gt;47282.89</td>
</tr>
</tbody>
</table>

set are removed from $D$.

1.3.2 The optimization algorithm

This section presents the pseudo-code of the proposed algorithm.

1. function OPTIMIZE($f(x)$)
2. Evaluate $f(x)$ at the vertices of $A = [0, 1]^d$
3. while stopping condition is not satisfied do
4. Select a rectangle $R^*$ satisfying $R^* = \max_{R \in D} \rho(R, \epsilon_n)$
5. Replace $R^*$ in $D$ by two rectangles resulting from dividing $R^*$
6. if $3m \in \{0, 1, \ldots\} : n = S + mT$ then
7. Run the CURE optimization algorithm on the unique vertices of the rectangles in $D$
8. Run the Hooke-Jeeves local search algorithm on the largest cluster and the cluster that contains $x_{on}$
9. Eliminate part of the rectangles from further search
10. end if
11. end while
12. end function

1.4 Numerical experiments

The proposed and the original [CGPv] algorithms have been implemented in C++, including the CURE clustering algorithm and the Hooke-Jeeves local descent algorithm. Extensive use is made of the range tree data structure [Lue78] - a nested collection of self-balancing AVL binary trees [Knu73], which store function evaluations for repeated retrieval. The data structure allows for efficient range queries to be made.
To evaluate the performance of the proposed algorithm we used a similar methodology to that of [SK06]. It uses a free implementation [GKLS] of the GKLS test function generator [GKLS03]. For comparison we include results from [SK06] for an implementation of the global optimization algorithm DIRECT according to its description in [Gab98]. We used 5 GKLS function classes each having 100 functions that are paraboloids distorted by polynomials, forming the surface spikes. The number of local minimizers was set to \( n_{local} = 10 \) and the global minimum value was \( f^* = -1 \). The rest of the parameters, completely defining each function class, are the problem dimension \( d \), the radius of the attraction region of the global minimizer \( \rho^* \) and the distance between the paraboloid vertex and the global minimizer \( r^* \). Their values for each of the classes are given in Table 1 and 2.

The algorithms were run with a budget of \( N_{max} = 10^6 \) function evaluations until a measurement \( x_i, i = 1, \ldots, n \), close to the global minimizer \( x^* \) is generated such that \( |x_{ij} - x^*_j| \leq \sqrt[3]{\Delta} |b_j - a_j|, j = 1, \ldots, d \). The values for \( \Delta \) are also given in Table 1 and 2. The proposed algorithm was run with the following parameters for the different classes: 1) \( S = 300, T = 300 \); 2) \( S = 500, T = 300 \); 3) \( S = 1500, T = 500 \); 4) \( S = 2000, T = 1000 \); 5) \( S = 40000, T = 5000 \) and \( \Delta \) values corresponding to the class.

Tables 1 and 2 show \( \max_{i=1,\ldots,100} n_i \) and \( \frac{1}{100} \sum_{i=1}^{100} n_i \), respectively, for each of the test classes, where \( n_i \) denotes the number of function evaluations performed before the stopping condition was met. The notation \( 10^6(m) \) means that for \( m \) functions the global minimizer was not found after \( N_{max} \) function evaluations. The groups 50\% and 100\% show the same characteristics for the best 50\% and all the functions for a given algorithm and class, correspondingly. The results show that the proposed algorithm improves the average and worst-case behavior of the original algorithm and is also comparable to DIRECT.

## 2 Conclusions

The presented enhancement to a statistically-justified global optimization algorithm results in improved performance in terms of the convergence speed-up for difficult global optimization problems with many local minimizers.

## References


