The mesh adaptive direct search algorithm for granular and discrete variables

Sébastien Le Digabel
Charles Audet
Christophe Tribes

GERAD and École Polytechnique de Montréal

The Optimization Days 2018

2018–05–08
Presentation outline

Blackbox optimization

Motivating example

The MADS algorithm

Computational experiments

Discussion
Blackbox optimization

Motivating example

The MADS algorithm

Computational experiments

Discussion
Blackbox optimization (BBO) problems

- Optimization problem:
  \[
  \min_{x \in \Omega} f(x)
  \]

- Evaluations of \( f \) (the objective function) and of the functions defining \( \Omega \) are usually the result of a computer code (a blackbox).

- Variables are typically continuous, but in this work, some of them are discrete – integers or granular variables.
Blackbox optimization

Motivating example

The MADS algorithm

Computational experiments

Discussion
Example: Trust-region parameter tuning (1/2)

- [Audet and Orban, 2006].

- The classical trust-region algorithm depends on four parameters $x = (\eta_1, \eta_2, \alpha_1, \alpha_2) \in \mathbb{R}_+^4$.

- Consider a collection of 55 test problems from CUTEr.

- Let $f(x)$ be the CPU time required to solve the collection of problems by a trust-region algorithm with parameters $x$.

- $f(x_0) \simeq 3h45$ with the textbook values $x_0 = (1/4, 3/4, 1/2, 2)$. 
Example: Trust-region parameter tuning (2/2)

This optimization produced $\hat{x}$ with $f(\hat{x}) \simeq 2h50 \,-\, 30\%$.
Example: Trust-region parameter tuning (2/2)

This optimization produced \( \hat{x} \) with \( f(\hat{x}) \approx 2h50 (-30\%) \).

Victory?
Example: Trust-region parameter tuning (2/2)

This optimization produced \( \hat{x} \) with \( f(\hat{x}) \approx 2h50 \ (-30\%) \).

Victory? No, because
\[
\hat{x} = (0.22125, 0.94457031, 0.37933594, 2.3042969).
\]
Granular variables

The initial point $x_0 = (0.25, 0.75, 0.50, 2.00)$ is frequently used because each entry is a multiple of $0.25$.

Its granularity is $G = 0.25$
Granular variables

The initial point \( x_0 = (0.25, 0.75, 0.50, 2.00) \) is frequently used because each entry is a multiple of 0.25.

Its granularity is \( G = 0.25 \)

- How can we devise a direct search algorithm so that it stops on a prescribed granularity?
  With a granularity of \( G = 0.05 \), the code might produce
  \[
  \hat{x} = (0.20, 0.95, 0.40, 2.30).
  \]
  Which is much nicer (for a human) than
  \[
  \hat{x} = (0.22125, 0.94457031, 0.37933594, 2.3042969).
  \]
Granular variables

The initial point $x_0 = (0.25, 0.75, 0.50, 2.00)$ is frequently used because each entry is a multiple of 0.25.

Its granularity is $G = 0.25$

- How can we devise a direct search algorithm so that it stops on a prescribed granularity?
  
  With a granularity of $G = 0.05$, the code might produce
  
  $\hat{x} = (0.20, 0.95, 0.40, 2.30)$.

  Which is much nicer (for a human) than
  
  $\hat{x} = (0.22125, 0.94457031, 0.37933594, 2.3042969)$.

- This may be achieved using integer variables, together with a relative scaling, but there is a simpler way for mesh-based methods.
Blackbox optimization

Motivating example

The MADS algorithm

Computational experiments

Discussion
Mesh Adaptive Direct Search (MADS) in $\mathbb{R}^n$

- [Audet and Dennis, Jr., 2006].
- Iterative algorithm that evaluates the blackbox at some **trial points** on a spatial discretization called the **mesh**.
- One iteration = **search** and **poll**.
- The search allows trial points generated anywhere on the mesh.
- The poll consists in generating a list of trial points constructed from **poll directions**. These directions grow dense.
- At the end of the iteration, the mesh size is reduced if no new success point is found.
[0] **Initializations** \( (x_0, \Delta_0: \text{initial poll size}) \)

[1] **Iteration** \( k \)

- let \( \delta^k \leq \Delta^k \) be the mesh size parameter

**Search**
- test a finite number of mesh points

**Poll** (if the Search failed)
- construct set of directions \( D_k \)
- test poll set \( P_k = \{ x_k + \delta^k d : d \in D_k \} \)
- with \( \|\delta^k d\| \approx \Delta_k \)

[2] **Updates**
- if success
  - \( x_{k+1} \leftarrow \text{success point} \)
  - increase \( \Delta^k \)
- else
  - \( x_{k+1} \leftarrow x_k \)
  - decrease \( \Delta^k \)
  - \( k \leftarrow k + 1, \text{stop if } \Delta^k \leq \Delta_{\text{min}} \text{ or go to [1]} \)
Poll illustration (successive fails and mesh shrinks)

\[ \delta^k = 1 \]
\[ \Delta^k = 1 \]

trial points = \{p_1, p_2, p_3\}
Poll illustration (successive fails and mesh shrinks)

\[
\begin{align*}
\delta^k &= 1 \\
\Delta^k &= 1 \\
\delta^{k+1} &= 1/4 \\
\Delta^{k+1} &= 1/2
\end{align*}
\]

trial points = \{p_1, p_2, p_3\} = \{p_4, p_5, p_6\}
Poll illustration (successive fails and mesh shrinks)

\[
\begin{align*}
\delta^k &= 1 \\
\Delta^k &= 1
\end{align*}
\]

\[
\begin{align*}
\delta^{k+1} &= 1/4 \\
\Delta^{k+1} &= 1/2
\end{align*}
\]

\[
\begin{align*}
\delta^{k+2} &= 1/16 \\
\Delta^{k+2} &= 1/4
\end{align*}
\]

trial points = \{p_1, p_2, p_3\} = \{p_4, p_5, p_6\} = \{p_7, p_8, p_9\}
Discrete variables in MADS – so far

- MADS has been designed for continuous variables.
- Some theory exists for categorical variables [Abramson, 2004].
- So far: Only a patch allows to handle integer variables: Rounding + minimal mesh size of one.

In this work, we start from scratch and present direct search methods with a natural way of handling discrete variables.
Mesh refinement on $\min(x - 1/3)^2$

<table>
<thead>
<tr>
<th>$\Delta^k$</th>
<th>$x^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>0.125</td>
<td>0.375</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.3125</td>
</tr>
<tr>
<td>0.03125</td>
<td>0.34375</td>
</tr>
<tr>
<td>0.015625</td>
<td>0.328125</td>
</tr>
<tr>
<td>0.0078125</td>
<td>0.3359375</td>
</tr>
<tr>
<td>0.00390625</td>
<td>0.33203125</td>
</tr>
<tr>
<td>0.001953125</td>
<td>0.333984375</td>
</tr>
</tbody>
</table>
Mesh refinement on $\min(x - 1/3)^2$

<table>
<thead>
<tr>
<th>$\Delta^k$</th>
<th>$x^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>0.125</td>
<td>0.375</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.3125</td>
</tr>
<tr>
<td>0.03125</td>
<td>0.34375</td>
</tr>
<tr>
<td>0.015625</td>
<td>0.328125</td>
</tr>
<tr>
<td>0.0078125</td>
<td>0.3359375</td>
</tr>
<tr>
<td>0.00390625</td>
<td>0.33203125</td>
</tr>
<tr>
<td>0.001953125</td>
<td>0.333984375</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta^k$</th>
<th>$x^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>0.125</td>
<td>0.375</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.3125</td>
</tr>
<tr>
<td>0.03125</td>
<td>0.34375</td>
</tr>
<tr>
<td>0.015625</td>
<td>0.328125</td>
</tr>
<tr>
<td>0.0078125</td>
<td>0.3359375</td>
</tr>
<tr>
<td>0.00390625</td>
<td>0.33203125</td>
</tr>
<tr>
<td>0.001953125</td>
<td>0.333984375</td>
</tr>
</tbody>
</table>

Idea: Instead of dividing $\Delta^k$ by 2, change it so that

- $10 \times 10^b$ refines to $5 \times 10^b$
- $5 \times 10^b$ refines to $2 \times 10^b$
- $2 \times 10^b$ refines to $1 \times 10^b$
Mesh refinement on $\min(x - 1/3)^2$

<table>
<thead>
<tr>
<th>$\Delta^k$</th>
<th>$x^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>0.125</td>
<td>0.375</td>
</tr>
<tr>
<td>0.0625</td>
<td>0.3125</td>
</tr>
<tr>
<td>0.03125</td>
<td>0.34375</td>
</tr>
<tr>
<td>0.015625</td>
<td>0.328125</td>
</tr>
<tr>
<td>0.0078125</td>
<td>0.3359375</td>
</tr>
<tr>
<td>0.00390625</td>
<td>0.33203125</td>
</tr>
<tr>
<td>0.001953125</td>
<td>0.333984375</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\Delta^k$</th>
<th>$x^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>0.05</td>
<td>0.35</td>
</tr>
<tr>
<td>0.02</td>
<td>0.34</td>
</tr>
<tr>
<td>0.01</td>
<td>0.33</td>
</tr>
<tr>
<td>0.005</td>
<td>0.335</td>
</tr>
<tr>
<td>0.002</td>
<td>0.332</td>
</tr>
<tr>
<td>0.001</td>
<td>0.333</td>
</tr>
</tbody>
</table>

Idea: Instead of dividing $\Delta^k$ by 2, change it so that $10 \times 10^b$ refines to $5 \times 10^b$ $5 \times 10^b$ refines to $2 \times 10^b$ $2 \times 10^b$ refines to $1 \times 10^b$

To get three decimals, one simply sets the granularity to 0.001. Integer variables are treated by setting the granularity to $G = 1$. 
Poll and mesh size parameter update

- The poll size parameter $\Delta^k$ is updated as
  \[ 10 \times 10^b \quad \longleftrightarrow \quad 5 \times 10^b \quad \longleftrightarrow \quad 2 \times 10^b \quad \longleftrightarrow \quad 1 \times 10^b \]

- The fine underlying mesh is defined with the mesh size parameter
  \[
  \delta^k = \begin{cases} 
  1 & \text{if } \Delta^k \geq 1, \\
  \max\{10^{2b}, G\} & \text{otherwise, i.e. } \Delta^k \in \{1, 2, 5\} \times 10^b.
  \end{cases}
  \]

- Example: Granularity of $G = 0.005$:

<table>
<thead>
<tr>
<th>$\delta^k$</th>
<th>$\Delta^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.01</td>
<td>0.5</td>
</tr>
<tr>
<td>0.01</td>
<td>0.2</td>
</tr>
<tr>
<td>0.01</td>
<td>0.1</td>
</tr>
<tr>
<td>0.005</td>
<td>0.05</td>
</tr>
<tr>
<td>0.005</td>
<td>0.02</td>
</tr>
<tr>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>0.005</td>
<td>0.005 ← stop</td>
</tr>
</tbody>
</table>
[0] **Initializations** \( (x_0, \Delta_0 \in \{1, 2, 5\} \times 10^b, \mathcal{G} \text{ granularity}) \)

[1] **Iteration** \( k \)

- Let \( \delta^k \leq \Delta^k \) be the mesh size parameter

**Search**
- Test a finite number of mesh points

**Poll** (if the Search failed)
- Construct set of directions \( D_k \)
- Test poll set \( P_k = \{x_k + \delta^k d : d \in D_k\} \)
- With \( \|\delta^k d\| \approx \Delta^k \)

[2] **Updates**

- If success
  - \( x_{k+1} \leftarrow \text{success point} \)
  - Increase \( \Delta^k \)
- Else
  - \( x_{k+1} \leftarrow x_k \)
  - Decrease \( \Delta^k \)
  - \( k \leftarrow k + 1 \), stop if \( \delta^k = \mathcal{G} \) or go to [1]
Theory

- MADS analysis relies on "the sequence of trial points are located on some discretization of the space of variables called the mesh".

- By multiplying or dividing $\Delta^k$ by a rational number $\tau$, [Torczon, 1997] showed that all trial points from iteration 0 to $\ell$ were located on a fine underlying mesh. The proof is not trivial and uses the fact that $\tau \in \mathbb{Q}$, and does not work for $\tau \in \mathbb{R}$ (paper won SIAM Outstanding Paper Prize).
Theory

- MADS analysis relies on “the sequence of trial points are located on some discretization of the space of variables called the mesh”.
- By multiplying or dividing $\Delta^k$ by a rational number $\tau$, [Torczon, 1997] showed that all trial points from iteration 0 to $\ell$ were located on a fine underlying mesh.
- With the new mesh, that technical part of the proof becomes:

Consider any trial point $t$ considered from iteration $k = 0$ to $\ell$. If a granularity of $G_i$ is requested on variable $i$,

$t_i$ lies on the mesh of granularity $G_i$

if no granularity is requested on variable $i$,

$t_i$ lies on the mesh of granularity $10^{b_i}$

with $b_i = \min\{b_i^k : k = 0, 1, \ldots, \ell\}$. 

Blackbox optimization

Motivating example

The MADS algorithm

Computational experiments

Discussion
## Results on continuous analytical problems

- 87 continuous analytical computational problems from the optimization literature ($n = 2$ to $20$, 19 constrained).
- 10 LHS starting points are considered for each problem, for a total of 870 instances.
- NOMAD 3.7.3 (previous release, classic mesh: XMesh) vs NOMAD 3.8.1 (current release, new mesh: GMesh).
- Data profiles [Moré and Wild, 2009].
Data profiles on continuous problems

XMesh vs GMesh for convergence tolerance
\( \tau \in \{10^{-3}, 10^{-5}, 10^{-7}\} \):
Results on problems with discrete variables

- Set of 94 analytical mixed-integer problems.
- 10 starting points for a total of 940 instances.
- GMesh vs XMesh.
Data profiles on discrete problems

XMesh vs GMesh for convergence tolerance

\( \tau \in \{10^{-3}, 10^{-5}, 10^{-7}\} \):

![Graphs showing data profiles for different tolerance levels.](image-url)
Comparison with other solvers on discrete problems

- Set of 12 unconstrained analytical mixed-integer problems.

- 10 starting points for a total of 120 instances.

- GMesh vs XMesh vs 3 other solvers:
  - DFL [Pillo et al., 2015].
  - MISO [Mueller, 2018].
  - BFO [Porcelli and Toint, 2017].
Data profiles for NOMAD vs other solvers

XMesh vs GMesh vs DFL vs MISO vs BFO for convergence tolerance $\tau \in \{10^{-3}, 10^{-5}, 10^{-7}\}$:
Computational times

Optimization times (in seconds) for 3 selected problems from a single starting point with evaluation budget of $150 \times n$ and a limit of 2,000 evaluations for MISO:

<table>
<thead>
<tr>
<th>n</th>
<th>GMesh</th>
<th>XMesh</th>
<th>DFL</th>
<th>MISO</th>
<th>BFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>5</td>
<td>1*</td>
<td>69</td>
<td>0.1*</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>16</td>
<td>3*</td>
<td>298</td>
<td>0.3</td>
</tr>
<tr>
<td>15</td>
<td>58</td>
<td>57</td>
<td>8*</td>
<td>2,168</td>
<td>0.2</td>
</tr>
</tbody>
</table>

* converged before reaching the maximum evaluation budget.
Trust-region parameter tuning

Find the values of the four parameters \( x = (\eta_1, \eta_2, \alpha_1, \alpha_2) \in \mathbb{R}^4_+ \) that minimize the overall CPU time to solve 55 CUTEr problems.

- A surrogate function \( s \) is defined as the time to solve a collection of small-sized problems.
- In 2006, \( f(x) \simeq 4h \) and \( s(x) \simeq 1m \). The surrogate was 200 times faster.
Trust-region parameter tuning

Find the values of the four parameters \( x = (\eta_1, \eta_2, \alpha_1, \alpha_2) \in \mathbb{R}_+^4 \) that minimize the overall CPU time to solve 55 CUTEr problems.

- A surrogate function \( s \) is defined as the time to solve a collection of small-sized problems.
- In 2006, \( f(x) \simeq 4h \) and \( s(x) \simeq 1m \). The surrogate was 200 times faster.

<table>
<thead>
<tr>
<th>Year</th>
<th>CPU ( f(x) )</th>
<th>CPU ( s(x) )</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006</td>
<td>13,461s</td>
<td>69.0s</td>
<td>200</td>
</tr>
<tr>
<td>2018</td>
<td>1,008s</td>
<td>2.3s</td>
<td>440</td>
</tr>
<tr>
<td>Ratio</td>
<td>14</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Standard $\times 2 \div 2$ versus New $\{1, 2, 5\} \times 10^b$
## Trust-region parameter tuning: Results

<table>
<thead>
<tr>
<th>Algo.</th>
<th>$G$</th>
<th>Solution</th>
<th>$f_{2018}$</th>
<th>Improv. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical</td>
<td>(0.25, 0.75, 0.5, 2)</td>
<td></td>
<td>1,008.0</td>
<td>0</td>
</tr>
<tr>
<td>XMesh</td>
<td>(0.2939819787, 0.979406601, 0.4716387306, 1.474147761)</td>
<td>733.6</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>GMesh</td>
<td>(0.672010424, 0.685829734, 0.061485394, 1.34816385)</td>
<td>727.0</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>GMesh</td>
<td>0.005 (0.845, 0.99, 0.485, 1.575)</td>
<td>697.6</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>GMesh</td>
<td>0.01 (0.74, 0.99, 0.17, 1.34)</td>
<td>688.7</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>GMesh</td>
<td>0.05 (0.2, 0.9, 0.2, 1.3)</td>
<td>768.4</td>
<td>24</td>
<td></td>
</tr>
</tbody>
</table>

**Observations:**

- On this example, the new strategies seem preferable.
- Trust-region recommendation for humans:

$$ (\eta_1, \eta_2, \alpha_1, \alpha_2) = (0.74, 0.99, 0.17, 1.34). $$
Blackbox optimization

Motivating example

The MADS algorithm

Computational experiments

Discussion
Discussion (1/2)

- New mesh parameter update rules to control the number of decimals $\{1, 2, 5\} \times 10^b$:
  - A native way to handle granularity of variables $G$.
  - Integer variables are handled by setting $G = 1$. 
Discussion (1/2)

- New mesh parameter update rules to control the number of decimals $\{1, 2, 5\} \times 10^b$:
  - A native way to handle granularity of variables $G$.
  - Integer variables are handled by setting $G = 1$.

- Computational experiments on trust-region parameters:
  - New parameters reduce CPU time by $\simeq 30\%$ (versus textbook).
  - New parameters have granularity 0.01 (readable by humans).
Discussion (1/2)

- New mesh parameter update rules to control the number of decimals $\{1, 2, 5\} \times 10^b$:
  - A native way to handle granularity of variables $G$.
  - Integer variables are handled by setting $G = 1$.

- Computational experiments on trust-region parameters:
  - New parameters reduce CPU time by $\approx 30\%$ (versus textbook).
  - New parameters have granularity 0.01 (readable by humans).

- Computational experiments on analytical problems:
  $\approx 3\%$ performance improvement over the previous NOMAD version.
Discussion (2/2)

- Associated paper submitted [Audet et al., 2018].

- This is part of our NOMAD 3.8 software:
  - The only additional input from the user is $G$.
  - ... and it is optional.
  - www.gerad.ca/nomad.
References I

Mixed variable optimization of a Load-Bearing thermal insulation system using a filter pattern search algorithm.

Audet, C. and Dennis, Jr., J. (2006).

The mesh adaptive direct search algorithm for granular and discrete variables.

Finding optimal algorithmic parameters using derivative-free optimization.

Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm.

Benchmarking derivative-free optimization algorithms.
Homepage.
https://ccse.lbl.gov/people/julianem/.

DFL - Derivative-Free Library; A software library for derivative-free optimization.
Software available at http://www.dis.uniroma1.it/~lucidi/DFL/.

BFO, A Trainable Derivative-free Brute Force Optimizer for Nonlinear Bound-constrained Optimization and Equilibrium Computations with Continuous and Discrete Variables.

On the convergence of pattern search algorithms.