METRIC–BASED MESH OPTIMIZATION USING SIMULATED ANNEALING

Nazmiye Acikgoz, Carlo L. Bottasso
Georgia Institute of Technology
Davide Detomi
Politecnico di Milano

European Congress on Computational Methods in Applied Sciences and Engineering
Jyväskylä, July 24–28, 2004
Outline

- Introduction and background on metric-based mesh optimization;
- A Gauss–Seidel mesh optimization algorithm with greedy acceptance criterion;
- Limits of greedy policy;
- Simulated Annealing optimization;
- Numerical examples (SA vs. greedy);
- Local Simulated Annealing optimization;
- Numerical examples (Local SA vs. SA);
- Conclusions.
Metric Based Mesh Optimization

Target metric (either derived from an error estimator, or user-defined):

\[ \overline{M} = \overline{M}(x) \]

Matching condition between target and current metrics:

\[ \int_{\Omega_K} (M_K - \overline{M}) \, d\Omega = 0 \]

Average target metric:

\[ \overline{M}_K := \frac{\int_{\Omega_K} \overline{M} \, d\Omega}{\int_{\Omega_K} \, d\Omega} \quad \text{(here sampled at mesh vertices)} \]

Definition: simplex \( K \) of metric \( M_K \) is compliant to target metric \( \overline{M}_K \) if

\[ M_K - \overline{M}_K = 0 \]

A grid \( \mathcal{T}_h \) is compliant if all its simplexes are compliant.
**Metric Based Mesh Optimization**

**Metric–driven optimization**: construct an iterative process that produces a sequence of meshes, approximating the grid compliance condition to a given tolerance.

Some **distance function** (IJNME 2004) measures the compliance defect:

\[
d_{M_K,\overline{M}_K} := \mathcal{D}(M_K, \overline{M}_K)
\]

\[
d_{M_K,\overline{M}_K} \geq 0, \ d_{M_K,\overline{M}_K} = 0 \text{ iff } M_K = \overline{M}_K
\]

Non–dimensional definition of the compliance residual \( f_K \):

\[
f_K := \mathcal{D}(M_K, \overline{M}_K)\mathcal{D}^{-1}(\overline{M}_K, M_K)
\]

Definition of compliance residual based on multiple metric distances:

\[
f_K = \frac{\sum_{i=1}^{n_f} w_i f_{K,i}}{\sum_{i=1}^{n_f} w_i}
\]

**Remark**: the choice of the compliance residual can affect the optimization process.
Metric Based Mesh Optimization

Given a metric $\bar{M}$ and a grid $T_h$, goal of the optimization process is to **minimize the objective function**

$$\min_{T_h} J(T_h)$$

where $J(T_h) := |f|_\infty = \max_{K \in T_h} f_K$

The solution is approximated using **Gauss–Seidel simplex removal**:

- Each simplex is visited;
- If the simplex is unacceptable ($f_K > \varepsilon$):
  - Apply a “virtual” local mesh modification operator;
  - Evaluate the quality of the affected elements ($\{K_{\text{old}}\}$);
  - Evaluate the quality of the proposed elements ($\{K_{\text{new}}\}$);
  - Implement the local modification if the objective function is decreased, i.e. if

$$\Delta J \leq 0 \quad \Delta J = J(\{K_{\text{new}}\}) - J(\{K_{\text{old}}\}) + \delta$$
Metric Based Mesh Optimization

Geometric operators:

Metric–based vertex repositioning with relaxation (modified Laplacian smoothing)

\[
x = \frac{1}{n_V} \sum_{i=1}^{n_V} (x_i + (x - x_i)),
\]

\[
= \frac{1}{n_V} \sum_{i=1}^{n_V} (x_i + e_i),
\]

New position:

\[
x' := \frac{1}{n_V} \sum_{i=1}^{n_V} (x_i + \tilde{e}_i)
\]

where \(\tilde{e}_i\) are unit edges in metric space, i.e.

\[
\tilde{e}_i = \frac{e_i}{\sqrt{e_i \cdot M_K e_i}}, \quad i = (1, n_V)
\]

Use relaxation:

\[
x'' := (1 - \omega) x + \omega x' \quad \omega \in [0, 1]
\]

Project onto model boundary with closest-point interrogation:

\[
x''' := \mathcal{P}(T, x'')
\]
Metric Based Mesh Optimization

Topological operators:

- Edge split:

Project onto model boundary with closest-point interrogation:

$$x' := P(T, x)$$

- Edge collapse:
Metric Based Mesh Optimization

- **Edge swap or removal:**
  \[(2n_R - 5)!/((n_R - 1)!(n_R - 2)!)\]
  possible configurations.
  Attempt only if \( n_R \leq 7 \).

- **Face swap** (multi-face removal):
Metric Based Mesh Optimization

- **Face and region metric split:**

  **Centroid:**
  
  \[ x := \frac{1}{d} \sum_{i=1}^{d} x_i \]

  **Split position:**
  
  \[ x' := \frac{1}{d} \sum_{i=1}^{d} (x_i + \tilde{e}_i) \]

  where \( \tilde{e}_i \) are **unit edges in metric space**, i.e.

  \[ \tilde{e}_i = e_i / \sqrt{e_i \cdot M K e_i} \]

  Project onto model boundary through closest-point interrogation (if required for face split):

  \[ x'' := P(T, x') \]
Limits of Greedy Policy

"Greedy" rule (acceptance criterion)

\[ \Delta J \leq 0 \quad \Delta J = J(\{K_{\text{new}}\}) - J(\{K_{\text{old}}\}) + \delta \]

works well in most instances, but can become trapped in local minima when clusters of elements lock into a “frozen” configuration.

This is most likely:

- In the proximity of the model boundary:
- When the solution space is very steep:
Limits of Greedy Policy

We consider the following transformation of the reference right-angled tetrahedron

\[ x = FGW \hat{x} \]

where

\[
W = \begin{bmatrix}
1 & 1/2 & 1/2 \\
0 & \sqrt{3}/2 & \sqrt{3}/6 \\
0 & 0 & \sqrt{2}/3
\end{bmatrix}
\]

**Pure change of volume:**

\[ F := fI \quad f > 0 \]

**Pure change of shape at constant volume:**

\[ G := \text{diag}(\sqrt{g}, \sqrt{g}, 1/g) \quad g > 0 \quad \det(G) = 1 \]

Note: this gives an idea of the shape of the solution space.
Limits of Greedy Policy

• **Combined measure**: metric edge length + metric inscribed radius

\[ f_{K,MEL,MIR} := \frac{(f_{K,MEL} + f_{K,MIR})}{2} \]
Limits of Greedy Policy

- **Metric Frobenius norm:**

\[ f_{K,DM} = \left| (M_K - \overline{M}_K)(\overline{M}_K^{-1} - M_K^{-1}) \right|, \]

\[ = \left| M_K \overline{M}_K^{-1} + \overline{M}_K M_K^{-1} - 2I \right|. \]

Very **steep solution space** in the proximity of the target.
Limits of Greedy Policy

Initial condition: $\Delta f = 0.6$, $g = 0.5$, target: +

Edge length + inscribed radius:

Metrics Frobenius norm:

Note: local retriangulations imply a finite number of steps of given size in the solution space. **It might be impossible to move downhill along a narrow valley.**
Relax strictly downhill acceptance rule, introducing a statistically based criterion that avoids remaining trapped in local minima.

If $\Delta J > 0$ (non-decreasing move), implement mesh modification if

$$r = \text{rand}(\text{seed}) \quad r \leq e^{-\Delta J / \theta}$$

where $\theta$ is the annealing temperature.

The temperature is progressively decreased:

- At the beginning ($\theta$ still high), ability to jump out of local minima;
- Close to convergence, practically revert to greedy rule.
Numerical Results

<table>
<thead>
<tr>
<th>Max. Gauge</th>
<th>Avrg. Gauge</th>
<th>Final Mesh Size</th>
<th>Normalized CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>3.30</td>
<td>0.09</td>
<td>18680</td>
</tr>
<tr>
<td>SA</td>
<td>0.19</td>
<td>0.08</td>
<td>18805</td>
</tr>
</tbody>
</table>

Remarks:
- Average quality is similar;
- Substantial improvement in quality of worst elements;
- Increased cost.
Numerical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Gauge</th>
<th>Avrg. Gauge</th>
<th>Final Mesh Size</th>
<th>Normalized CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>3.77</td>
<td>0.16</td>
<td>1126</td>
<td>1.00</td>
</tr>
<tr>
<td>SA</td>
<td>0.16</td>
<td>0.08</td>
<td>1174</td>
<td>1.58</td>
</tr>
</tbody>
</table>

Remark: similar conclusions as in the previous case.
Local Simulated Annealing

Target only really bad elements, i.e. elements that are “lagging behind”.

Label bad elements and their neighbors (to add dofs):

\[ f_{\text{max}} = \max_{K \in \mathcal{T}_h} f_K \quad f_{\text{avg}} = \frac{1}{n} \sum_{K \in \mathcal{T}_h} f_K \]

If \( f_K > f_{\text{avg}} + (f_{\text{max}} - f_{\text{avg}}) \alpha \),

- find neighbors: \( \{ N \}_K = \text{neigh}(K, n_{\text{layers}}) \)
- mark them: \( \text{mark}(K' \in \{ N \}_K, \text{“ReallyBad”}) \)

Local SA criterion: if \( \Delta J > 0 \) and \( \text{flag}(K, \text{“ReallyBad”}) \) (non-decreasing move for a bad element), implement mesh modification if

\[ r = \text{rand}(\text{seed}) \quad r \leq e^{-\Delta J / \theta} \]
Numerical Results

<table>
<thead>
<tr>
<th></th>
<th>Max. Gauge</th>
<th>Avrg. Gauge</th>
<th>Final Mesh Size</th>
<th>Normalized CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>3.30</td>
<td>0.09</td>
<td>18680</td>
<td>1.00</td>
</tr>
<tr>
<td>SA</td>
<td>0.19</td>
<td>0.08</td>
<td>18805</td>
<td>1.43</td>
</tr>
<tr>
<td>Local SA</td>
<td>0.33</td>
<td>0.08</td>
<td>18934</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Remarks:
- Quality similar to SA;
- Cost similar to greedy.
Anisotropic Mesh Adaption

Max. Gauge
Avrg. Gauge
Final Mesh Size
Normalized CPU

<table>
<thead>
<tr>
<th>Method</th>
<th>Max. Gauge</th>
<th>Avrg. Gauge</th>
<th>Final Mesh Size</th>
<th>Normalized CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>0.36</td>
<td>0.08</td>
<td>220446</td>
<td>1.00</td>
</tr>
<tr>
<td>Local SA</td>
<td>0.36</td>
<td>0.08</td>
<td>221821</td>
<td>1.42</td>
</tr>
</tbody>
</table>

Leading edge
Trailing edge
Conclusions

Simulated Annealing optimization:
- Statistically based acceptance criterion of SA algorithm allows for escaping from local minima;
- Average quality is not affected, while worst quality is very significantly improved;
- Effective for removing bad spots (especially starting from very crude meshes, and in the proximity of the model boundary);
- Somewhat increased computational cost.

Local Simulated Annealing optimization:
- Targets bad elements that “lag behind” during the optimization process;
- Reduces the computational cost to the level of the greedy approach;
- Final mesh quality similar to SA.