Balance-Enforced Multi-Level Algorithm for Multi-Criteria Graph Partitioning
Rémi Barat, Cédric Chevalier, François Pellegrini

To cite this version:
Rémi Barat, Cédric Chevalier, François Pellegrini. Balance-Enforced Multi-Level Algorithm for Multi-Criteria Graph Partitioning. SIAM Workshop on Combinatorial Scientific Computing (CSC16), Oct 2016, Albuquerque, United States. hal-01680657

HAL Id: hal-01680657
https://inria.hal.science/hal-01680657
Submitted on 11 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Balance-Enforced Multi-Level Algorithm for Multi-Criteria Graph Partitioning

Rémi Barat\textsuperscript{1,2}, Cédric Chevalier\textsuperscript{1}, François Pellegrini\textsuperscript{2,3}
\textsuperscript{1} CEA, DAM, DIF, F-91297 Arpajon, France
\textsuperscript{2} University of Bordeaux, France
\textsuperscript{3} INRIA, France

12 October 2016
SIAM Combinatorial Scientific Computing
Outline

1. Objective
   - Context
   - Model
   - State of the art

2. Approach
   - The multi-level framework
   - Contributions
   - Example

3. Experiments
   - Mono-criterion partitioning (mesh of 3500 cells)
   - Multi-criteria partitioning (mesh of 3500 cells)
   - Multi-criteria partitioning (mesh of 22800 cells)
Outline

1. Objective
   - Context
   - Model
   - State of the art

2. Approach
   - The multi-level framework
   - Contributions
   - Example

3. Experiments
   - Mono-criterion partitioning (mesh of 3500 cells)
   - Multi-criteria partitioning (mesh of 3500 cells)
   - Multi-criteria partitioning (mesh of 22800 cells)
High Performance Computing on distributed memory architectures.

To get an efficient code, one must:

1. balance the workloads of each processor
2. overlap or minimize communications
3. take care of memory accesses
4. exploit full processor characteristics

We focus on the 1st and 2nd items.

Direct application: multi-physics numerical simulations using 2D or 3D meshes.
Hypergraph model

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Dual Hypergraph $H = (V, E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Mesh Diagram]</td>
<td>![Dual Hypergraph Diagram]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cell $c_i$</th>
<th>vertex $v_i \in V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight vector of a cell</td>
<td>weight vector of a vertex</td>
</tr>
<tr>
<td>$c_i$ and its neighboring cells $N_i$</td>
<td>hyperedge $e = N_i \cup c_i \in E$</td>
</tr>
<tr>
<td>communicate $c_i$ means $y$ communications</td>
<td>weight $y$ on the hyperedge corresponding to cell $c_i$</td>
</tr>
</tbody>
</table>

**Problem : Hypergraph partitioning**

Let $p$ be the number of processors. We search for an indexed family $(V_k)_{0 \leq k < p}$ of subsets of $V$ pairwise disjoint and of union $V$, respecting:

1. some **constraints**: well-balanced workloads
2. an **objective**: minimize the communications.

NP-Hard Problem, no algorithm can always return the optimal solution.
Main existing software:

<table>
<thead>
<tr>
<th>Software</th>
<th>Representations</th>
<th>Multi-Criteria</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scotch</td>
<td>Topological</td>
<td>No</td>
<td>INRIA, F. Pellegrini et. al.</td>
</tr>
<tr>
<td>MeTiS</td>
<td>Topological</td>
<td>Yes</td>
<td>University of Minnesota, G. Karypis et. al.</td>
</tr>
<tr>
<td>Zoltan</td>
<td>Geometric</td>
<td>Yes</td>
<td>Sandia National Laboratories, K. Devine et. al.</td>
</tr>
<tr>
<td></td>
<td>Topological</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Current limitations for the codes in CEA, DAM, DIF:

- Scotch does not fit: real need of a multi-criteria partitioner
- MeTiS does not meet the balance constraints
- Zoltan geometric representations are inefficient for our meshes

⇒ Lack of efficient multi-criteria partitioning tools.
Outline

1. Objective
   - Context
   - Model
   - State of the art

2. Approach
   - The multi-level framework
   - Contributions
   - Example

3. Experiments
   - Mono-criterion partitioning (mesh of 3500 cells)
   - Multi-criteria partitioning (mesh of 3500 cells)
   - Multi-criteria partitioning (mesh of 22800 cells)
Classic algorithm: The multi-level framework

A 3-phases algorithm:

1. **Coarsening**
2. **Initial partitioning of the coarsened hypergraph**
3. **Uncoarsening and refinement**
A 3-phases algorithm:

1. Coarsening
2. Initial partitioning of the coarsened hypergraph → New algorithm focusing on balance constraints
3. Uncoarsening and refinement → Adapted Fiduccia-Mattheyses algorithm
Initial partitioning algorithm

Problem: partition a set of vectors of numbers

- The vertices’ weights alone are considered, not the hyperedges.
- Some algorithms exist in mono-criterion (number partitioning), but in our knowledge not in multi-criteria.

Algorithm 1 Initial partitioning algorithm

Require: $V$ set of vertices, $\Pi$ partition

1: $b_{max} \leftarrow \max_{\text{criterion } c} \text{Imbal}_c(\Pi)$
2: repeat
3: for $v \in V$ do
4: if changing partition of $v$ decreases $b_{max}$ then
5: $\Pi \leftarrow$ change partition of $v$
6: update $b_{max}$
7: end if
8: end for
9: until No more vertex move can decrease $b_{max}$
Example: initial partitioning

Simple instance:

- 8 vertices
- 2 criteria
- 2 partitions

Given a partition, choose a vertex to move:

Movement of a vertex from partition 2 to partition 1: balance gain of $30\% - 25\% = +5\%$

$b_{\text{max}}: 30\%$

$b_{\text{max}}: 25\%$
Refinement algorithm: Fiduccia-Mattheyses

Key points:

- Move vertices according to their gain ("moves").
- Avoid opposite moves: lock on the moved vertices.
- When no more moves are possible: restore the best partition found.
- If improvement: start a new "pass". Otherwise, end of the algorithm.

Algorithm 2 Fiduccia-Mattheyses algorithm

Require: Partition respecting the constraints

repeat  # Make a pass
2: Unlock all vertices, compute their gains
   while possible moves remain do
4: Move vertex of best gain and lock it
   Update neighbor gains and save current partition
6: end while
   Restore the best partition reached in the pass
8: until No improvement on the best partition quality
Lots of possible variations:

<table>
<thead>
<tr>
<th>Options</th>
<th>Our choice</th>
<th>Scotch</th>
<th>MeTiS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prescribed tolerance</td>
<td>strict</td>
<td>relaxed at lower levels</td>
<td>relaxed ( \propto \frac{1}{2 \times \text{graph size}} )</td>
</tr>
<tr>
<td>Select move</td>
<td>best gain</td>
<td>best gain</td>
<td>best gain</td>
</tr>
<tr>
<td>Tie breaking</td>
<td>first</td>
<td>lowest imbalance</td>
<td>first</td>
</tr>
<tr>
<td>Inner loop stop condition</td>
<td></td>
<td>120</td>
<td>between 25 and 150 (1% \times \text{graph size})</td>
</tr>
</tbody>
</table>

- Other remarks: hypergraph model, 2 independent runs by default, rebalancing phases
Summary of the algorithm

Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

⇒ Each solution found is **guaranteed** to respect all balance constraints
Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

Each solution found is guaranteed to respect all balance constraints

Mesh of 600 triangles
Vertex weights: 3 criteria
Edge weights depend on vertex weights
Summary of the algorithm
Example: initial partitioning

Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

⇒ Each solution found is guaranteed to respect all balance constraints

| Initial partition of the coarsest hypergraph |
| imbalances | 0.3% | 2.7% | 2.7% |
| communications | 583 |
Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

⇒ Each solution found is guaranteed to respect all balance constraints

<table>
<thead>
<tr>
<th>Initial partition (level -3)</th>
<th>Refinement (level -3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>imbalances</td>
<td>imbalances</td>
</tr>
<tr>
<td>0.3% 2.7% 2.7%</td>
<td>0.2% 4.7% 4.8%</td>
</tr>
<tr>
<td>communications</td>
<td>communications</td>
</tr>
<tr>
<td>583</td>
<td>197</td>
</tr>
</tbody>
</table>
Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

⇒ Each solution found is guaranteed to respect all balance constraints

<table>
<thead>
<tr>
<th>Level</th>
<th>Imbalances</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>IP (-3)</td>
<td>R (-3)</td>
</tr>
<tr>
<td>coms:</td>
<td>583</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>Refinement (level -2)</td>
<td>2.3%</td>
</tr>
<tr>
<td></td>
<td>IP (-3)</td>
<td>R (-3)</td>
</tr>
<tr>
<td>coms:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>Imbalances</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>IP (-3)</td>
<td>R (-3)</td>
</tr>
<tr>
<td>coms:</td>
<td>0.8%</td>
<td>0.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Final partition (level 0)</td>
<td>0.8%</td>
</tr>
<tr>
<td></td>
<td>IP (-3)</td>
<td>R (-3)</td>
</tr>
<tr>
<td>coms:</td>
<td>28</td>
<td>97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>Imbalances</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>Refinement (level -2)</td>
<td>2.3%</td>
</tr>
<tr>
<td></td>
<td>IP (-3)</td>
<td>R (-3)</td>
</tr>
<tr>
<td>coms:</td>
<td>0.8%</td>
<td>0.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Final partition (level 0)</td>
<td>0.8%</td>
</tr>
<tr>
<td></td>
<td>IP (-3)</td>
<td>R (-3)</td>
</tr>
<tr>
<td>coms:</td>
<td>28</td>
<td>97</td>
</tr>
</tbody>
</table>
Summary of the algorithm

Example: uncoarsening and refinement

Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

⇒ Each solution found is **guaranteed** to respect all balance constraints
Algorithmic contribution: multi-level for multi-criteria partitioning

1. Classic coarsening (Heavy-Edge Matching)
2. Greedy initial partitioning returning a solution respecting the balance constraints
3. Refinement of the objective function respecting the balance constraints

⇒ Each solution found is guaranteed to respect all balance constraints

---

<table>
<thead>
<tr>
<th>Level</th>
<th>IP (-3) coms:</th>
<th>R (-3) coms:</th>
<th>R (-2) coms:</th>
<th>R (-1) coms:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>583</td>
<td>97</td>
<td>52</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Final partition (level 0)</th>
<th>Imbalances</th>
<th>Communications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.8%</td>
<td>0.6%</td>
</tr>
</tbody>
</table>
Outline

1 Objective
   ▪ Context
   ▪ Model
   ▪ State of the art

2 Approach
   ▪ The multi-level framework
   ▪ Contributions
   ▪ Example

3 Experiments
   ▪ Mono-criterion partitioning (mesh of 3500 cells)
   ▪ Multi-criteria partitioning (mesh of 3500 cells)
   ▪ Multi-criteria partitioning (mesh of 22800 cells)
Experiment 1
Comparison with MeTiS and Scotch (mono-criterion)

### Instance

<table>
<thead>
<tr>
<th># cells</th>
<th>3500</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertex weights statistics:</td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>10</td>
</tr>
<tr>
<td>max</td>
<td>2457</td>
</tr>
<tr>
<td>average</td>
<td>318</td>
</tr>
<tr>
<td>std</td>
<td>507</td>
</tr>
<tr>
<td>edge weights:</td>
<td></td>
</tr>
<tr>
<td>hypergraph model</td>
<td>weight of cell</td>
</tr>
<tr>
<td>graph model</td>
<td>sum of weights of ends</td>
</tr>
</tbody>
</table>

### Parameters

| runs | 500 (random numbering of the graph vertices for each run) |
| tolerance | 5% |
| MeTiS version | 5.1.0¹ |
| Scotch version | 6.0.4² |

¹ MeTiS is used with vertex sizes provided, so that it minimizes exactly communication volume (unlike Scotch which minimizes the edge-cut).

² By default, Scotch launches 2 independent runs and returns the best partition found.

Bi-partition example
The darker a cell, the heavier its weight
Blue line: border
## Experiment 1

*Comparison with MeTiS and Scotch (mono-criterion)*

<table>
<thead>
<tr>
<th>Software</th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Scotch</th>
</tr>
</thead>
<tbody>
<tr>
<td>constraints valid solutions</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>communications average</td>
<td>3756</td>
<td>5392</td>
<td>3519</td>
</tr>
<tr>
<td>std</td>
<td>1047</td>
<td>751</td>
<td>535</td>
</tr>
<tr>
<td>min</td>
<td>2431</td>
<td>2908</td>
<td>2443</td>
</tr>
<tr>
<td>median</td>
<td>3434</td>
<td>5482</td>
<td>3514</td>
</tr>
<tr>
<td>max</td>
<td>8551</td>
<td>6959</td>
<td>5301</td>
</tr>
</tbody>
</table>

**Observations:**
- Scotch is the best
- Our algorithm statistics seem close
# Experiment 1

Comparison with MeTiS and Scotch (mono-criterion)

<table>
<thead>
<tr>
<th>Software constraints</th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Scotch</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid solutions</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Communications**

<table>
<thead>
<tr>
<th></th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Scotch</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>3756</td>
<td>5392</td>
<td>3519</td>
</tr>
<tr>
<td>std</td>
<td>1047</td>
<td>751</td>
<td>535</td>
</tr>
<tr>
<td>min</td>
<td>2431</td>
<td>2908</td>
<td>2443</td>
</tr>
<tr>
<td>median</td>
<td>3434</td>
<td>5482</td>
<td>3514</td>
</tr>
<tr>
<td>max</td>
<td>8551</td>
<td>6959</td>
<td>5301</td>
</tr>
</tbody>
</table>

**Observations:**

- Very different behaviors
- High discrepancy
Experiment 2  
Comparison with MeTiS (multi-criteria)

### Instance

<table>
<thead>
<tr>
<th># cells</th>
<th>3500</th>
</tr>
</thead>
</table>

**Vertex weights statistics (3 criteria):**

<table>
<thead>
<tr>
<th>min</th>
<th>10</th>
<th>10</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>2487</td>
<td>2403</td>
<td>2464</td>
</tr>
<tr>
<td>average</td>
<td>296</td>
<td>288</td>
<td>257</td>
</tr>
<tr>
<td>std</td>
<td>473</td>
<td>448</td>
<td>444</td>
</tr>
</tbody>
</table>

**Edge weights:**

- Hypergraph model: 1st weight of cell
- Graph model: sum of 1st weights of ends

### Parameters

<table>
<thead>
<tr>
<th>runs</th>
<th>500 (random numbering of the graph vertices for each run)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tolerance</td>
<td>5%</td>
</tr>
<tr>
<td>MeTiS version</td>
<td>5.1.0¹</td>
</tr>
</tbody>
</table>

¹ MeTiS is used with vertex sizes provided.

Bi-partition example  
One color = one criterion  
Blue line: border
### Experiment 2

**Comparison with MeTiS (multi-criteria)**

<table>
<thead>
<tr>
<th>Software</th>
<th>Our algorithm</th>
<th>MeTiS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>constraints statistics:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valid solutions</td>
<td>100%</td>
<td>60%</td>
</tr>
<tr>
<td><strong>communication statistics:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>2733</td>
<td>2436</td>
</tr>
<tr>
<td>std</td>
<td>2316</td>
<td>1729</td>
</tr>
<tr>
<td>min</td>
<td>215</td>
<td>340</td>
</tr>
<tr>
<td>median</td>
<td>1888</td>
<td>1839</td>
</tr>
<tr>
<td>max</td>
<td>9673</td>
<td>6093</td>
</tr>
</tbody>
</table>

**Observations:**
- MeTiS seems to achieve better performance in terms of partition quality.
- However, its policy to relax constraints leads to invalid solutions.
### Experiment 2

**Comparison with MeTiS (multi-criteria)**

<table>
<thead>
<tr>
<th>Software</th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Failsafe-MeTiS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>constraints statistics:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valid solutions</td>
<td>100%</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>communication statistics:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>2733</td>
<td>2436</td>
<td></td>
</tr>
<tr>
<td>std</td>
<td>2316</td>
<td>1729</td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>215</td>
<td>340</td>
<td></td>
</tr>
<tr>
<td>median</td>
<td>1888</td>
<td>1839</td>
<td></td>
</tr>
<tr>
<td>max</td>
<td>9673</td>
<td>6093</td>
<td></td>
</tr>
</tbody>
</table>

### Observations:

- **Failsafe-MeTiS**: if solution found is invalid, relaunched with half-tolerance.

- Better performance when constraints are tougher!
Experiment 2
Comparison with MeTiS (multi-criteria)

<table>
<thead>
<tr>
<th>Software</th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Failsafe-MeTiS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>constraints statistics:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valid solutions</td>
<td>100%</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>communication statistics:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>2733</td>
<td>2436</td>
<td>2291</td>
</tr>
<tr>
<td>std</td>
<td>2316</td>
<td>1729</td>
<td>1517</td>
</tr>
<tr>
<td>min</td>
<td>215</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>median</td>
<td>1888</td>
<td>1839</td>
<td>1787</td>
</tr>
<tr>
<td>max</td>
<td>9673</td>
<td>6093</td>
<td>6093</td>
</tr>
</tbody>
</table>

Observations:
- Failsafe-MeTiS: if solution found is invalid, relaunched with half-tolerance.
- Better performance when constraints are tougher!
# Experiment 2

## Comparison with MeTiS (multi-criteria)

### Software
<table>
<thead>
<tr>
<th></th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Failsafe-MeTiS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>constraints statistics:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>valid solutions</td>
<td>100%</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>communication statistics:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>2733</td>
<td>2436</td>
<td>2291</td>
</tr>
<tr>
<td>std</td>
<td>2316</td>
<td>1729</td>
<td>1517</td>
</tr>
<tr>
<td>min</td>
<td>215</td>
<td>340</td>
<td>340</td>
</tr>
<tr>
<td>median</td>
<td>1888</td>
<td>1839</td>
<td>1787</td>
</tr>
<tr>
<td>max</td>
<td>9673</td>
<td>6093</td>
<td>6093</td>
</tr>
</tbody>
</table>

### Observations:
- The comparison is less straightforward
- Our algorithm gets lots of solutions of very good quality...
- ...but also some of very bad quality
- Relaxing the constraints does not lead to better solutions more often here
- The discrepancy is greater for this instance.
Experiment 3
Comparison with MeTiS (multi-criteria)

**Instance**

| # cells | 22800 |
|----------------|
| vertex weights statistics (3 criteria): |
| min | 10 | 10 | 1 |
| max | 2403 | 9671 | 1 |
| average | 148 | 322 | 1 |
| std | 418 | 1074 | 0 |

**Parameters**

| runs | 60 (random numbering of the graph vertices for each run) |
| tolerance | 5% |
| MeTiS version | 5.1.0¹ |

¹ MeTiS is used with vertex sizes provided.
Experiment 3

Comparison with MeTiS (multi-criteria)

<table>
<thead>
<tr>
<th>Software</th>
<th>Our algorithm</th>
<th>MeTiS</th>
<th>Failsafe-MeTiS</th>
</tr>
</thead>
<tbody>
<tr>
<td>runs</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

**Constraints statistics:**

| Valid solutions | 100% | 47% | 100% |

**Communication statistics:** \((\times 1000)\)

<table>
<thead>
<tr>
<th>Average</th>
<th>43.4</th>
<th>57.1</th>
<th>56.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std</td>
<td>13.5</td>
<td>9.5</td>
<td>8.8</td>
</tr>
<tr>
<td>Min</td>
<td>28.0</td>
<td>41.5</td>
<td>41.5</td>
</tr>
<tr>
<td>Median</td>
<td>38.9</td>
<td>57.1</td>
<td>56.2</td>
</tr>
<tr>
<td>Max</td>
<td>75.7</td>
<td>71.6</td>
<td>71.6</td>
</tr>
</tbody>
</table>

Observations:

- MeTiS returns lots of invalid solutions, but does not perform better than Failsafe-MeTiS.
- Our algorithm reaches better partitions for this instance.
- Still a very high discrepancy, no matter the tool.
Objective: accelerate multi-physics simulations by balancing the workload and minimizing the communications.

Approach and contributions:
- Adaptation of the multi-level framework to multi-criteria graphs or hypergraphs
- New initial partitioning algorithm
- Refinement respecting the balance constraints

Implementation of a Python prototype

Comparison with some existing tools:
- Studies more precisely the algorithms behavior
- Shows their lack of robustness
- Questions MeTiS policy to relax constraints
Currently: implementation (open-source) of the multi-criteria algorithms in Scotch

- Validation on real size instances
- Validation on a simulation code
- New release next year

Enforce the algorithm robustness by:

- Analyzing the algorithms behavior
- Studying the influence of each parameter
- Working on the graph numbering

Set up of a parallel version of the algorithms
Thank you