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 CSE 2017, Mar 2017, Atlanta, United States. hal-01671514

HAL Id: hal-01671514
https://inria.hal.science/hal-01671514
Submitted on 22 Dec 2017

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 1,2
Cédric Chevalier 1
François Pellegrini 2,3
Speaker: Jean-Sylvain Camier 1
1 CEA, DAM, DIF, F-91297 Arpajon, France
2 University of Bordeaux, France
3 INRIA, France

28 February 2017
SIAM Conference on Computational Science and Engineering
Outline

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

2. Approach
   - The multi-level framework
   - Scotch multi-criteria algorithm

3. Results

4. Perspective
   - Improving robustness

5. Conclusion
Outline

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

2. Approach
   - The multi-level framework
   - Scotch multi-criteria algorithm

3. Results

4. Perspective
   - Improving robustness

5. Conclusion
In the field of numerical simulation, one challenge is to reduce the restitution time.

We consider simulations:
- Running on distributed memory architectures
- Using 2D or 3D meshes
- Coupling several physics (multi-physics simulations)

Running these simulations efficiently needs to:
1. Balance the workloads of each processor for each phase of the computation
2. Overlap or minimize communications
3. Take care of memory accesses
4. Exploit full processor characteristics

We focus on the first 2 items.
Algorithm 1 Example of the temporal loop of a multi-physics simulation
(2 computing phases: $f$ and $g$, coupling the variables $X$ and $Y$)

\[
\text{for } t \in [0, t_{\text{end}} - 1] \text{ do}
\]
\[
X(t + 1) \leftarrow f(t + 1, X(t), Y(t))
\]
\[
Y(t + 1) \leftarrow g(t + 1, X(t + 1), Y(t))
\]
\[
\text{end for}
\]

Objectives:

1. Balance the workloads of each processor for each phase
2. Minimize communications between processors

<table>
<thead>
<tr>
<th>Input</th>
<th>Model</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry</td>
<td>Weights</td>
<td>Multi-criteria Dual Graph (1 color = 1 criterion)</td>
</tr>
<tr>
<td>Weights for phase 1:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weights for phase 2:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>imbalance c1: 1.9%</td>
<td>imbalance c2: 2.9%</td>
<td>edgecut: 11</td>
</tr>
</tbody>
</table>
Model

Mesh

Dual Graph $G = (V, E)$

Problem: Graph partitioning

We search for a partition $P$ of $V$ respecting:

1. some **constraints**: load-balancing
   (imbalance of $P$ is for all criteria smaller than a given tolerance)

2. an **objective**: minimize the edgecut
   (sum of the weights of the edges cut by the partition).

NP-Hard Problem.
Main existing software:

<table>
<thead>
<tr>
<th>Software</th>
<th>Representations</th>
<th>Multi-criteria</th>
<th>Origin</th>
<th>Limitations for us</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scotch</td>
<td>Topological</td>
<td>No</td>
<td>INRIA</td>
<td>No multi-criteria (yet)</td>
</tr>
<tr>
<td>MeTiS</td>
<td>Topological</td>
<td>Yes</td>
<td>University of Minnesota</td>
<td>Does not often meet the balance constraints</td>
</tr>
<tr>
<td>PaToH</td>
<td>Topological</td>
<td>Yes</td>
<td>Bilkent University</td>
<td>No parallel version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No open-source version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Does not always meet the balance constraints</td>
</tr>
<tr>
<td>Zoltan</td>
<td>Geometric</td>
<td>Yes</td>
<td>Sandia National Laboratories</td>
<td>Geometric representations are inefficient</td>
</tr>
<tr>
<td></td>
<td>Topological</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⇒ Lack of efficient multi-criteria partitioning tools.
1. Objective
   - Context
   - Example & Model
   - State of the art

2. Approach
   - The multi-level framework
   - Scotch multi-criteria algorithm

3. Results

4. Perspective
   - Improving robustness

5. Conclusion
A 3-phases algorithm:

1. Coarsening

2. Initial partitioning of the coarsened hypergraph

3. Uncoarsening and refinement
Our approach: Multi-level multi-criteria algorithm

A 3-phases algorithm:

1. **Coarsening**
   - (Ongoing) Selection of a matching policy improving the quality of the returned solutions

2. **Initial partitioning of the coarsened hypergraph**
   - New algorithm focusing on balance constraints

3. **Uncoarsening and refinement**
   - Adapted Fiduccia-Mattheyses algorithm

Implemented in Scotch
Coarsening:
- Classical algorithm: Heavy-Edge-Matching

Initial Partitioning:
- Local optimization of the balance
- Enforce balance for all the constraints
- Presented in SIAM CSC 2016

Refinement:
- Fiduccia-Mattheyses algorithm: local optimization of the edgecut with hill-climbing
- Moves that unbalance the partition more than the tolerance are forbidden

⇒ Finding a balanced partition of the coarsest graph guarantees that the returned partition will respect the balance constraints.
Outline

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

2. Approach
   - The multi-level framework
   - Scotch multi-criteria algorithm

3. Results

4. Perspective
   - Improving robustness

5. Conclusion
Instances and Parameters

Comparison between
- MeTiS (5.1.0)
- PaToH (3.2)
- Scotch (6.1.0 - not yet released multi-criteria version)

Instances considered:

<table>
<thead>
<tr>
<th>Name</th>
<th>Geometry</th>
<th>Vertices</th>
<th>Edges</th>
<th>Criteria</th>
<th>Edge weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>mushroom_1</td>
<td>2D</td>
<td>22800</td>
<td>45253</td>
<td>3</td>
<td>$\sqrt{w_i + w_j}$</td>
</tr>
<tr>
<td>mushroom_2</td>
<td>2D</td>
<td>22800</td>
<td>45253</td>
<td>3</td>
<td>independant</td>
</tr>
<tr>
<td>onera</td>
<td>3D</td>
<td>85567</td>
<td>166817</td>
<td>3</td>
<td>$\sqrt{w_i + w_j}$</td>
</tr>
<tr>
<td>wave</td>
<td>3D</td>
<td>156317</td>
<td>1059331</td>
<td>3</td>
<td>$\sqrt{w_i + w_j}$</td>
</tr>
</tbody>
</table>

Partitioning parameters:
- Imbalance tolerance 5%
- Number of parts: 2
- Each algorithm is executed 100 times on each instance with random seeds.
Results

1 - How to compare the results

\[ f_{\text{algo}}(x) = \text{number of valid solutions of algo of cut} \leq x. \]

- We want to minimize the cut of each returned solution: left is better
Results

2 - Comparison on the quality of the returned solutions

- *mushroom_1*

- *mushroom_2*

- *onera*

- *wave*
Outline

1 Objective
   - Context
   - Example & Model
   - State of the art

2 Approach
   - The multi-level framework
   - Scotch multi-criteria algorithm

3 Results

4 Perspective
   - Improving robustness

5 Conclusion
• Python prototype: Crack
• Allows to test different parameters on the multi-level
• An example: settings of the coarsening parameters
  - Heavy-Edge-Matching
  - Particular order of the vertices:
    - sorted by increasing degrees
  - Do not allow to match 2 vertices if the resulting vertex is overweighted
Comparison of Crack on the previous instances

- mushroom_1
- mushroom_2
- onera
- wave
Outline

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

2. Approach
   - The multi-level framework
   - Scotch multi-criteria algorithm

3. Results

4. Perspective
   - Improving robustness

5. Conclusion
Summary:

- Scotch next release will support multi-criteria graph partitioning
- This new version implements a new initial partitioning and sets the refinement phase to guarantee that the returned solution will respect the balance constraints
- Scotch results are promising

Perspectives:

- Enforce the algorithm robustness by the study of variations of the algorithms
- Validation on more instances
  - Validation on a simulation code
- Set up of a parallel version of the algorithms
Thank you
Imbalance distributions

mushroom_1

mushroom_2

onera

wave