Balance-Enforced Multi-Level Algorithm for Multi-Criteria Mesh Partitioning

Rémi Barat*
Cédric Chevalier*
François Pellegrini†

2 September 2016

1 Context

Nowadays, numerical simulations model increasingly complex phenomena. They require deeply coupled multi-physics codes that are designed to run on large distributed memory computers. On these kinds of architecture, data decomposition is critical to achieve good performance.

While distributing the data between the processes, two challenges must be addressed: balancing the workload between all processes, while minimizing the communications between processes.

In multi-physics simulations, one iteration gathers the computations for several phases (see algorithm 1). The computation costs depend on the physical phase, so a partition well-balanced for one phase can be imbalanced for another. Computing one partition per phase must update the data distribution after each phase, leading to a great amount of communications, as well as one partition computation per phase. Finding a partition well-balanced for all phases garantees a correct parallel efficiency.

Algorithm 1 Typical temporal loop of a numerical simulation with two computing phases, f and g, coupling the variables X and Y from two different models.

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

2 Model

Load balanced partitioning of a mesh with minimal communications can be reduced as a graph or hypergraph partitioning problem. As the computing task differs for each cell and each phase, a weight vector is associated to the corresponding vertex. An edge cut by a partition represents a communication between processes. The edge can be valued to modelize communication costs. Classicaly, the objective is to find a partition of the vertices that balances the weights of each partition, while minimizing the edge-cut.

Hypergraph partitioning model for mesh load balancing being more accurate than graph, we will focus on hypergraph partitioning. However, all the algorithms presented can be used also for graph partitioning.

A partition is not valid when its imbalance for a criterion is greater than a given threshold, the tolerance. Whereas balancing the weight of each part for each criterion is a constraint, minimizing the cut of the partition is the objective. Thus, we compare well-balanced partitions using their cut.

Most of the current approaches do not strictly enforce constraints, returning partitions that are not always valid in respect to the constraints. We present an algorithm which strictly enforces balance constraints for all criteria.

3 Our algorithm

Our approach uses a multi-level framework where the key algorithms have been designed to always enforce balance constraints.

The coarsening phase is classical. We use a Heavy-Edge-Matching scheme adapted for hypergraph. However, unlike a lot of partitioning tools, we do not take into account the vertices weights during this phase.

The initial partitioning phase is a simple but new initial partitioning method. It is based on a local optimization of the equilibrium, and this phase does not take into account the hyperedges; it is actually a number partitioning algorithm (but with vectors of numbers). The objective is only to find a balanced partition.

The refinement algorithm of the uncoarsening phase is the Fiduccia-Mattheyses algorithm. Only moves that do not unbalance the partition above the tolerance are allowed. Some tools such as Metis allow such moves, which make them return unbalance solutions. On the

*CEA, DAM, DIF, F–91297 Arpajon, France, {remi.barat.ocre|cedric.chevalier}@cea.fr
†Labri & Inria Bordeaux Sud-Ouest, Université de Bordeaux, 33400 Talence, France, francois.pellegrini@labri.fr

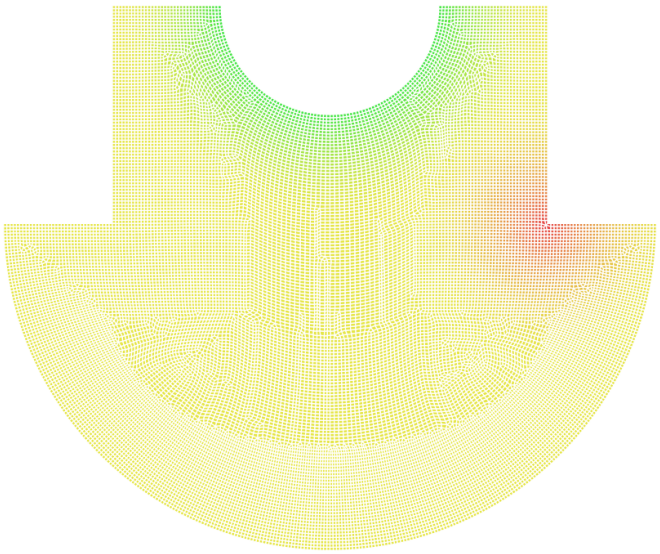

Figure 1: $2D$ mesh of 22800 cells used for our multi-criteria experiments. Three criteria have been defined and are represented on the mesh using three colors. The first two, green and red, are localized. The last one, blue, is not visible since gives a unitary weight for every cell.

contrary, if our initial partition is balanced, then the final returned solution is garanteed to be balanced.

4 Results

We will present results on a mono-criterion and on two multi-criteria meshes. We implemented our algorithm in Python: although the complexity are the same, we can only test it on small instances, of 3584 and 22800 cells respectively. The latter is presented in figure 1.

We will compare to Scotch (for the mono-criterion instance) and Metis. Since the latter returns a non-negligeable part of invalid solutions (that is, imbalanced more than the given tolerance of 5% in our case), we also used what we called *fail-safe*-Metis. In this version, we call Metis normally, but whenever it returns an invalid partition, Metis is re-run with half the tolerance. In practice, *fail-safe*-Metis returns only valid solutions.

Each algorithm is launched a number of times on each of the three instances. Figure 2 presents the communication volume distributions of valid instances returned by 64 runs of each algorithm, for the three criteria instance of 22800 cells.

From this small example, we can already see that there is a great discrepancy between the returned solutions, for all algorithms. We do not know of papers reporting this important phenomenon yet. We believe that when comparing algorithms, one must also study

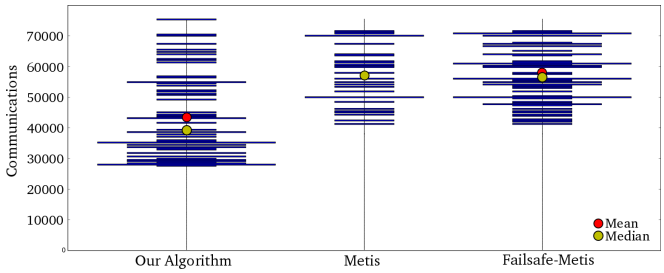

Figure 2: Distributions of the volume of communications of 64 bi-partitions of a mesh of 22800 cells with 3 criteria returned by our algorithm, Metis and *fail-safe*-Metis. Solutions whose unbalance is greater than 5% are discarded. The lower the bars, the better.

their variance and maybe worst solution found, and not only the mean and best solution found.

In this case, Metis only returned 50% of valid solutions, that is why we compare with *fail-safe*-Metis. Our algorithm seems to achieve better solutions. However, we see that we must improve the variance to avoid local minima where Metis also seems to be attracted.

5 Key points

We present a multi-level hypergraph partitioning algorithm adapted to multi-criteria. Contrary to most state-of-the-art algorithms used for multi-criteria partitioning, it is focused on balancing load for each criteria. Its results are thus valid with respect to the prescribed tolerances, which is already a significant improvement over tools such as Metis.

We describe three experiments, and show the large discrepancy between results in term of partition quality for all algorithms, even for a mono-criterion partitioning. This discrepancy illustrates that hypergraph partitioning is still a hard problem, even on topologically simple objects such as meshes.

Even if our algorithm complexity is equivalent to state-of-the-art ones, we currently only have a Python prototype which is not efficient enough in term of run times to tackle the large meshes we are interested in. We are thus currently implementing our multi-criteria multi-level algorithm in Scotch. Meanwhile, we plan on improving our algorithm to converge faster, exploiting more efficiently hypergraph properties that came from mesh topology (low degree vertices, etc.). We are hoping that it will provide a credible alternative to Metis to partition multi-physics meshes, by returning balanced partitions of multi-criteria graphs.