New graph partitioning techniques for load balancing of coupled simulation

Maria Predari, Aurélien Esnard

New graph partitioning techniques for load balancing of coupled simulation

Maria Predari[1], Esnard Aurelien[2]

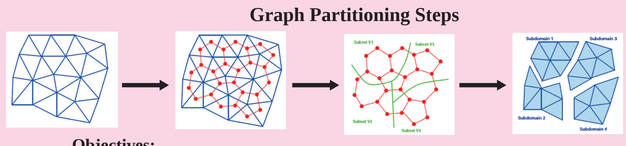
1. Inria, University of Bordeaux, France (maria.predari@inria.fr), 2. Inria, University of Bordeaux, France (aurelien.esnard@inria.fr).

Introduction

Load balancing in scientific computing

In the field of scientific computing, the load balancing is a crucial issue which determines the performance of parallel applications. A very common approach to solve the load-balancing problem is based on graph model. To equilibrate the load between N processors, one performs a (N-way) graph partitioning in N parts, each part being assigned to a given processor.

Graph Partitioning Steps



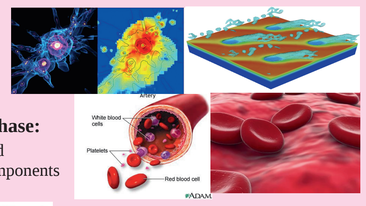
Graph Model:
- vertex represents computation task
- edge represents communication cost

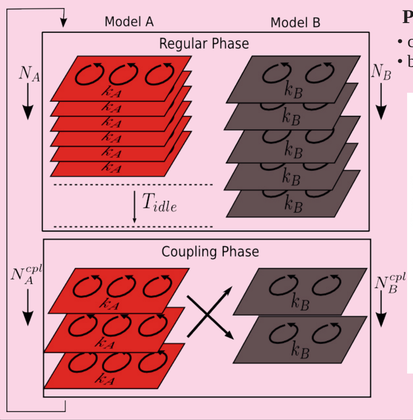
Objectives:
- balance computational load (nb vertices per part)
- minimize communication costs (nb edgecut)

Unfortunately: NP-hard problem!

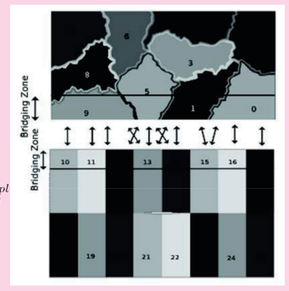
Multi-physics, Multi-scale, Coupled Simulations

Applications that emerge nowadays in real-life problems often have more complex structure and may consist of numerous component simulations, coupled together representing different models.



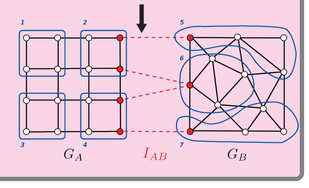


Problem during coupling phase:
- computations highly imbalanced
- bad data distribution among components



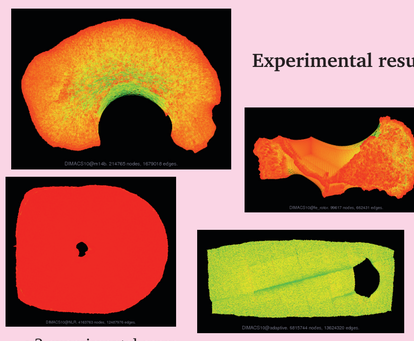
Solution:

Find a coupling-aware partitioning that takes into account the coupling phase explicitly as a trade off to the partitioning during the regular phase.



Results

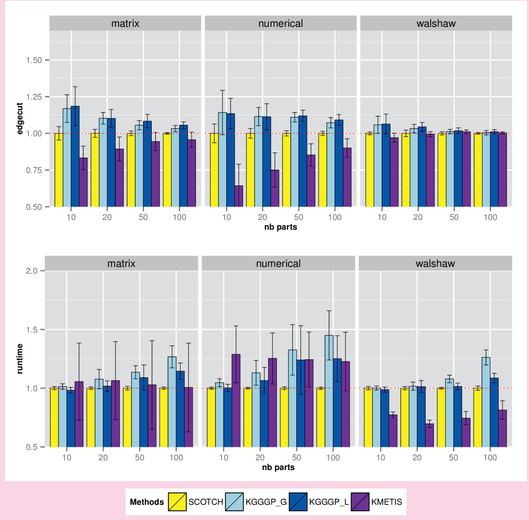
Experimental results of KGGGP algorithm on Dimacs graph dataset



- 3 experimental cases
- 2 implementations KGGGP_G, KGGGP_L (global and local)
- comparison with SCOTCH, kMETIS, PaToH, Zoltan
- evaluate partitioning quality, time performance

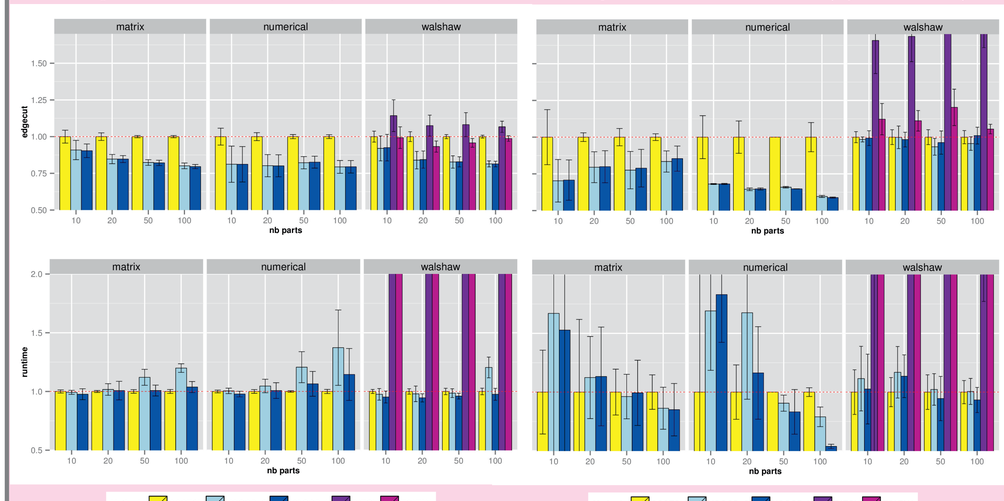
group	graph	# vtx	# edges	avg d°	min d°	max d°
walshaw	fe_rotor	99 617	662 431	13.30	5	125
walshaw	144	144 649	1 074 393	14.86	4	26
walshaw	wave	156 317	1 059 331	13.55	3	44
walshaw	m14b	214 765	1 679 018	15.64	4	40
matrix	audikw1	943 695	38 354 076	81.28	20	344
matrix	ecology1	1 000 000	1 998 000	4.00	2	4
matrix	thermal2	1 227 087	3 676 134	5.99	2	10
matrix	af_shell10	1 508 065	25 582 130	33.93	14	34
numerical	NACA0015	1 039 183	3 114 818	5.99	3	10
numerical	333SP	3 712 815	11 108 633	5.98	2	28
numerical	NLR	4 163 763	12 487 976	6.00	3	20
numerical	adaptive	6 815 744	13 624 320	4	2	4

Experiment1: no initial fixed vertices



Experiment2: initial fixed vertices (bubble scheme)

Experiment3: initial fixed vertices (repartition scheme)



Experimental results of copartitioning algorithms AWARE and PROJREPART

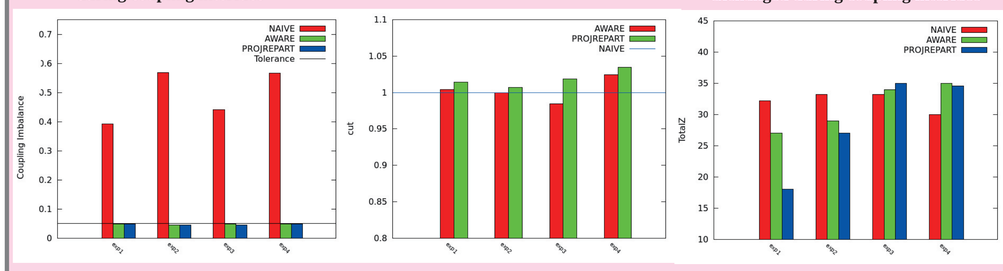
- synthetically generated mesh structures
- use of KGGGP implentation as partitioning algorithm
- dependancies on the surface of mesh structures
- comparison with NAIVE method
- imbalance tolerence up to 5% of total weight

Exp.	Graph A	Graph B
exp1	cube-hexa-25x25x25	cube-hexa-100x100x100
exp2	cube-hexa-25x25x25	cube-hexa-70x70x70
exp3	cube-tetra-40630	cube-hexa-100x100x100
exp4	cube-tetra-40630	cube-tetra-486719

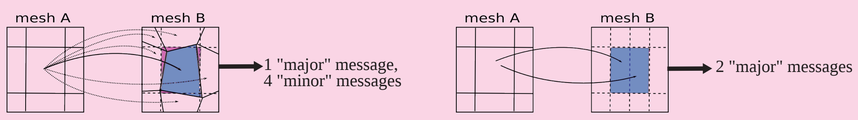
Results on coupling imbalance during coupling interface

Results on global edgecut

Results on total number of messages exchanged during coupling interface



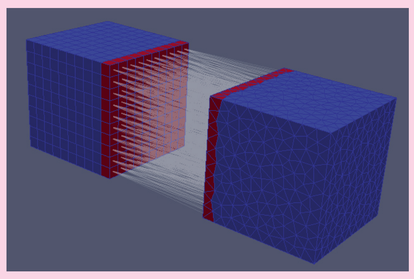
Types of messages for components with different discretization alignment



Major messages are prefered since with one communication cost we exchange more information!

Methods

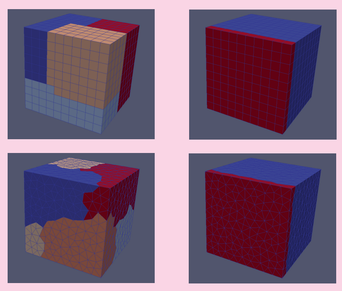
Example of coupled simulation with 2 components A, B (cubic structures) and coupling intersection on the surface.
Each structure has different discretization (hexaedric and tetraedric).



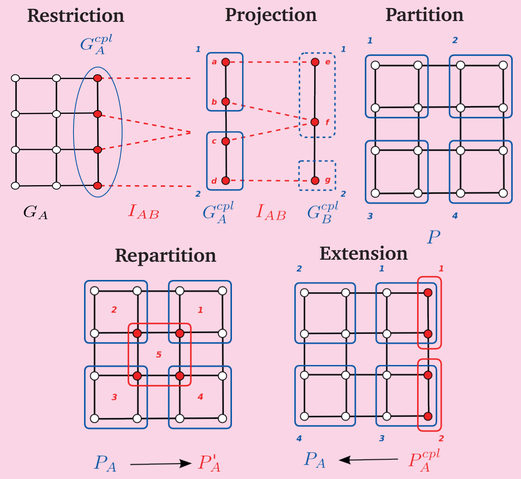
NAIVE: each component model is partitioned independently

- NAIVE is used as the state-of-the-art for copartitioning
- components are not aware of their coupling interfaces

Imbalance during coupling phase!



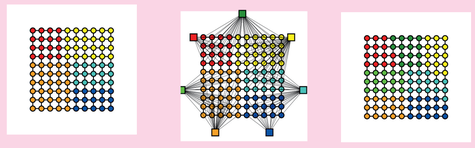
Graph Operators



- **Restriction:** returns a subgraph of a graph induced to a set of edges.
- **Partition:** returns a balanced partition of a graph or subgraph with respect to an imbalance factor.
- **Projection:** returns a N-way partition on a graph similar to a N-way partition on another graph.
- **Repartition:** returns a M-way balanced partition using a former N-way partition on the same graph.
- **Extension:** returns a balanced partition using an existing partition on a subgraph of the graph.

Attention: some operators need to handle fixed vertices

Repartition example from 5 to 7 parts

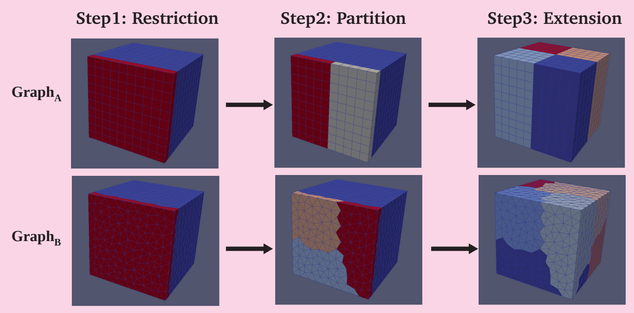


Additional "square" vertices should remain in place after the partitioning

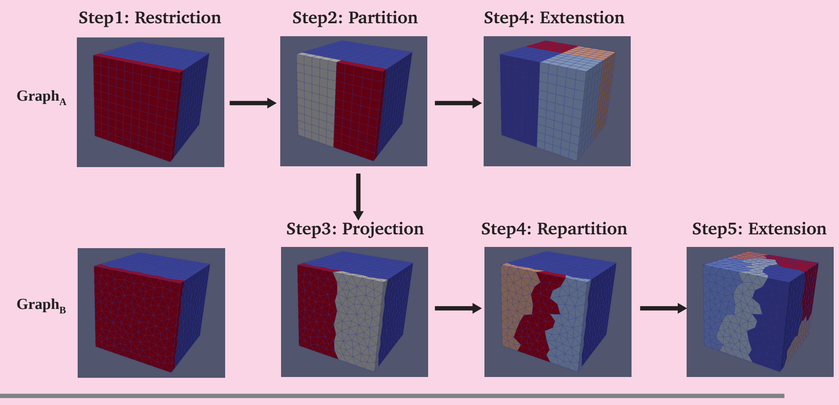
Copartitioning Algorithms

To describe our co-partitioning algorithms, we use sequences of graph operators that may be implemented with any known partitioning algorithm, such as recursive bisection, spectral based partitioning, etc

AWARE: each component model is aware of it's coupling interface



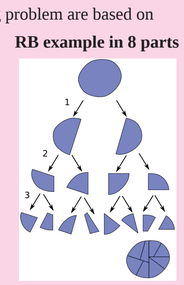
PROJREPART: components are also aware of other component's coupling interface



Which algorithm we use for the actual partitioning and why?

Nowadays, the most common approaches to solve the graph partitioning problem are based on the multilevel approach to compress the problem and on the recursive-bisection heuristic to solve it on a smaller instance. The partition is then projected back to the original graph structure applying local refinement algorithms.

RB example in 8 parts

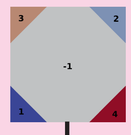


Why RB paradigm is not good for the copartitioning

Extension, Projection and Repartition operators need to handle vertices that must remain in place during the partitioning procedure (fixed vertices)

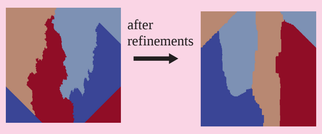
Motivating example

Partition of square graph with fixed vertices using RB based algorithm (in 4 parts)



Vertices on the corners of the square graph are initially fixed to parts

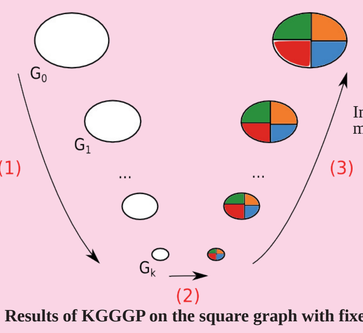
before refinements



RB based algorithm can not successfully respect both the inherent numbering constraint and the additional constraint of fixed vertices. This problem mainly appears when the part numbering of fixed vertices opposes to those of RB.

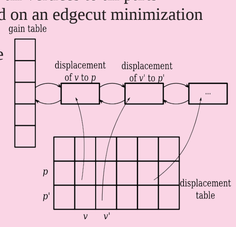
Recursive Bisection fails to partition the graph!!
- Large number of edgecut
- Disconnected components

A Multilevel k -way Greedy Graph Growing Partitioning with Initial Fixed Vertices (KGGGP)



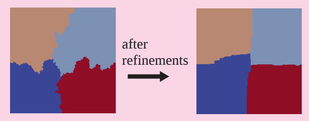
(1) Coarsening
(2) Direct initial partitioning
(3) Uncoarsening with k-way refinements

Implementation of KGGGP inside SCOTCH's multilevel framework:
- extension of greedy bisection to k parts
- at each step, selection of the best global displacement of all vertices to all parts
- selection based on an edgecut minimization criterion
- use of FM-like structures



Results of KGGGP on the square graph with fixed vertices

before refinements

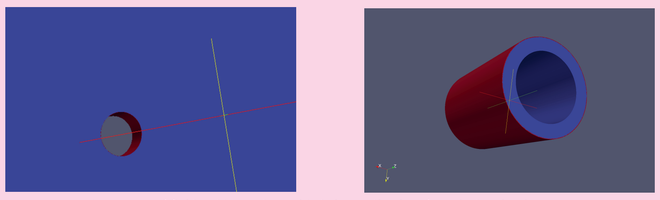


KGGGP sucessfully partition the graph with initial fixed vertices in 4 parts as opposed to RB!

Conclusions

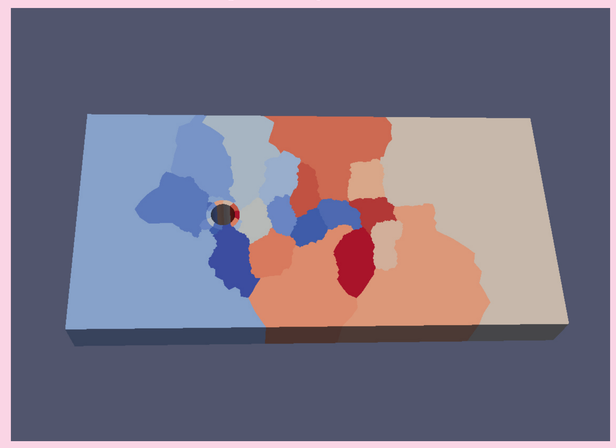
Algorithmic conclusions
- a new algorithm for graph partitioning KGGGP that handles initial fixed vertices
- two new copartitioning algorithms, AWARE and PROJREPART for coupled simulations

Copartitioning conclusions
- good load balancing during the coupling phase for both methods at a slight increase of edgecut
- in simple experiments the number of messages exchanged between components is minimized



graph structure of fluid and heat solver with their coupling interfaces (red regions)

Copartitioning overview



Future Work
- test copartitioning algorithms with real life coupled simulations
- implement parallel version of KGGGP algorithm

Example

In the field of aeronautic propulsion the behavior of hot components is impacted by complex interactions between different physics such as turbulent combustion, radiation and heat conduction. To predict such thermal environments, coupling of these different heat transfer models is necessary.

- 2 codes: fluid flow solver, heat transfer solver for solids
- fluid solver: hexaedric discretization
- heat solver:hexaedric and prism discretization
- coupling in the surfaces