Graph Operators for Coupling-aware Graph Partitioning Algorithms

Maria Predari, Aurélien Esnard

Graph Operators for Coupling-aware Graph Partitioning Algorithms

Maria Predari Aurélien Esnard

Université de Bordeaux

INRIA HiePACS & LABRI

CIMI15
July 2015

Outline

Outline

Context

Efficiency of numerical simulations requires a distribution of the computational mesh to several processors

Load Balancing Problem

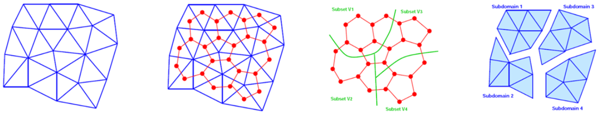
Distribute equally the load is crucial for the performance of simulations

A common approach is based on graph representation:

- vertex $\rightarrow$ process
- edge $\rightarrow$ dependency

Graph Partitioning Problem

Divide the graph in equal parts and assign them to different processors



[Teresco, J. D., Devine, K. D., and Flaherty, J. E. (2006)]

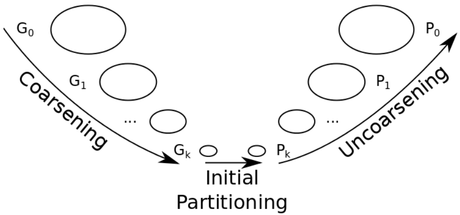
NP-hard problem $\rightarrow$ heuristic algorithms

Multilevel Graph Partitioning

Most common graph partitioning techiques use a multilevel framework to simplify the partitioning

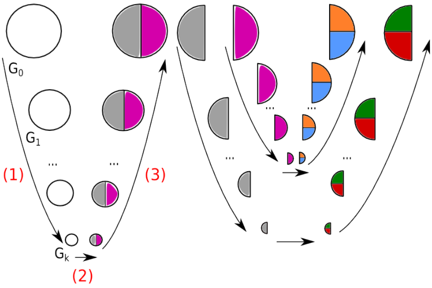
3 phases of a V-cycle:

1. Coarsening Phase
   vertex contraction: HEM
2. Initial Partitioning Phase
   perform partitioning with any known algorithm: bisection, spectral, etc.
3. Uncoarsening Phase
   Projection of the initial partitioning back to the initial graph (refinement)



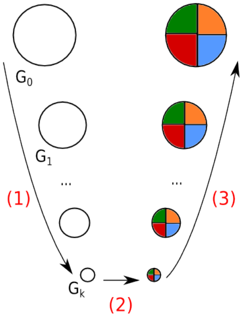
Multilevel Graph Partitioning

Multilevel Recursive Bisection
(MLRB)



k-1 V-cycles for a k-way partitioning

Multilevel k-way Partitioning
(MLKW)

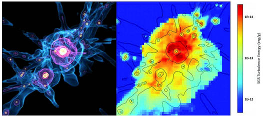


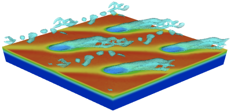
1 V-cycle with direct k-way partitioning

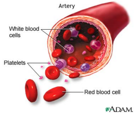
Challenges

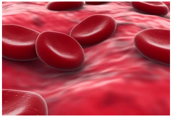
- emerging complex simulations where traditional load balancing strategies are not adequate
- multi-scale, multi-physics, coupled simulations

Domain	Simulation	Type	Components
astrophysics	structure and evolution of stars	multiplysics	Newtonian gravitational interactions model
			star aging model
biology	modeling thrombus development	multiscale	macroscopic dynamics of bload flow
			microscopic interactions between cells
aerospace engineering	combustion chamber	multiphysics, coupled	Navier-Stokes equations for fluid flow
			heat tranfer solver for solid
environemental science	climate modeling	multiphysics, coupled	atmospheric model
			ocean model
			sea-ice model
			land model



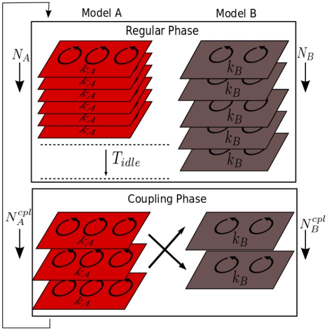






Coupled Simulation Model

A coupled simulation consists of a number of component simulations that interact periodically



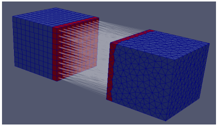
Coupled simulation model between components A and B

Regular Phase

1. Components solve individual systems
2. Parallel execution of components
3. Synchronization step (T_{idle})

Coupling Phase

1. Domains intersect - coupling interface
2. Data exchange between components



Related Work

We identify two sub-problems of coupled simulations:

1. Resource Distribution Problem:
   find nb of processors for the components on both phases

2. Data Distribution Problem:
   find a good data distribution for both phases of the coupled simulation

Related Work

We identify two sub-problems of coupled simulations:

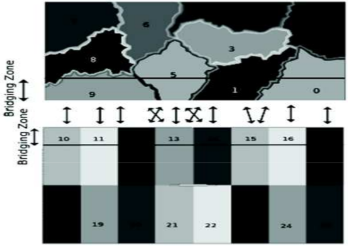
1. Resource Distribution Problem:
   find nb of processors for the components on both phases
   - empiric approach as AVBP-AVTP [F. Duchaine, et al. (2012)]
   - dynamic adaptation of processors as CESM [Graig, A.P., et al. (2012)]

2. Data Distribution Problem:
   find a good data distribution for both phases of the coupled simulation
   - `NAIVE` approach:
     partition each component independently
     $\rightarrow$ imbalance in coupling phase
   - multi-constraint graph partitioning [Karypis, G. (2003)]
     $\rightarrow$ no control on the nb of processors in different phases

Motivation

With a classic graph partitioning, during a coupling phase:

- Load is not well balanced
- Bad inter-components communications



multi-scale simulation in material physics

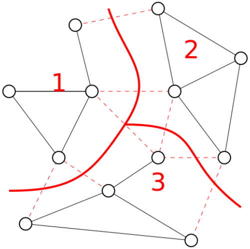
Coupling-aware partitioning (co-partitioning)

Partitioning that takes into account the coupling phase explicitly
$\Rightarrow$ as a trade-off to the partitioning in regular phase

Outline

Classic Graph Partitioning

k-way graph partitioning $\rightarrow$ divide graph in k parts. Assign it to k processors



3 *parts of weight*$= 4$ *and edgecut*$= 8$

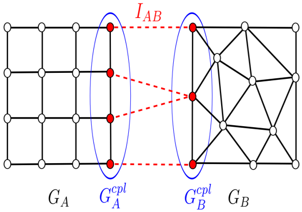
Objectives:

- balance weight between parts
- minimize number of edge cut

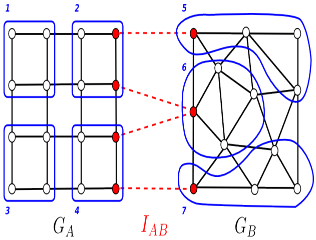
Coupled simulations

enrich the classic graph model to take into account multiple components and the coupling phase

Definition of Co-partitioning



Coupling interface with interedges



k-way co-partition with $k = k_A + k_B$

Graph Model

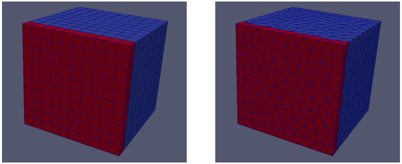
- G_X: component in regular phase
- G_X^{cpl}: component in coupling phase
- I_{XY}: inter-component communication

k-way co-partitioning Objectives

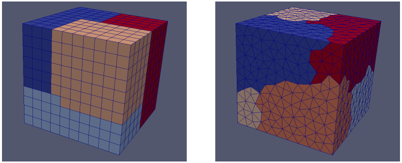
- For each X component:
  1. k_X partitioning in regular phase
  2. k_X^{cpl} partitioning in coupling interface
- For each coupling phase XY:
  3. minimize inter-component communication

Algorithms

State-of-the-art approach: each component is partitioned independently (`NAIVE`)



coupling interfaces of meshA and meshB

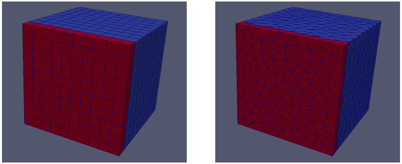


naive partitions of meshA and meshB

<u>Problem:</u>
Components are not aware of their coupling interfaces ⇒
Imbalance during coupling phase!

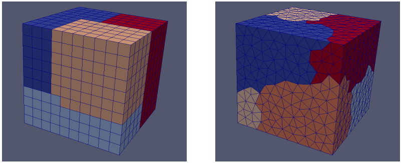
Algorithms

State-of-the-art approach: each component is partitioned independently (NAIVE)



coupling interfaces of meshA and meshB

Problem:
Components are not aware of their coupling interfaces ⇒
Imbalance during coupling phase!



naive partitions of meshA and meshB

We propose two *k*-way co-partitioning algorithms:

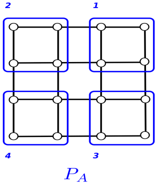
- AWARE
- PROJREPART

Algorithmic description as a sequence of graph operators:

Partition, Restriction, Projection, Repartition, Extension [Predari, M., Esnard, A. (14)]

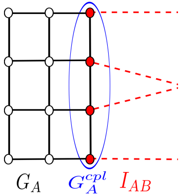
Partition

computes a partition of a graph with any algorithm



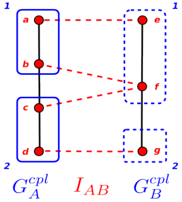
Restriction

finds the subgraph in coupling phase



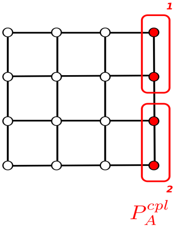
Projection

computes a partition similar to another one



Graph Operators II

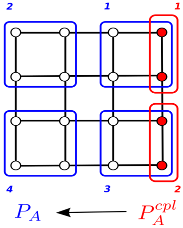
<u>Extension</u>: extends an initially smaller partition to the rest of a graph



- Input: initial partition P_A^{clp} on a subgraph
- Constraint: vertices in P_A^{clp} should remain fixed

Graph Operators II

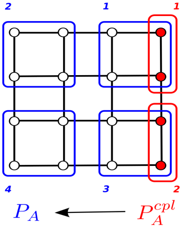
Extension: extends an initially smaller partition to the rest of a graph



- Input: initial partition P_A^{clp} on a subgraph
- Constraint: vertices in P_A^{clp} should remain fixed
- Output: partition P_A of the whole graph

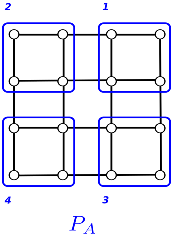
Graph Operators II

Extension: extends an initially smaller partition to the rest of a graph



- Input: initial partition P_A^{clp} on a subgraph
- Constraint: vertices in P_A^{clp} should remain fixed
- Output: partition P_A of the whole graph

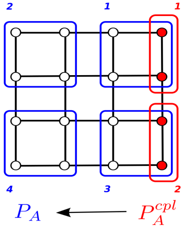
Repartition: changes the numbers of part in a partition [Vuchener , C., Esnard, A. (2012)]



- Input: initial partition P_A on the whole graph
- Objective: good migration matrix (greedy strategy)

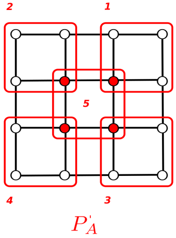
Graph Operators II

Extension: extends an initially smaller partition to the rest of a graph



- Input: initial partition P_A^{clp} on a subgraph
- Constraint: vertices in P_A^{clp} should remain fixed
- Output: partition P_A of the whole graph

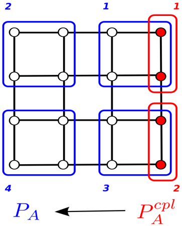
Repartition: changes the numbers of part in a partition [Vuchener , C., Esnard, A. (2012)]



- Input: initial partition P_A on the whole graph
- Objective: good migration matrix (greedy strategy)
- Output: balanced partition P'_A on the whole graph

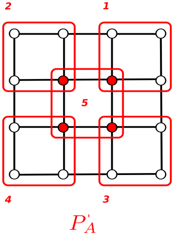
Graph Operators II

Extension: extends an initially smaller partition to the rest of a graph



- Input: initial partition P_A^{clp} on a subgraph
- Constraint: vertices in P_A^{clp} should remain fixed
- Output: partition P_A of the whole graph

Repartition: changes the numbers of part in a partition [Vuchener , C., Esnard, A. (2012)]



- Input: initial partition P_A on the whole graph
- Objective: good migration matrix (greedy strategy)
- Output: balanced partition $P_A^{'}$ on the whole graph

Partitioning implementation should handle fixed vertices!

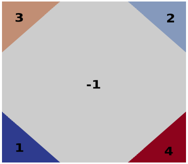
Graph Partitiong Tools for Initial Fixed Vertices

List of graph partitioning tools that handle fixed vertices.

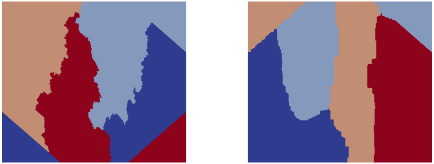
Tools	Type	Fixed	Parallel	Multilevel Scheme	Initial Part.	Available
Scotch	graph	yes	no	MLKW	RB	source
RM-Metis	graph	only *k*	no	MLKW	greedy	no
HMetis	hypergraph	yes	no	MLRB	–	binary
PaToH	hypergraph	yes	no	MLRB	–	binary
KPaToH	hypergraph	yes	no	MLKW	RB*	no
Zoltan(PHG)	hypergraph	yes	yes	MLRB	–	source

Limitations of Recursive Bisection (RB) based Algorithms

Initial fixed vertices scheme



Partitioning quality of RB



before refinements after refinements

RB fails to partition a simple graph in 4 parts with initial fixed vertices
⇒ inherent numbering constraint VS fixed vertex constraint

k-way Greedy Graph Growing Partitioning (KGGGP)

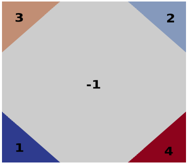
KGGGP: initial partitioning algorithm inside SCOTCH's framework:

- extension of greedy bisection to k parts
- at each step, selection of the best global displacement (v, k)
- selection based on an edgecut minimization criterion
- use of FM-like structures

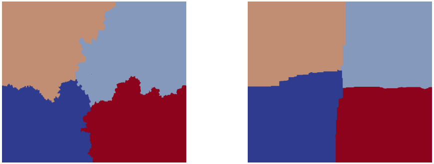
Time Complexity: $O(k|E|)$

Optimization: local selection of best displacement ([Karypis, G., Kumar, V. (1998)])

Initial fixed vertices scheme



Partitioning quality of KGGGP

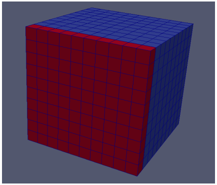


before refinements after refinements

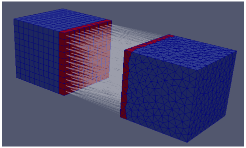
Each component is aware of it's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

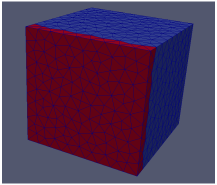
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Part(G_B^{cpl}) \rightarrow P_B^{cpl}$
5. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
6. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



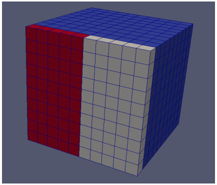
mesh B

AWARE Algorithm

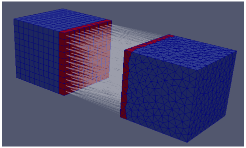
Each component is aware of it's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

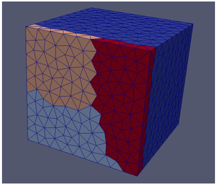
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Part(G_B^{cpl}) \rightarrow P_B^{cpl}$
5. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
6. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



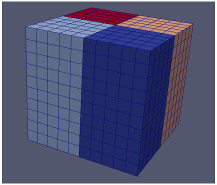
mesh B

AWARE Algorithm

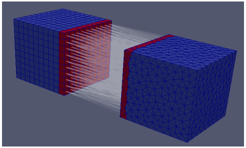
Each component is aware of it's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

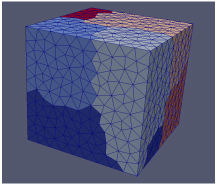
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Part(G_B^{cpl}) \rightarrow P_B^{cpl}$
5. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
6. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



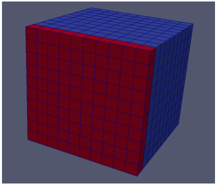
mesh B

PROJREPART Algorithm

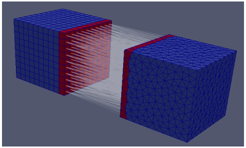
Components are also aware of other component's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

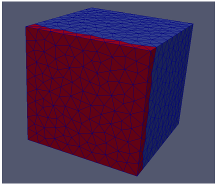
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Proj(G_A^{cpl}, G_B^{cpl}, P_A^{cpl}) \rightarrow \tilde{P}_B^{cpl}$
5. $Repart(G_B^{cpl}, \tilde{P}_B^{cpl}) \rightarrow P_B^{cpl}$
6. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
7. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



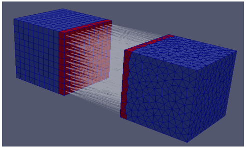
mesh B

PROJREPART Algorithm

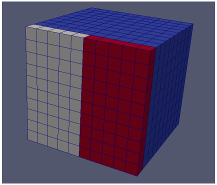
Components are also aware of other component's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

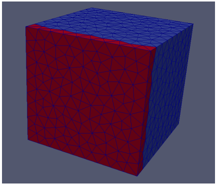
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Proj(G_A^{cpl}, G_B^{cpl}, P_A^{cpl}) \rightarrow \tilde{P}_B^{cpl}$
5. $Repart(G_B^{cpl}, \tilde{P}_B^{cpl}) \rightarrow P_B^{cpl}$
6. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
7. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



coupling overview with interedges



mesh A



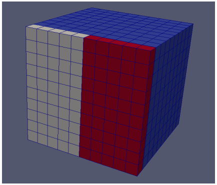
mesh B

PROJREPART Algorithm

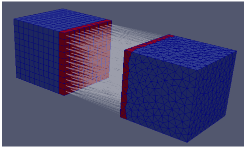
Components are also aware of other component's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

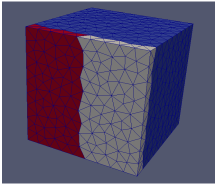
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Proj(G_A^{cpl}, G_B^{cpl}, P_A^{cpl}) \rightarrow \tilde{P}_B^{cpl}$
5. $Repart(G_B^{cpl}, \tilde{P}_B^{cpl}) \rightarrow P_B^{cpl}$
6. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
7. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



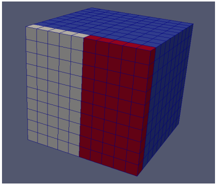
mesh B

PROJREPART Algorithm

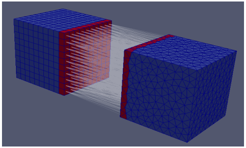
Components are also aware of other component's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

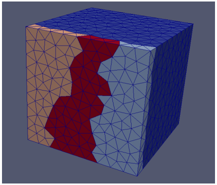
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Proj(G_A^{cpl}, G_B^{cpl}, P_A^{cpl}) \rightarrow \tilde{P}_B^{cpl}$
5. $Repart(G_B^{cpl}, \tilde{P}_B^{cpl}) \rightarrow P_B^{cpl}$
6. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
7. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



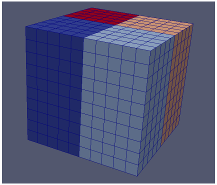
mesh B

PROJREPART Algorithm

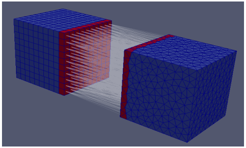
Components are also aware of other component's coupling interface

Input: $G_A, G_B, K_A, K_B, K_A^{cpl}, K_B^{cpl}$ Output: P_A, P_B

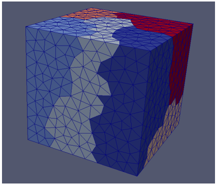
1. $Rest(G_A) \rightarrow G_A^{cpl}$
2. $Rest(G_B) \rightarrow G_B^{cpl}$
3. $Part(G_A^{cpl}) \rightarrow P_A^{cpl}$
4. $Proj(G_A^{cpl}, G_B^{cpl}, P_A^{cpl}) \rightarrow \tilde{P}_B^{cpl}$
5. $Repart(G_B^{cpl}, \tilde{P}_B^{cpl}) \rightarrow P_B^{cpl}$
6. $Ext(G_A, G_A^{cpl}, P_A^{cpl}) \rightarrow P_A$
7. $Ext(G_B, G_B^{cpl}, P_B^{cpl}) \rightarrow P_B$



mesh A



coupling overview with interedges



mesh B

Outline

Experimental Description for KGGGP algorithm

Graph Description (DIMACS'10 Collection)

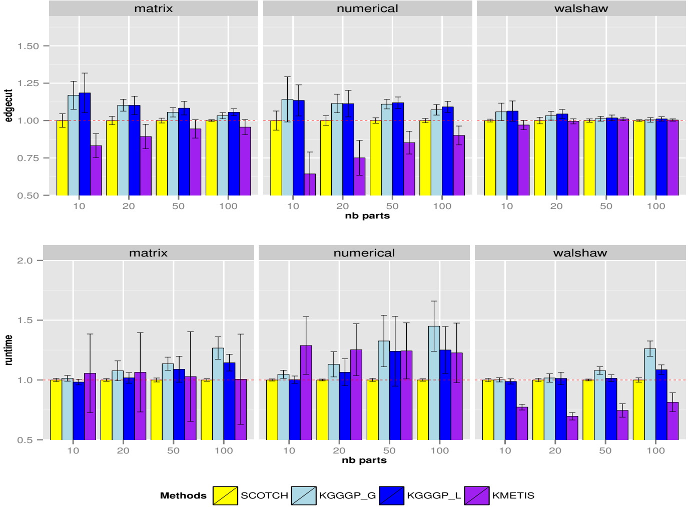
collection	graph	# vtx	# edges	avg d°	min d°	max d°
walshaw	fe_rotor	99 617	662 431	13.30	5	125
walshaw	144	144 649	1 074 393	14.86	4	26
walshaw	wave	156 317	1 059 331	13.55	3	44
walshaw	m14b	214 765	1 679 018	15.64	4	40
matrix	audikw1	943 695	38 354 076	81.28	20	344
matrix	ecology1	1 000 000	1 998 000	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	channel	4 802 000	42 681 372	17.78	6	18
numerical	adaptive	6 815 744	13 624 320	4.00	2	4

Metrics: partitioning quality, time performance

- our implementations : KGGGP_G, KGGGP_L
- tools: SCOTCH 6.0.3, kMETIS 5.1.0, PaToH 3.0, Zoltan 3.81
- imbalance tolerance = 5%
- results from 3 experiments on average values of each graph

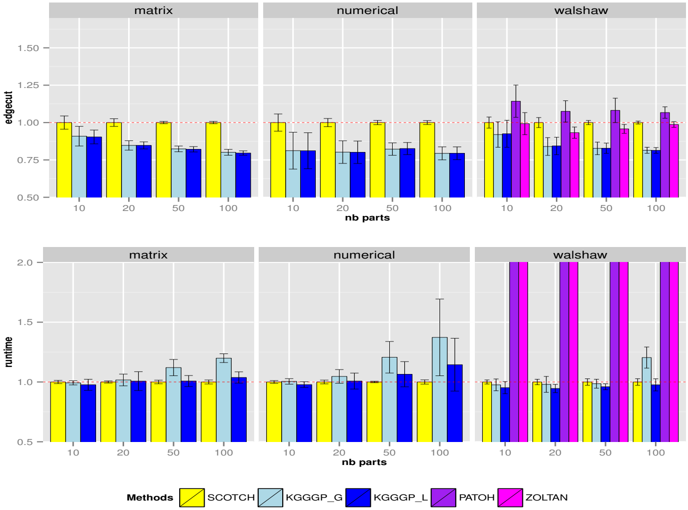
Exp1 : Evaluation of KGGGP without fixed vertices

Relative results to SCOTCH



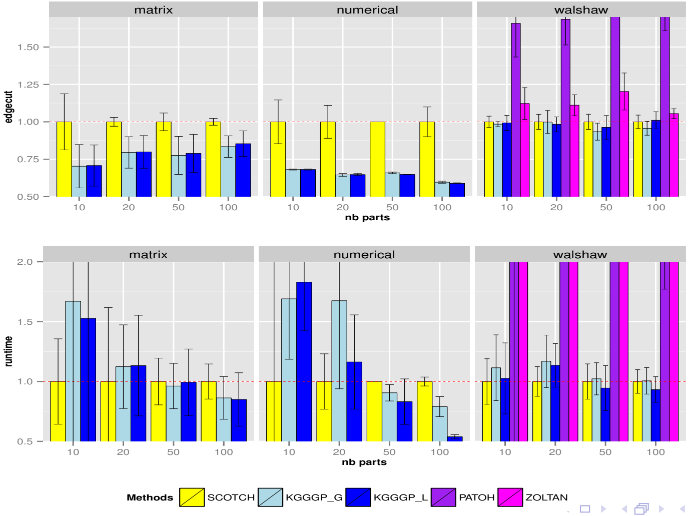
Exp2 : Evaluation of KGGGP with fixed vertices

- vertices are fixed based on a bubble scheme (seeds & levelset)



Exp3 : Evaluation of KGGGP with fixed vertices

- vertices are fixed based on a repartition scheme (initial imbalance partition)

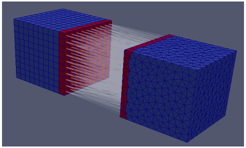


Experimental Description for co-partitioning

- synthetically generated mesh structures
- use of `KGGGP` implementation

Description of the experiments

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

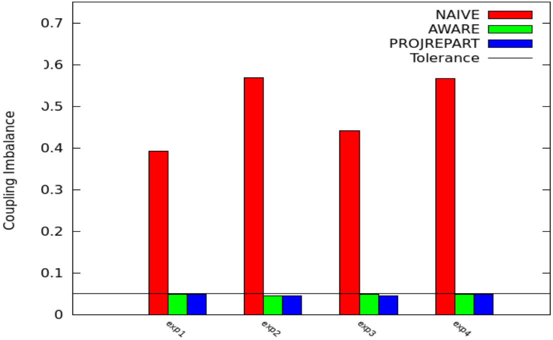


Metrics:

- load balancing in coupling phase
- global edgecut
- number of messages exchanged

Results shown for $k_A = 16$ and $k_B = 48$

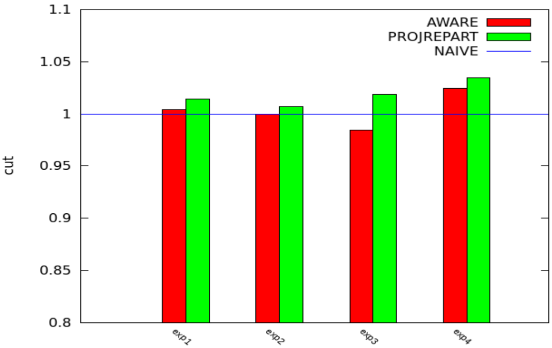
Load Balancing in Coupling Phase



Load balance during the coupling phase of mesh B

- `AWARE` and `PROJREPART` respect the imbalance tolerance (typically 5% of the ideal weight)
- `NAIVE` highly imbalanced during coupling phase

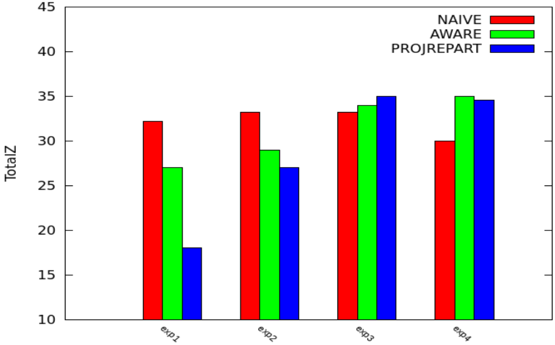
Edgecut



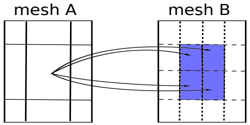
Total edgecut of mesh B

- `NAIVE` is used as the reference
- `AWARE` and `PROJREPART` do not severely degrade the total egdecut

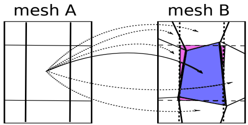
Number of Messages



Total number of messages exchanged during coupling interface



well-aligned



bad-aligned

- exp1 - exp2: hexaedric elements are well aligned
- exp3 - exp4: elements of different discretization misaligned

Outline

Conclusion and Future Work

Conclusion

1. A new algorithm for graph partitioning with fixed vertices `KGGGP`
2. Two new algorithms for graph partitioning of coupled simulations: `AWARE` and `PROJREPART`

Experimental results:

2. Good load balancing during the coupling phase for both methods at a slight increase of edge-cut
3. In simple experiments the number of messages exchanged between the two graph is minimized

Prospects for co-partitioning

- More experiments with real-life and larger graphs
- Parallel version

End

Thank you for you attention. Any questions?