PhD Defense

Cédric Lachat

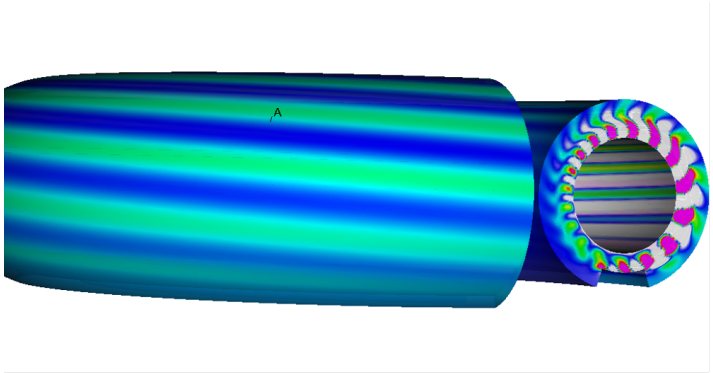
Design and validation of distributed-memory, parallel remeshing algorithms based on a sequential remesher

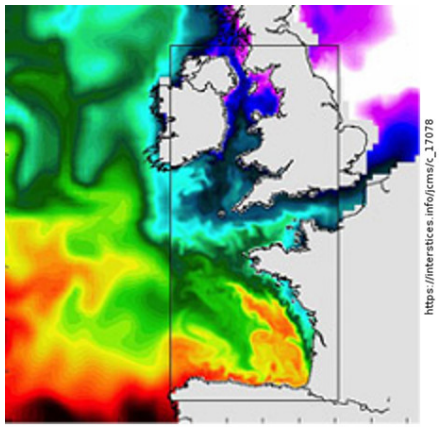
SUMMARY

1
Introduction

Context

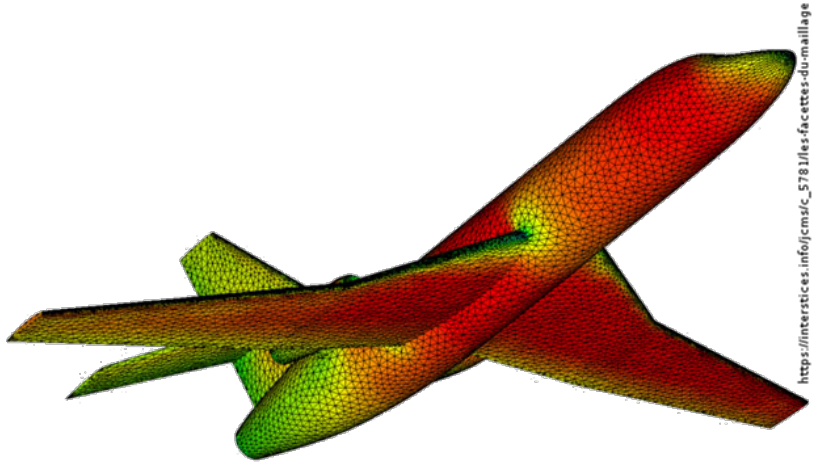
- Numerical simulations are needed in multiple domains including:
  - thermonuclear fusion
  - aeronautics
  - meteorology, ...
- Problems become bigger and more and more complex: numerical simulations cannot be run on a single workstation
  $\rightarrow$ Want of parallelized computations
  $\rightarrow$ Need to distribute data across the processors: domain decomposition

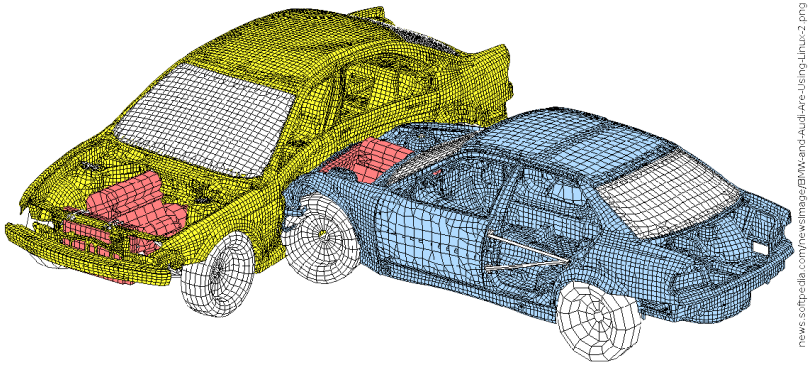




Numerical simulation

- Space discretization:
  - mesh
- Finite number of points on which values of the problem are computed, e.g.:
  - temperature
  - pressure
  - speed,...
- Solution precision depends on mesh quality:
  - need for remeshing





Aim of this PhD thesis

- Achieve parallel remeshing based on a sequential remesher
- Devise a scalable method
  $\rightarrow$ distributed memory paradigm
  $\rightarrow$ communication across subdomain boundaries
- Since distributed meshes are considered, need to devise a complete framework for handling distributed meshes and related computations
- Parallel remeshing requires:
  - to iterate on mesh components to perform local computations
  - to associate values with components

  $\rightarrow$ made available to users through an API
- Improve locality of accesses in memory hierarchy:
  - proper renumbering of entities on each subdomain

State of the art (1/2)

- Parallel partitioning
  - spectral [Pothen et al., 1990], combinatorial [Kernighan and Lin, 1970, Fiduccia and Mattheyses, 1982], evolutionist [Dongarra et al., 1988]
  - only multi-level [Chevalier, 2007, Chevalier and Safro, 2009] provides good efficiency and quality on big graph
- Parallel dynamic load balancing
  - many tools: DRAMA [Maerten et al., 1999], Zoltan [Devine et al., 1999, 2002], PLUM [Oliker et al., 2000], ParFUM [Lawlor et al., 2006]...

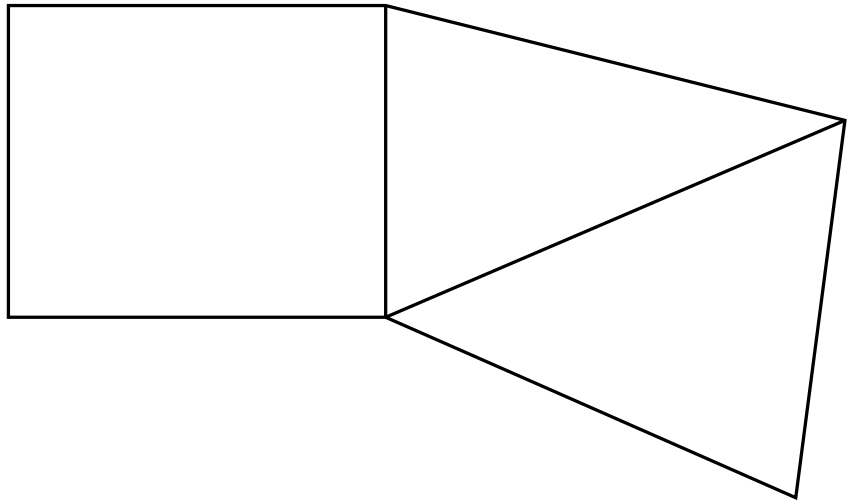
State of the art (2/2)

- Parallel remeshing
  - parallelize remeshing techniques [Castanos and Savage, 1996, Laemmer, 1997, Oliker et al., 2000, Chrisochoides and Nave, 2003, Casagrande et al., 2005, Lawlor et al., 2006]
  - re-use of sequential algorithms [Coupez et al., 2000, Cavallo et al., 2005, Dobrzynski and Remacle, 2007, Tremel et al., 2007, Digonnet et al., 2007, Ramadan et al., 2009]
- Renumbering [Löhner, 1993, Burgess and Giles, 1997, Löhner, 1998, Amenta et al., 2003, Isenburg et al., 2006, Loseille and Alauzet, 2009]

2
Data structures

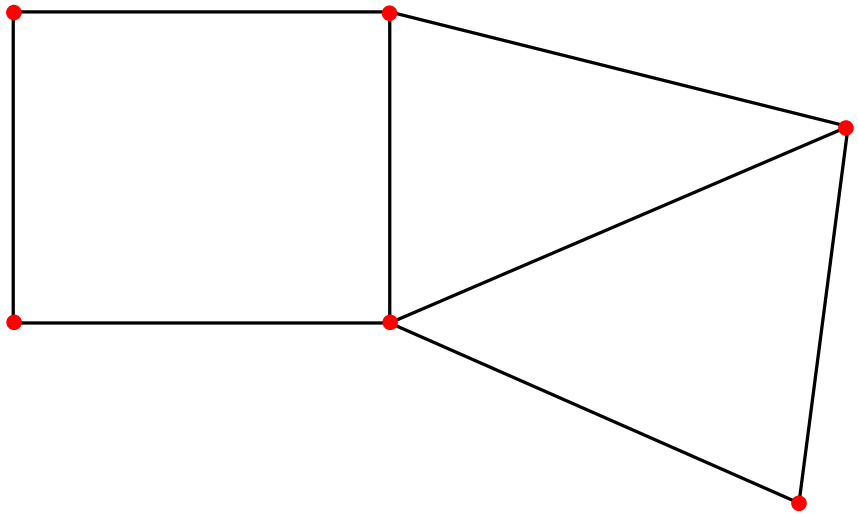
Definitions

- **Mesh:**



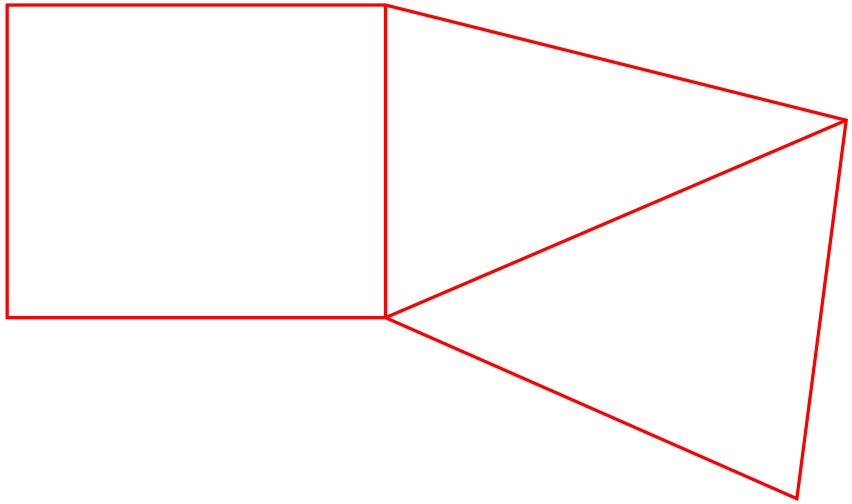
Definitions

- **Mesh:**
  - **Node**



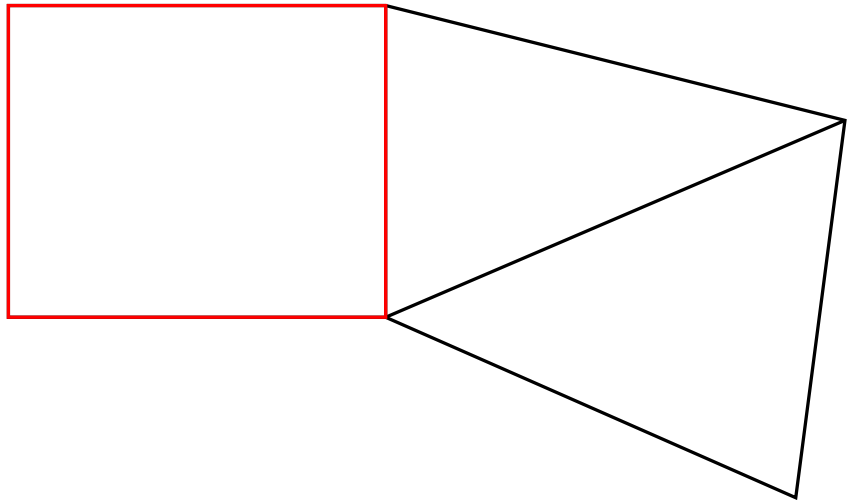
Definitions

- **Mesh:**
  - **Node**
  - **Edge**



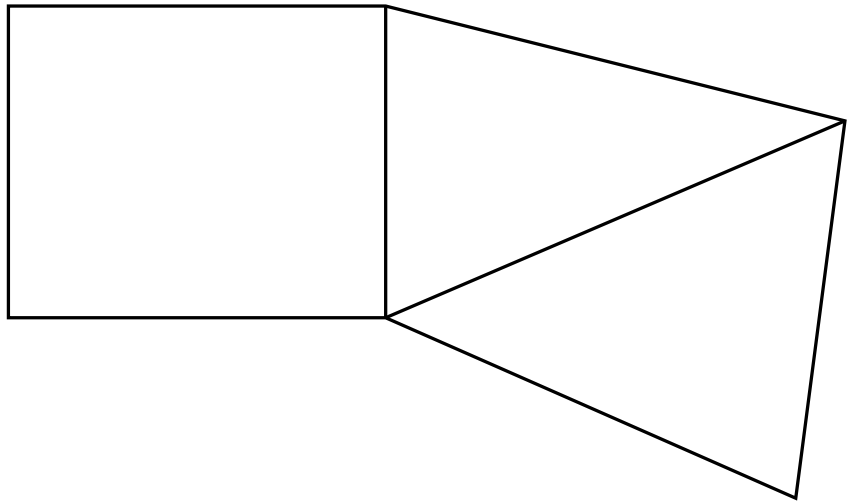
Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**



Definitions

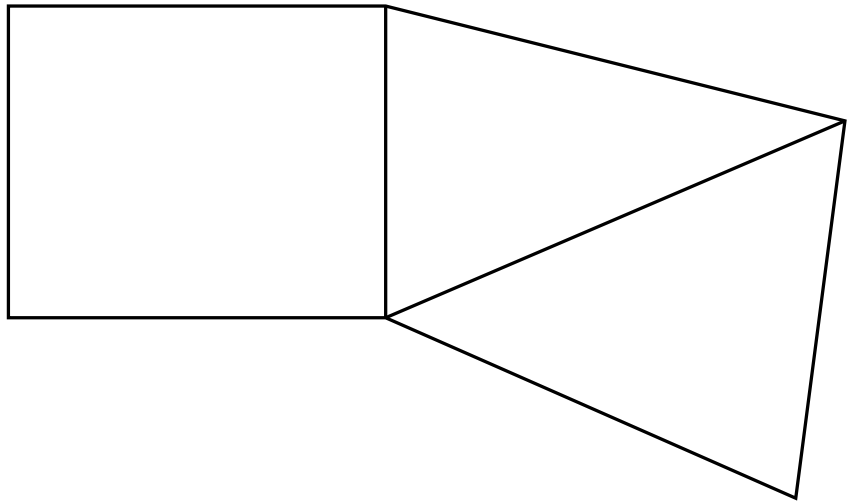
- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**



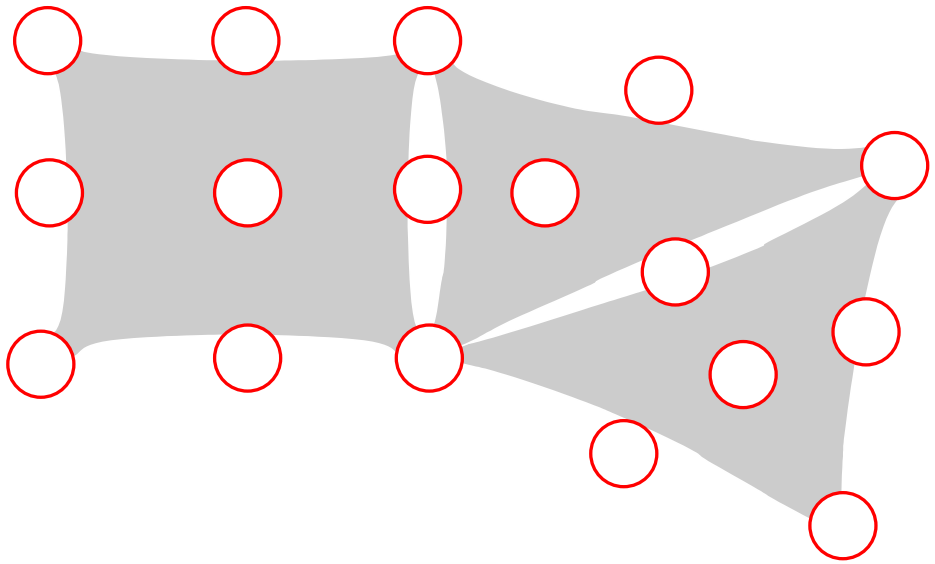
- **Mesh representation:**

Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

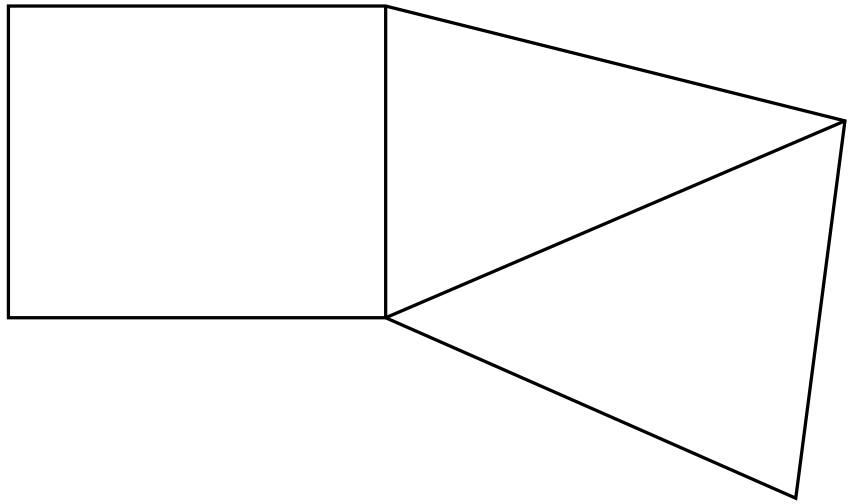


- **Mesh representation:**
  - **Vertex**

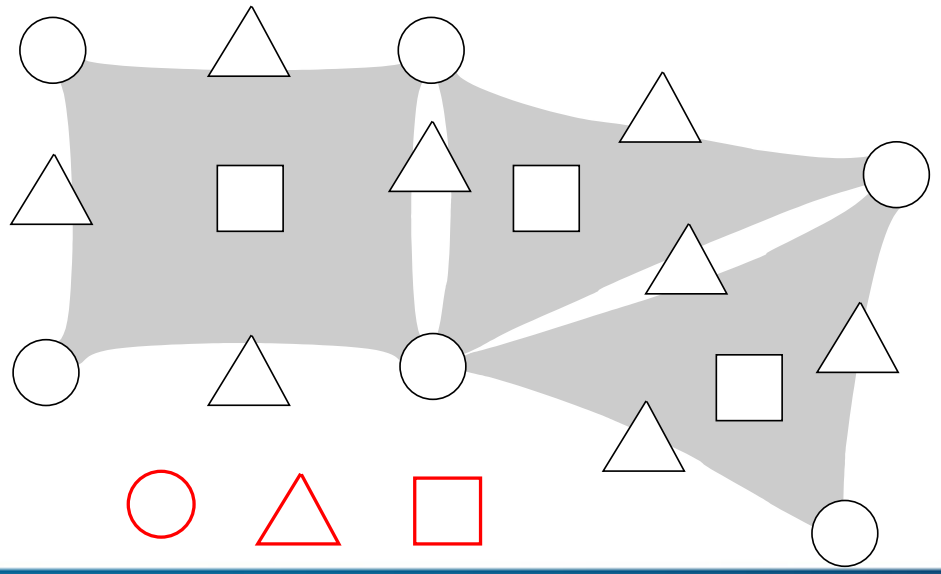


Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

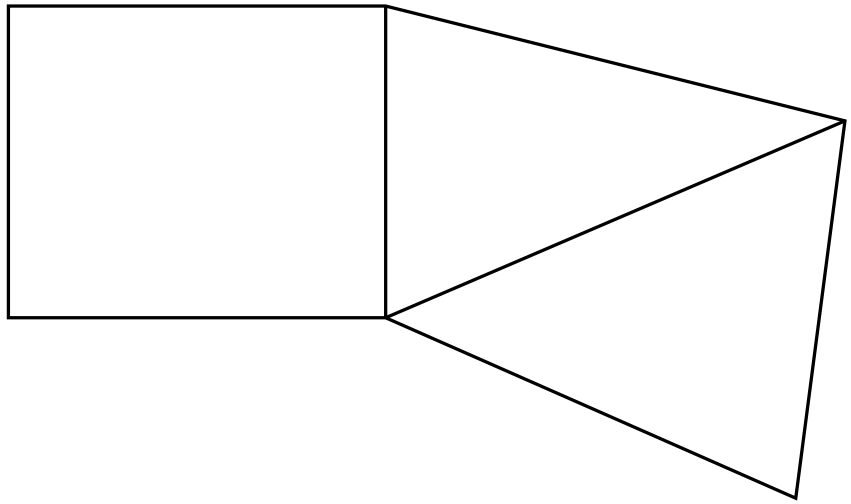


- **Mesh representation:**
  - **Vertex**
  - **Entity**

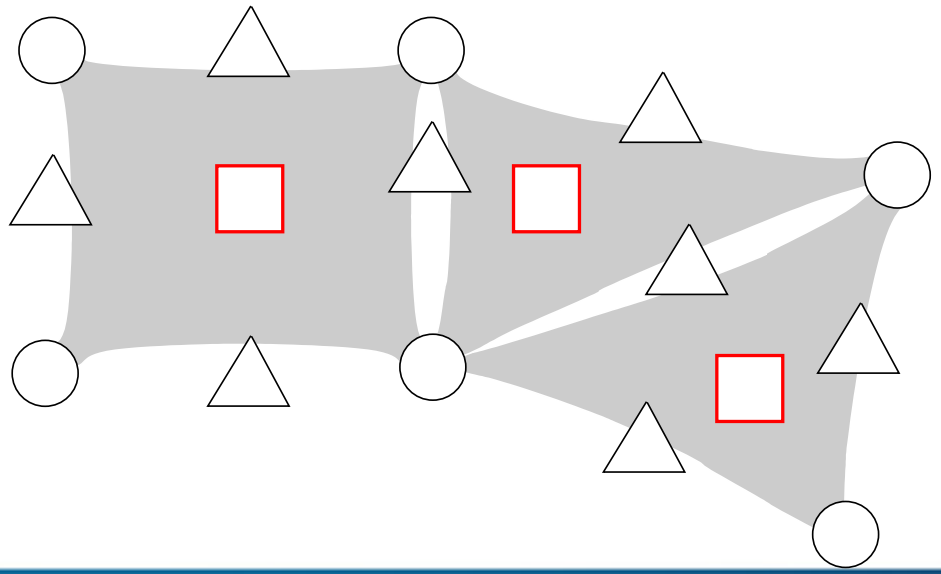


Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

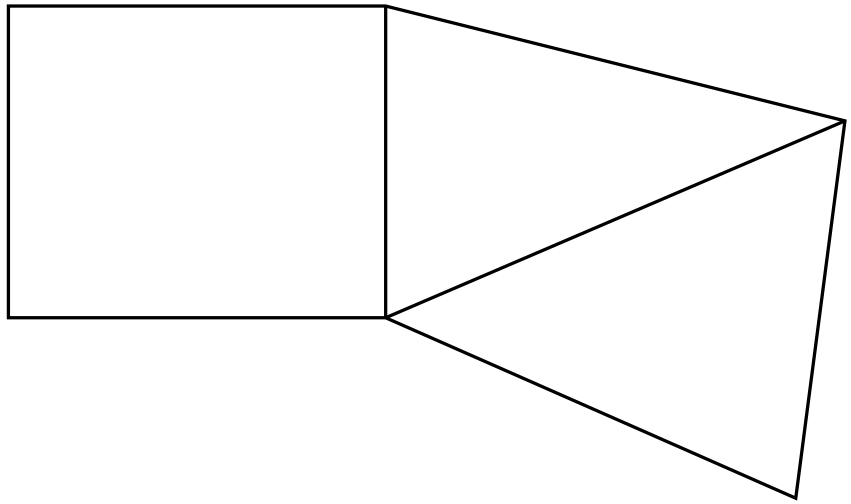


- **Mesh representation:**
  - **Vertex**
  - **Entity**
    - **main entity**

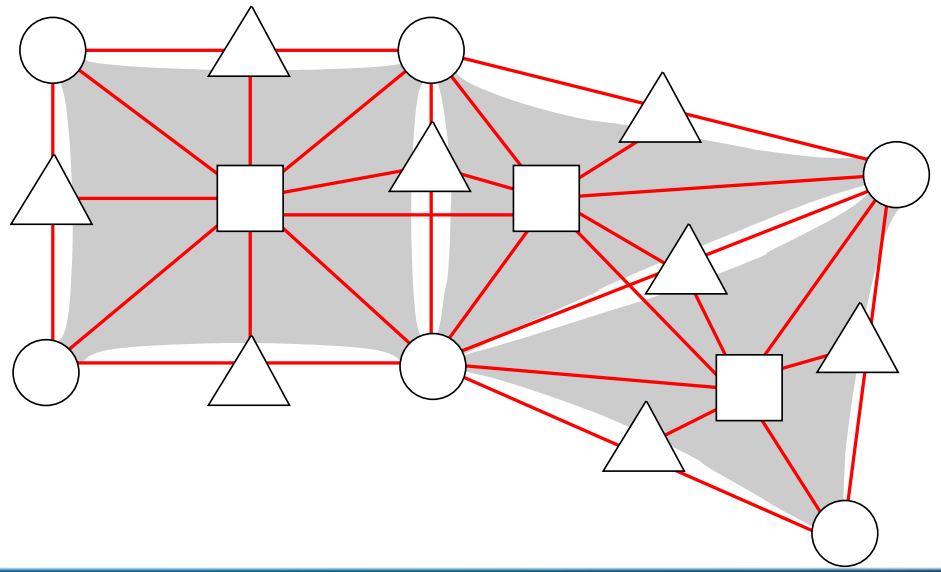


Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

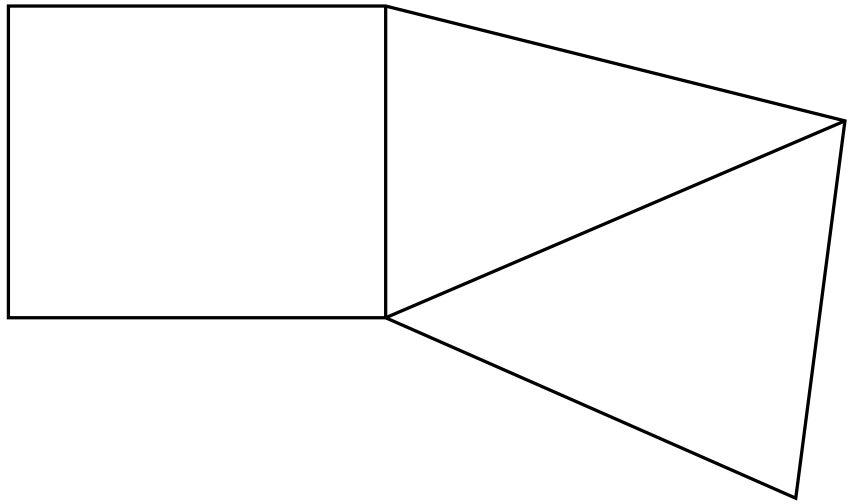


- **Mesh representation:**
  - **Vertex**
  - **Entity**
    - **main entity**
  - **Relation**

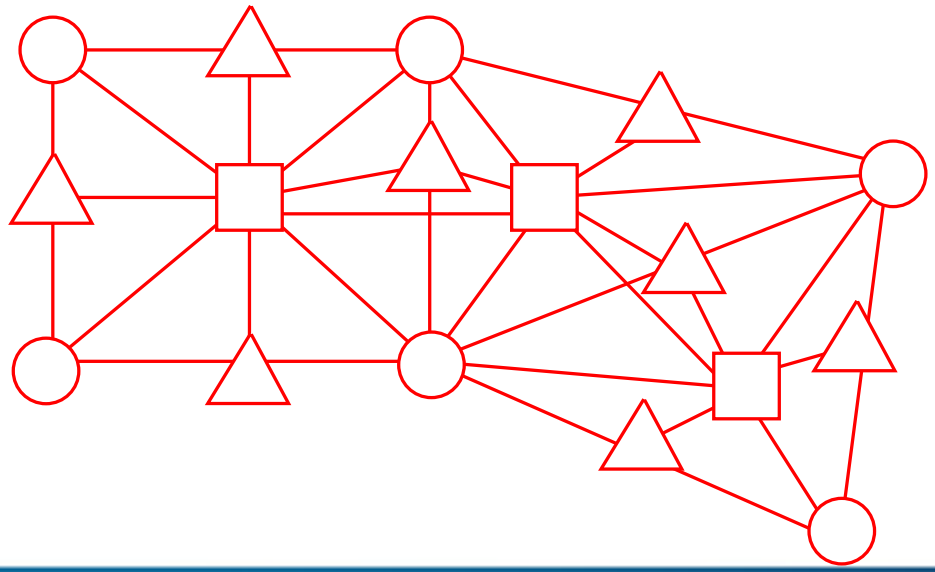


Definitions

- **Mesh:**
  - **Node**
  - **Edge**
  - **Element**

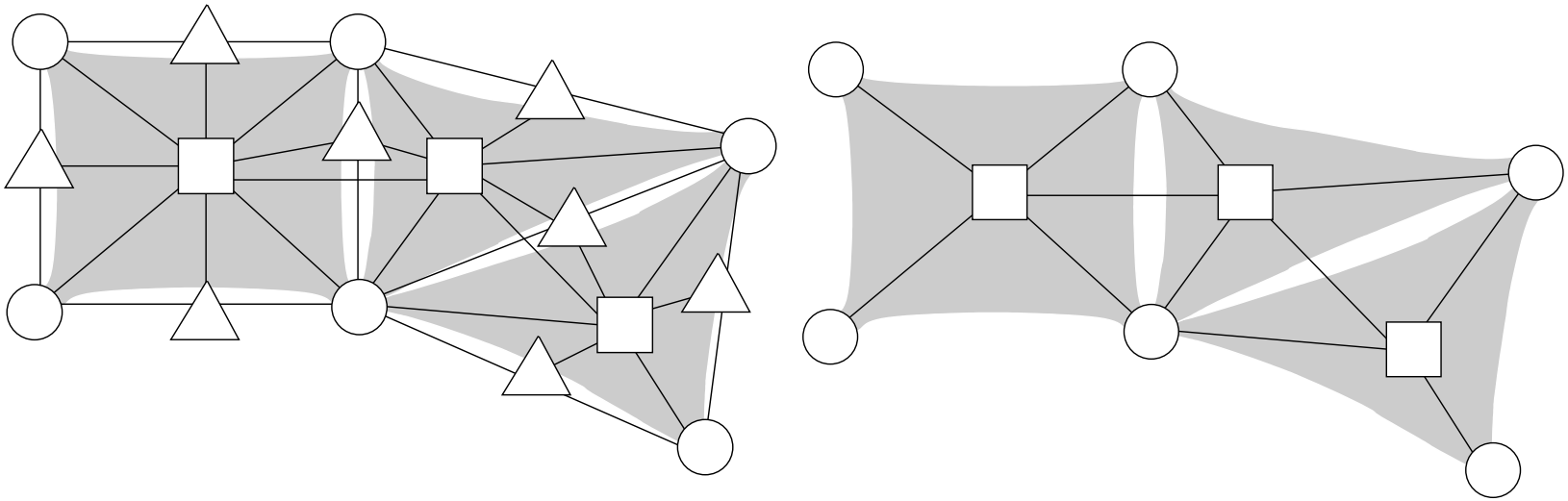


- **Mesh representation:**
  - **Vertex**
  - **Entity**
    - **main entity**
  - **Relation**
  - **Enriched graph**



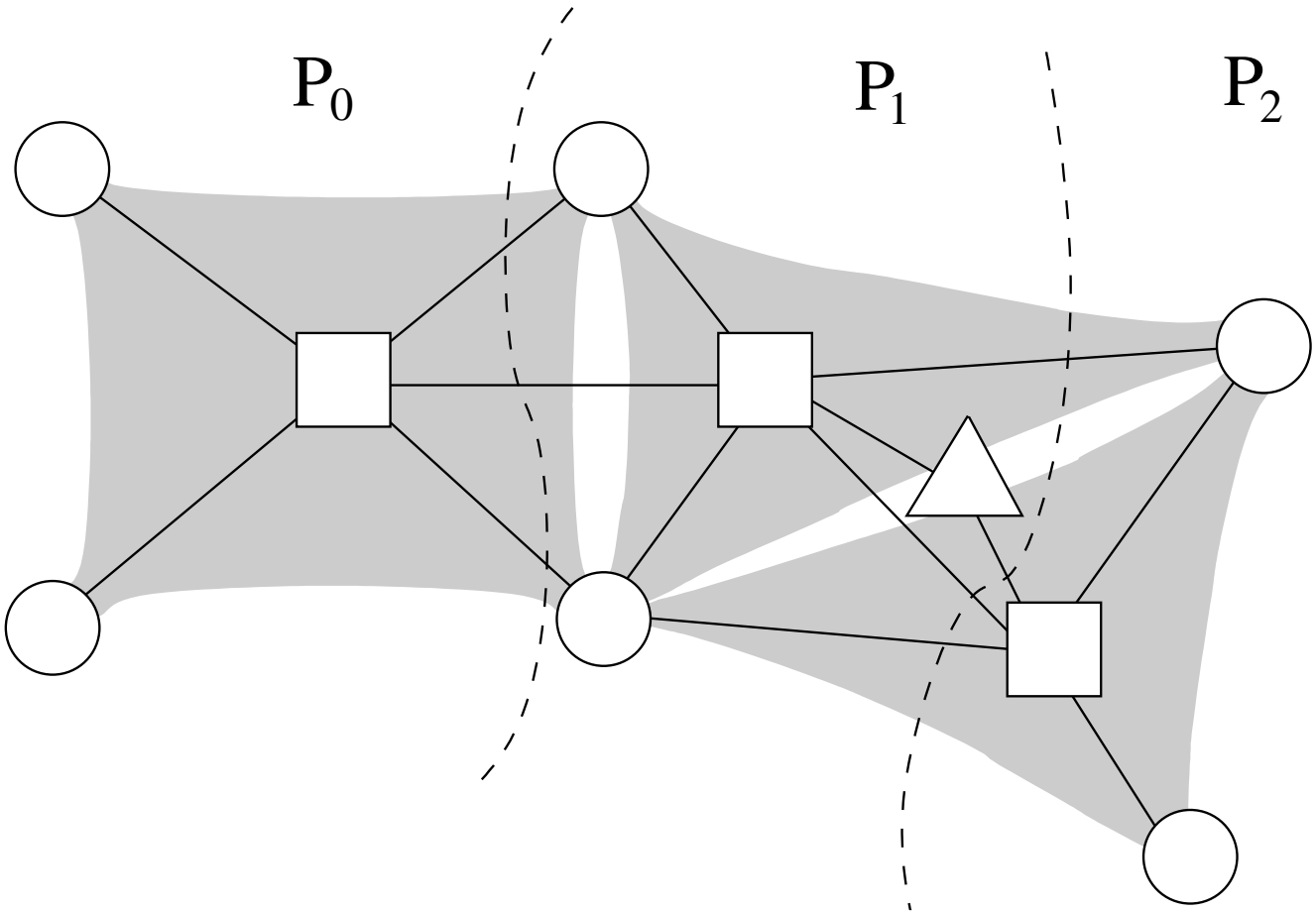
Examples

- The same mesh can lead to different enriched graphs
  - Depending on the requirements of the numerical schemes



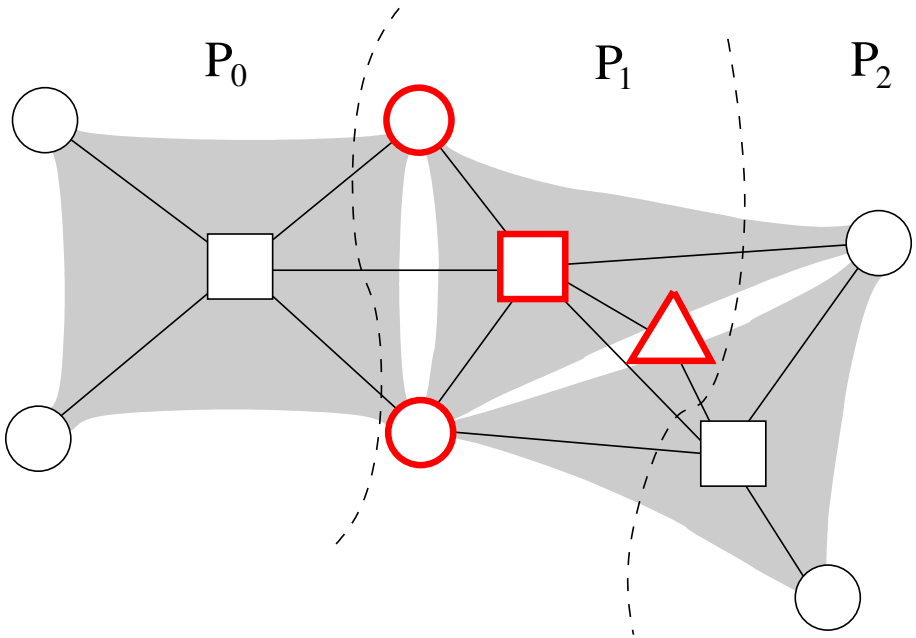
Distributed mesh

- On three processors



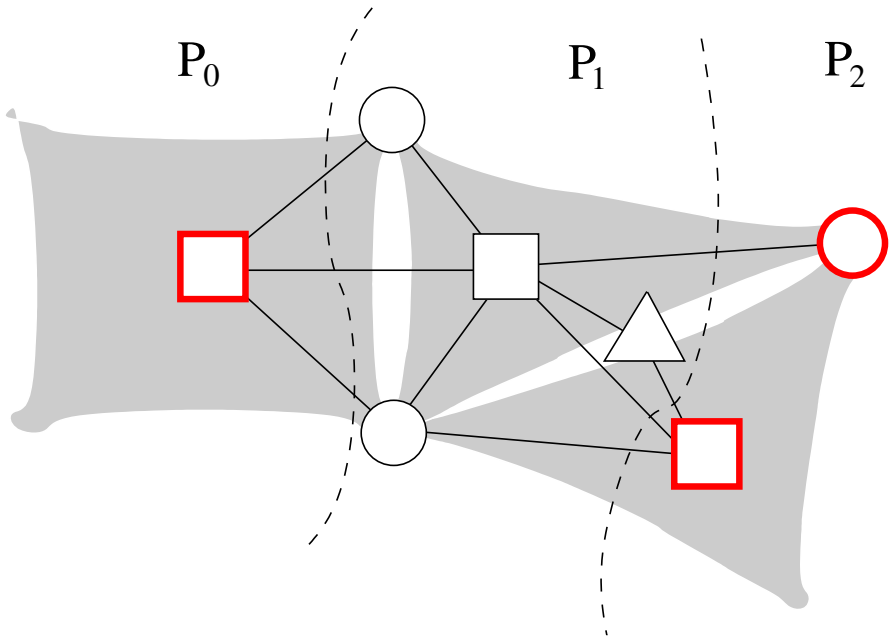
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



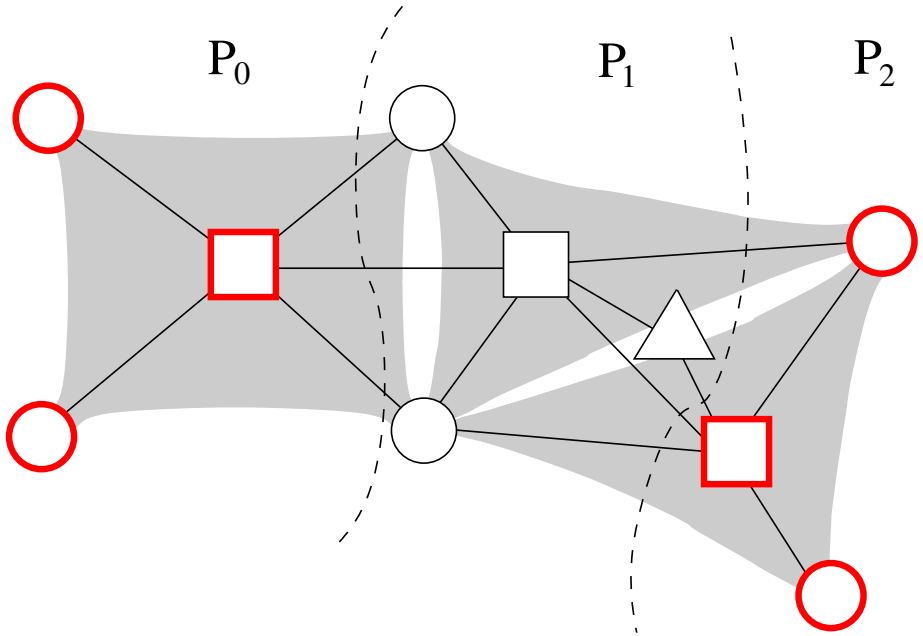
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



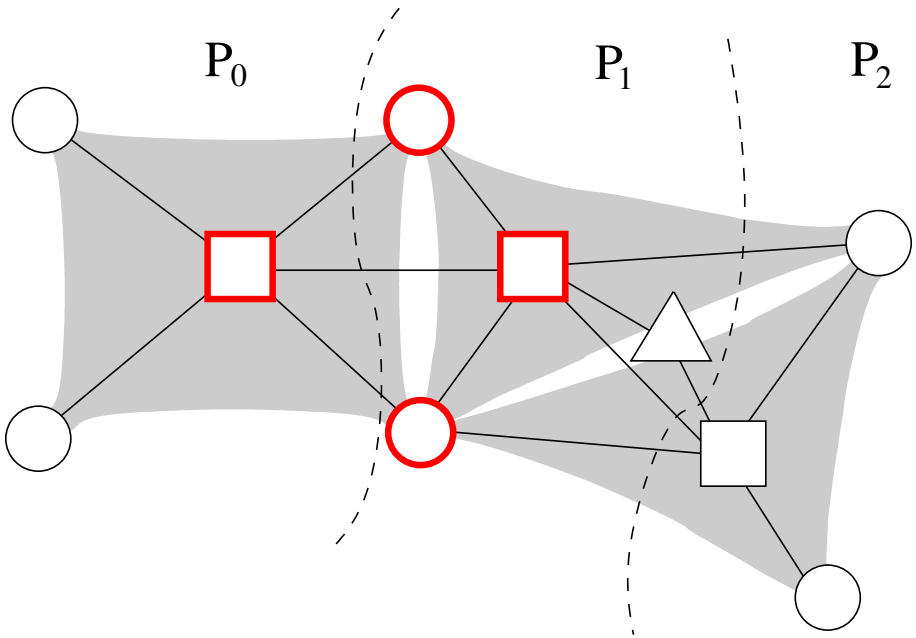
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



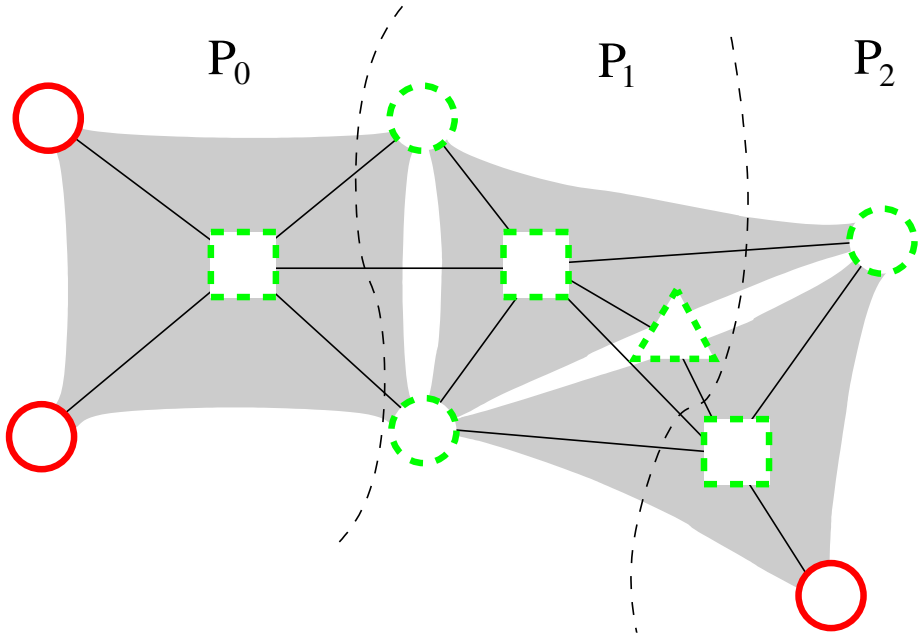
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



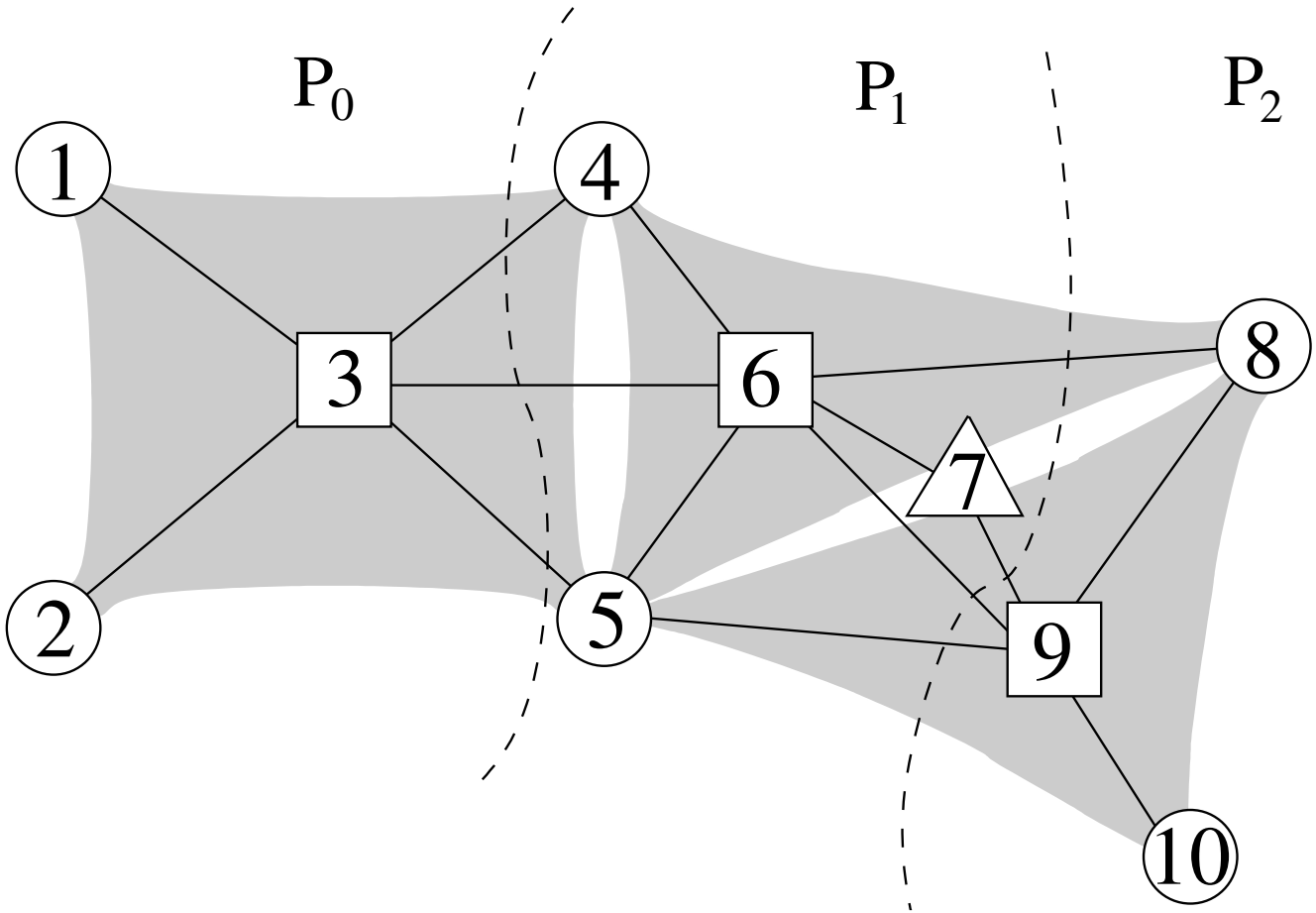
Definitions

- Vertices, locally on P_1:
  - **Local**
  - **Halo**
  - **Overlap**
- Vertices, between P_0 and P_1:
  - **Frontier**
- Vertices, globally:
  - **Internal**



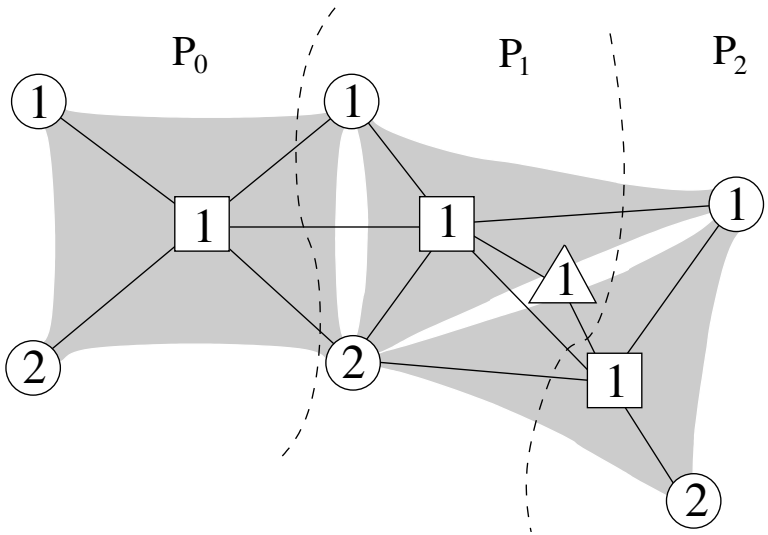
Global numbering

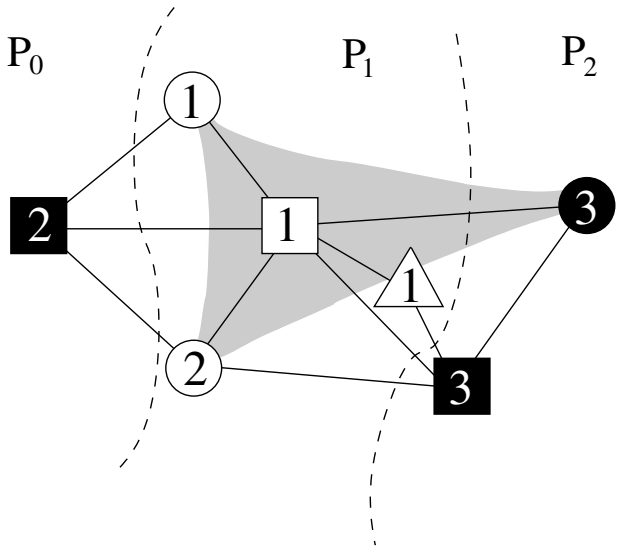
- Necessary to describe the mesh in its entirety



Local numbering

- Local indices are mandatory to address local data arrays
- For all local vertices on every subdomain, vertices are indexed per entity
- Local indices are extended to index halo vertices as well, e.g. with respect to processor P_1:





3
Parallel remeshing

State of the art (1/2)

- First class of parallel remeshing techniques:
  - Parallelization of existing sequential remeshing techniques
    - introduced in 1996 [Castanos and Savage, 1996] for 2D meshes
    - 3D remeshing in 2000 for homogeneous meshes [Oliker et al., 2000]
    - Delaunay triangulation in 2003 [Chrisochoides and Nave, 2003]
    - 3D remeshing for mixed meshes [Lawlor et al., 2006]
  - Problems:
    - Difficulties to parallelize each operator of the remesher
    - Remeshing some element requires neighborhood information
    - Too much communication between subdomains is required to achieve quality as high as in sequential processing

$\rightarrow$ This class of parallel remeshing methods cannot handle large-size meshes distributed across a large number of processors

State of the art (2/2)

- Second class of parallel remeshing:
  - Re-use sequential remeshers in a parallel framework:
    - introduced in 2000 [Coupez et al., 2000, Dobrzynski and Remacle, 2007]
    - 3D remeshing for mixed meshes [Cavallo et al., 2005]
    - remeshing with hierarchical transport [Digonnet et al., 2007]
    - multi-grid remeshing [Ramadan et al., 2009]

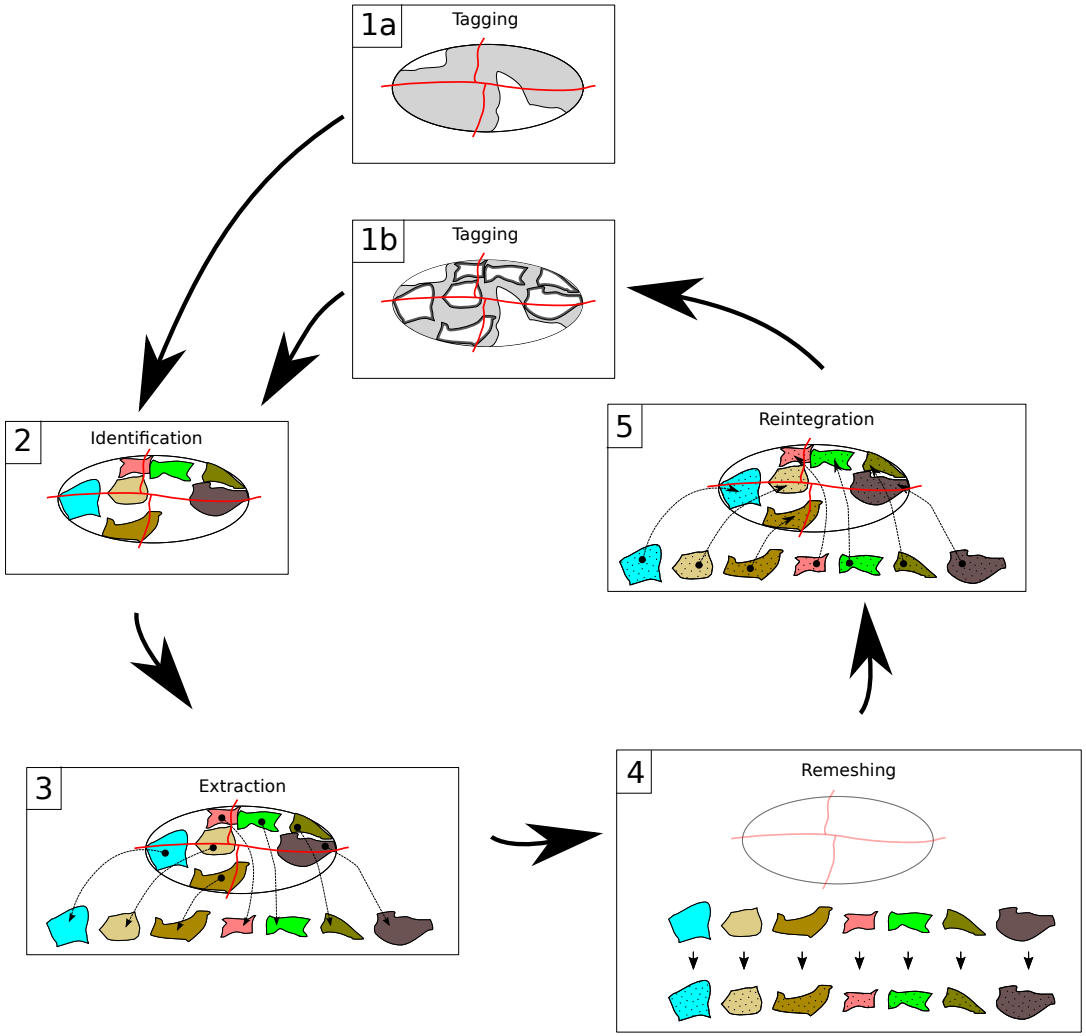
  $\rightarrow$ Our approach is to generalize this class of parallel remeshing techniques so as to allow for plugging-in any sequential remesher

Requirements (1/2)

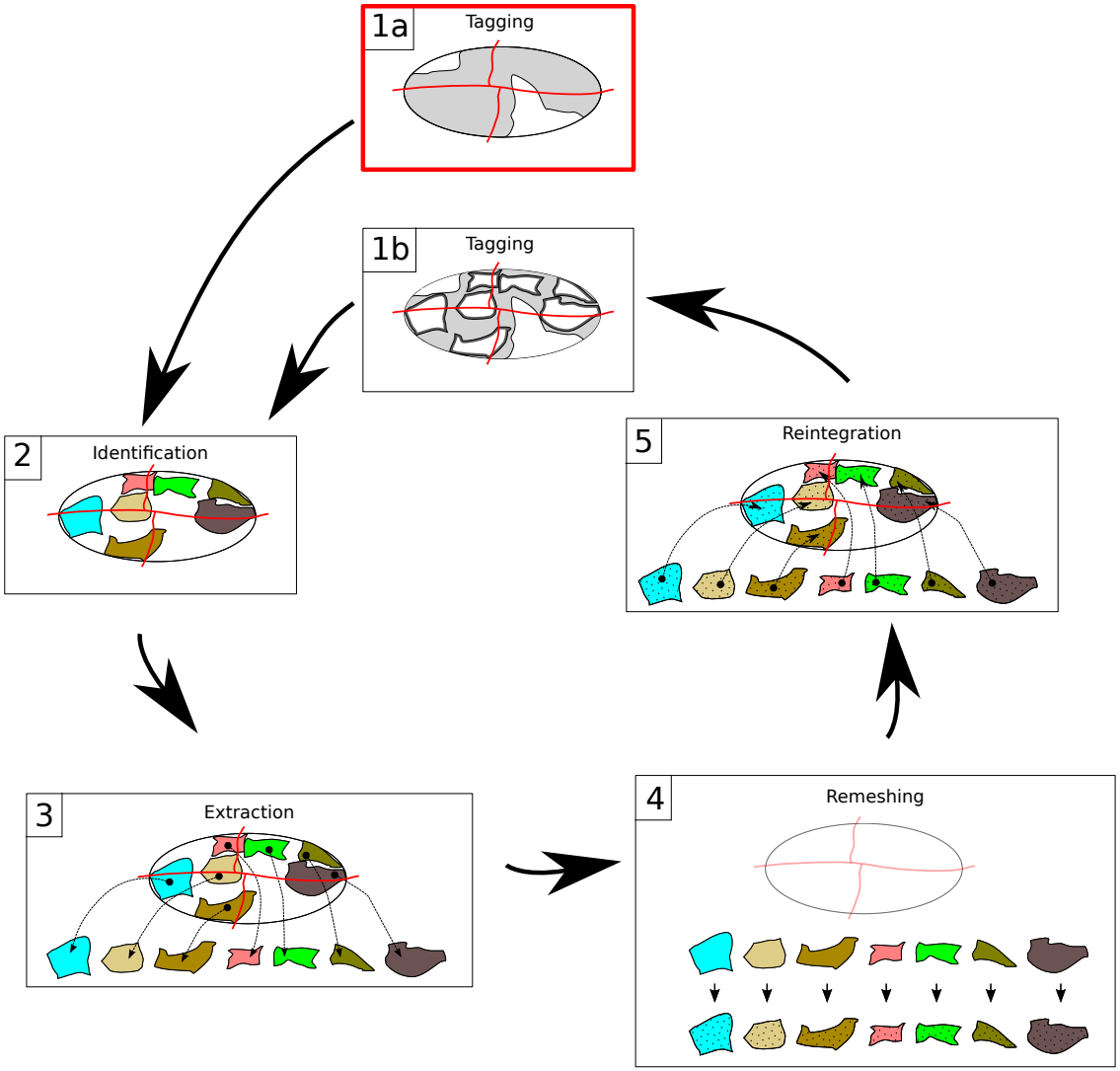
- How to remesh in parallel on distributed meshes?
  - Subdomain size could be greater than mesh size allowed by the sequential remesher
    $\rightarrow$ Need to identify in parallel non-overlapping zones of prescribed size / workload
  - Zone frontiers must be left unmodified so as to ease reintegration of remeshed zones into the distributed mesh

Global scheme:

Iterative process until all tagged elements are remeshed

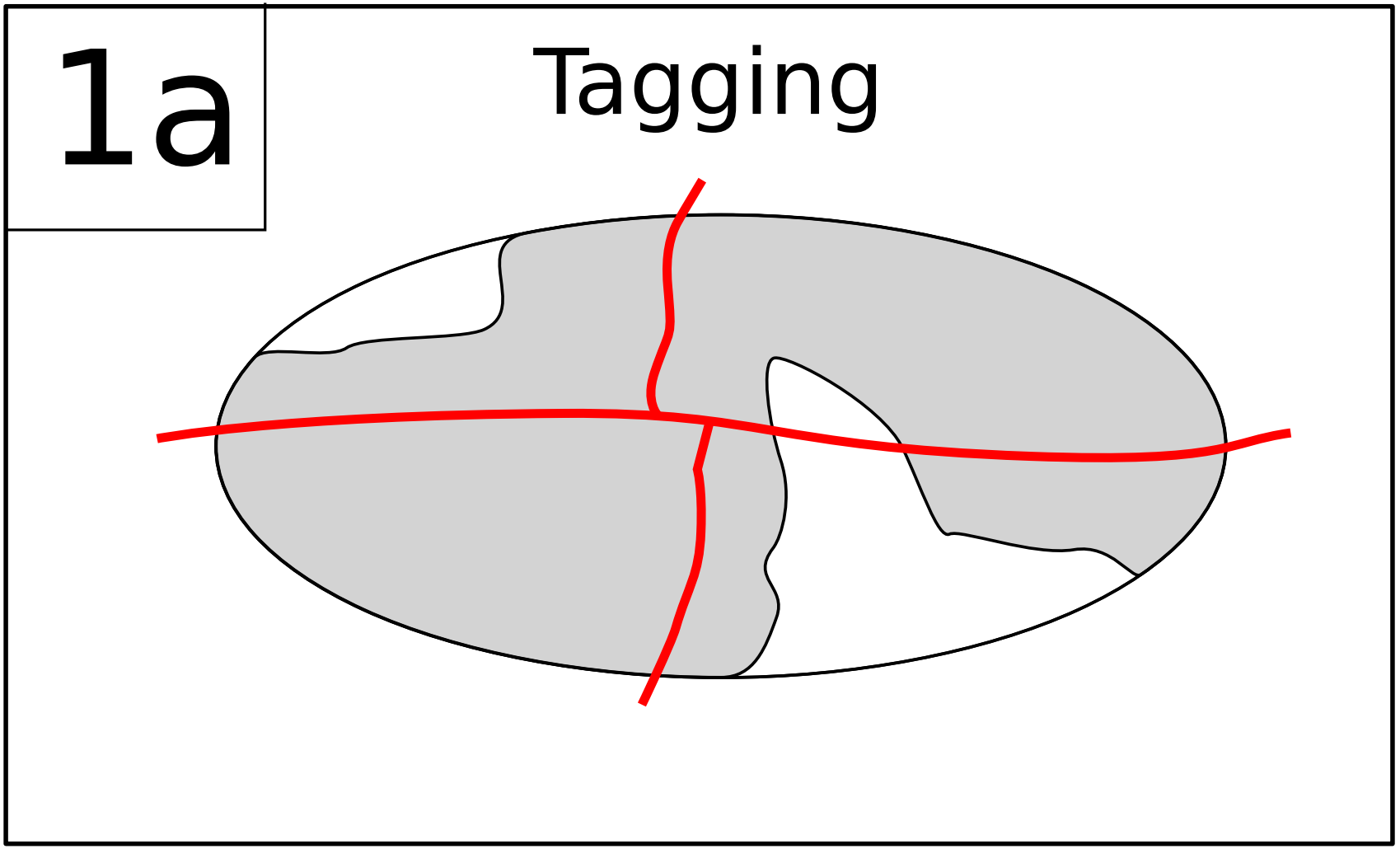


Global scheme:

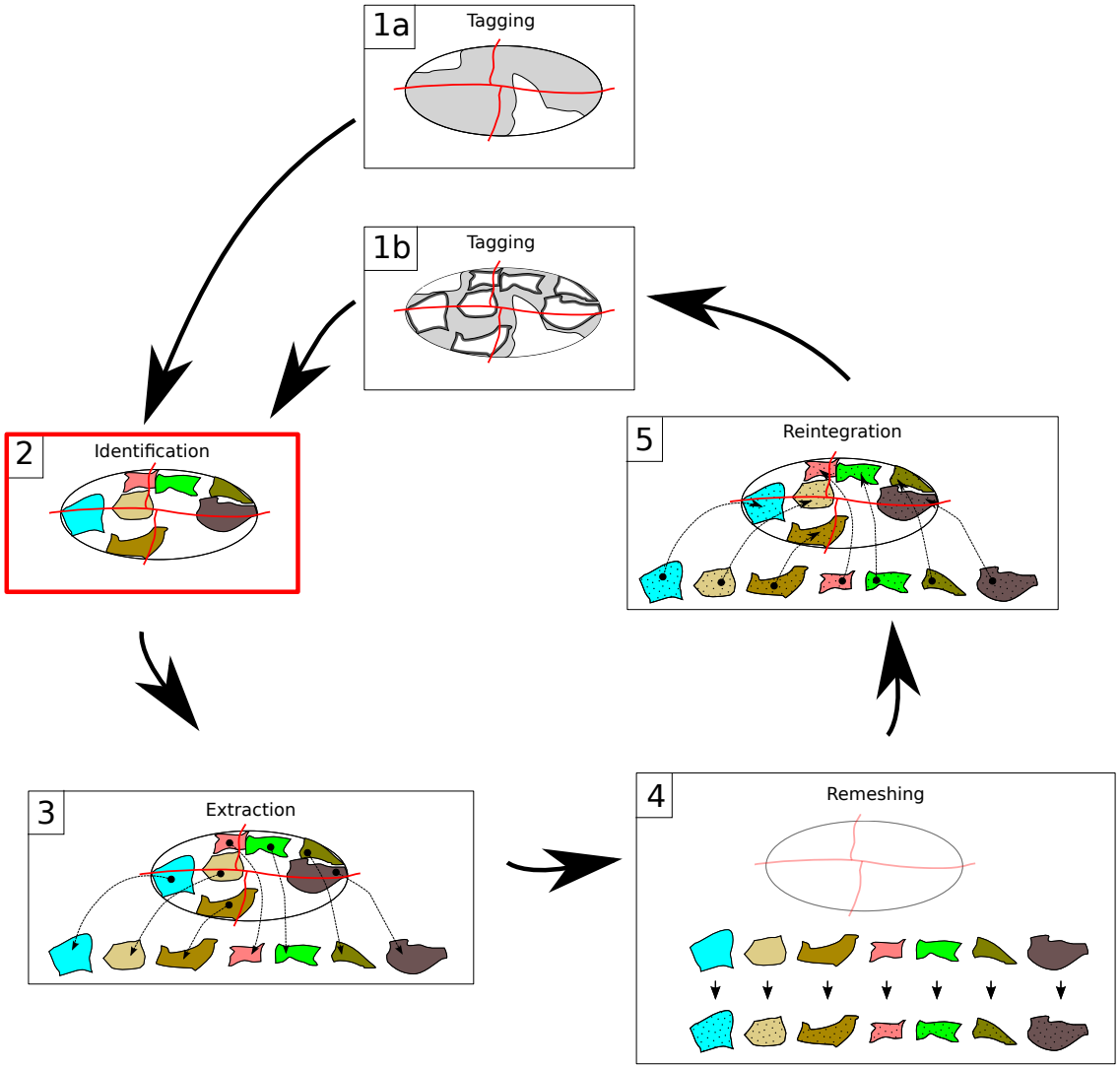


Global scheme:

Tag elements which need to be remeshed

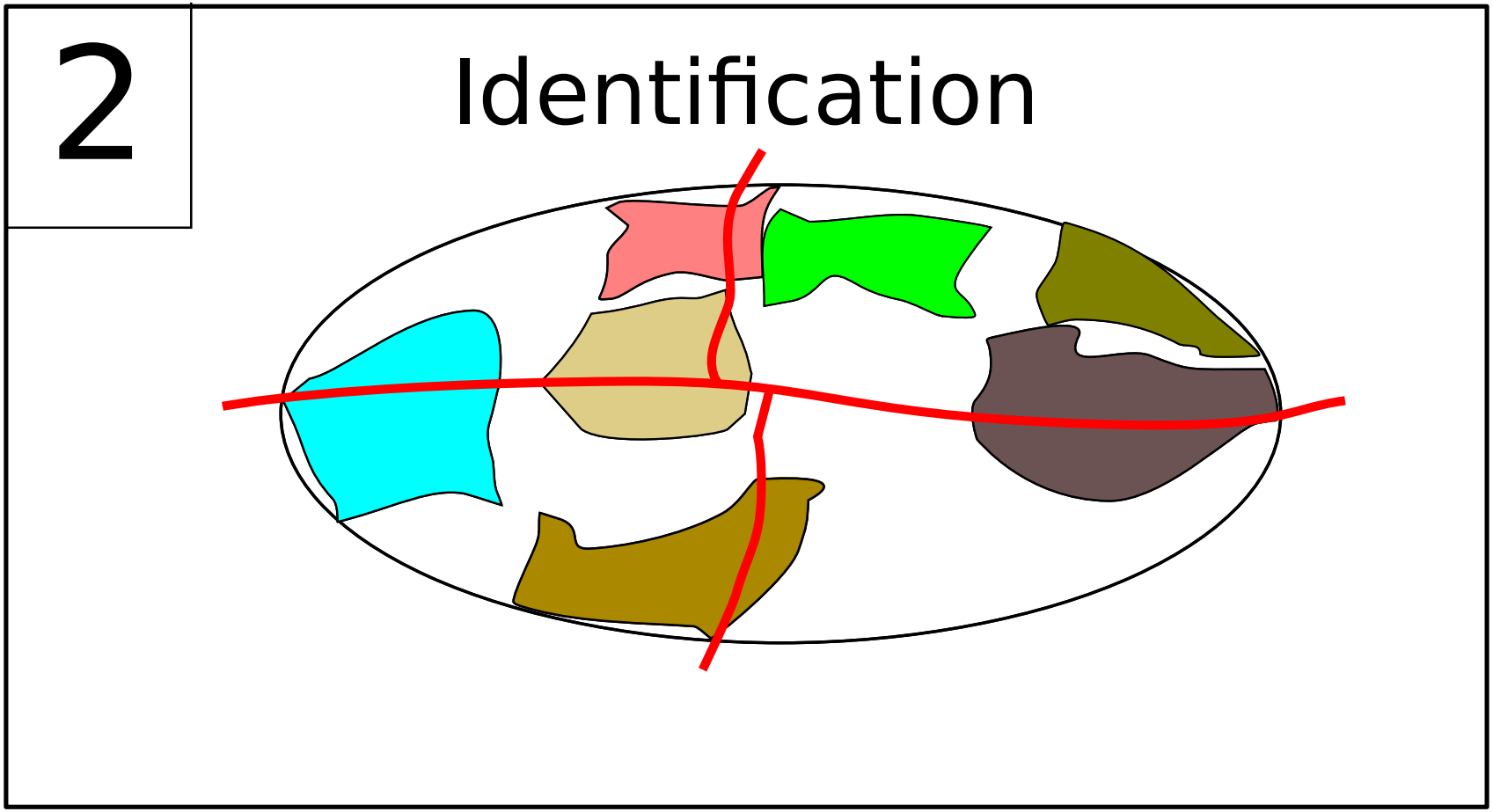


Global scheme:

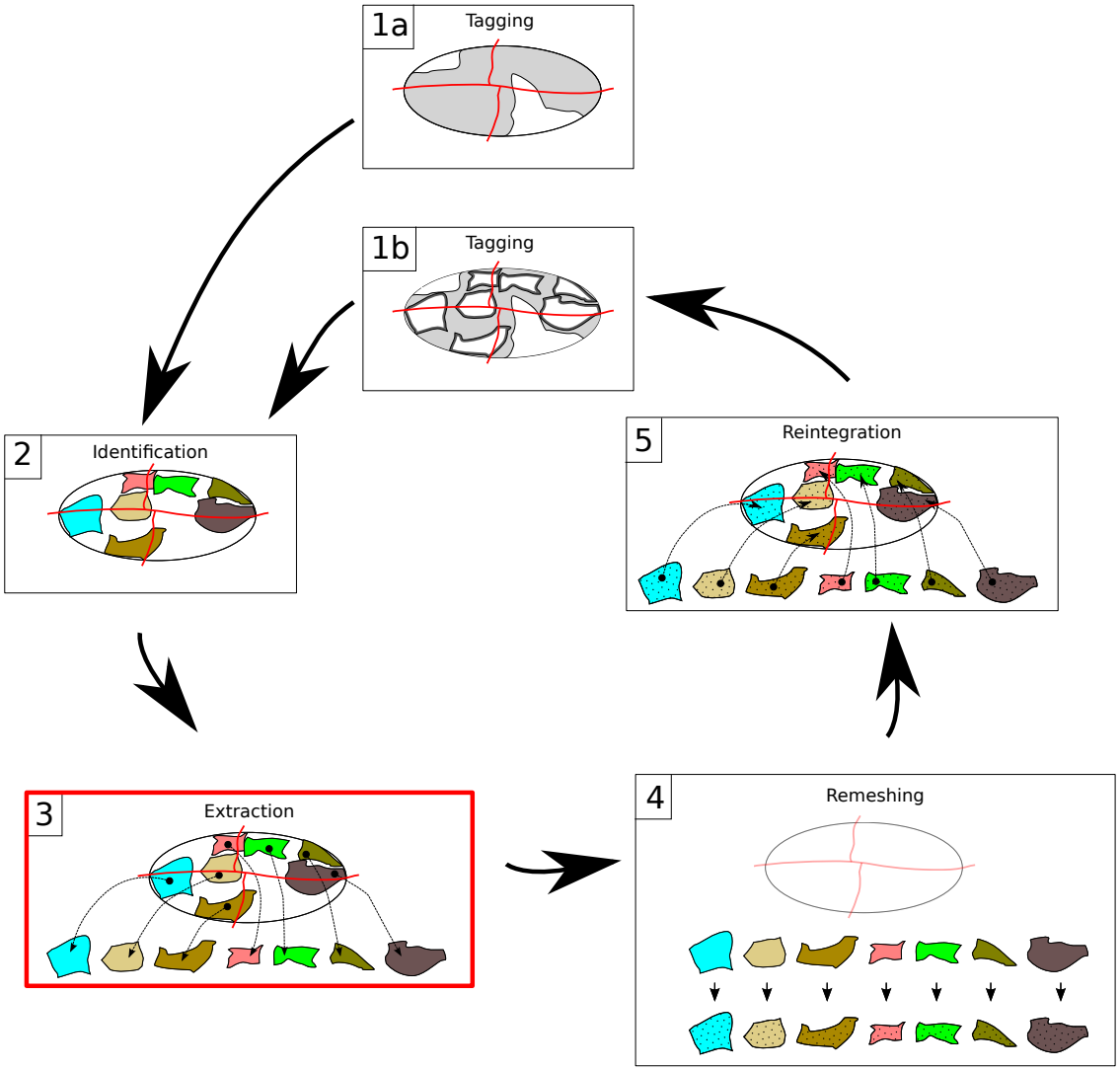


Global scheme:

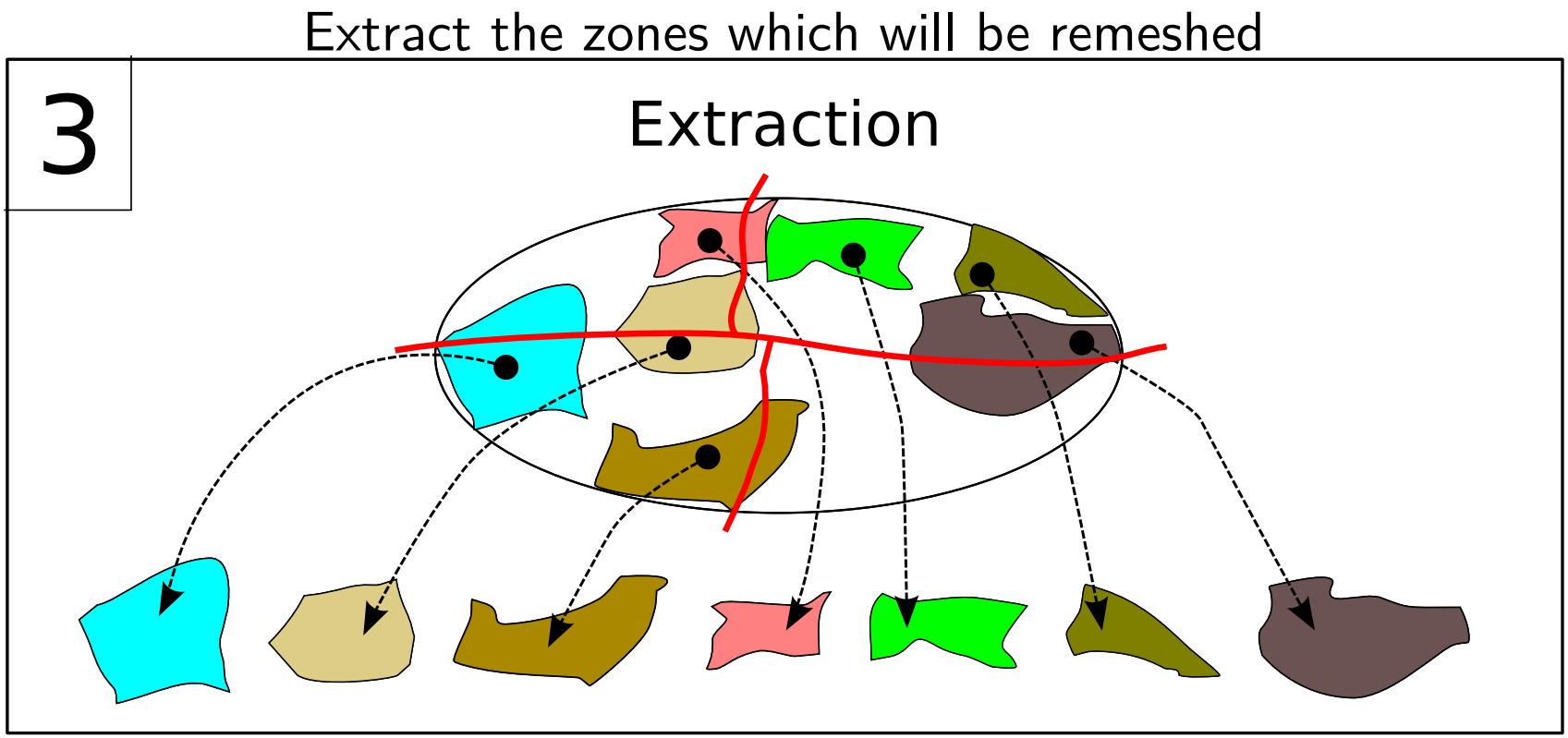
Identify non overlapping zones in parallel



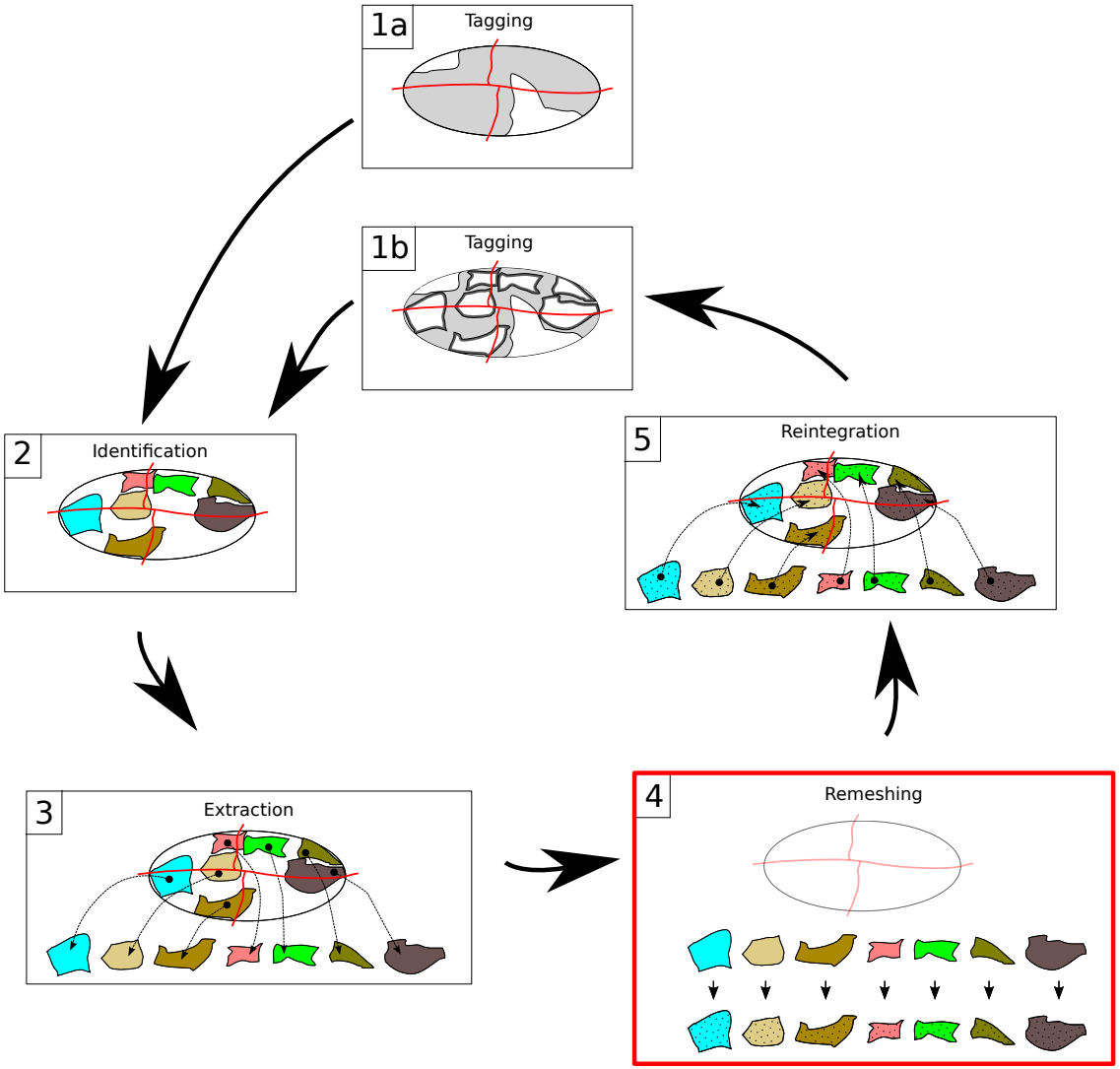
Global scheme:



Global scheme:

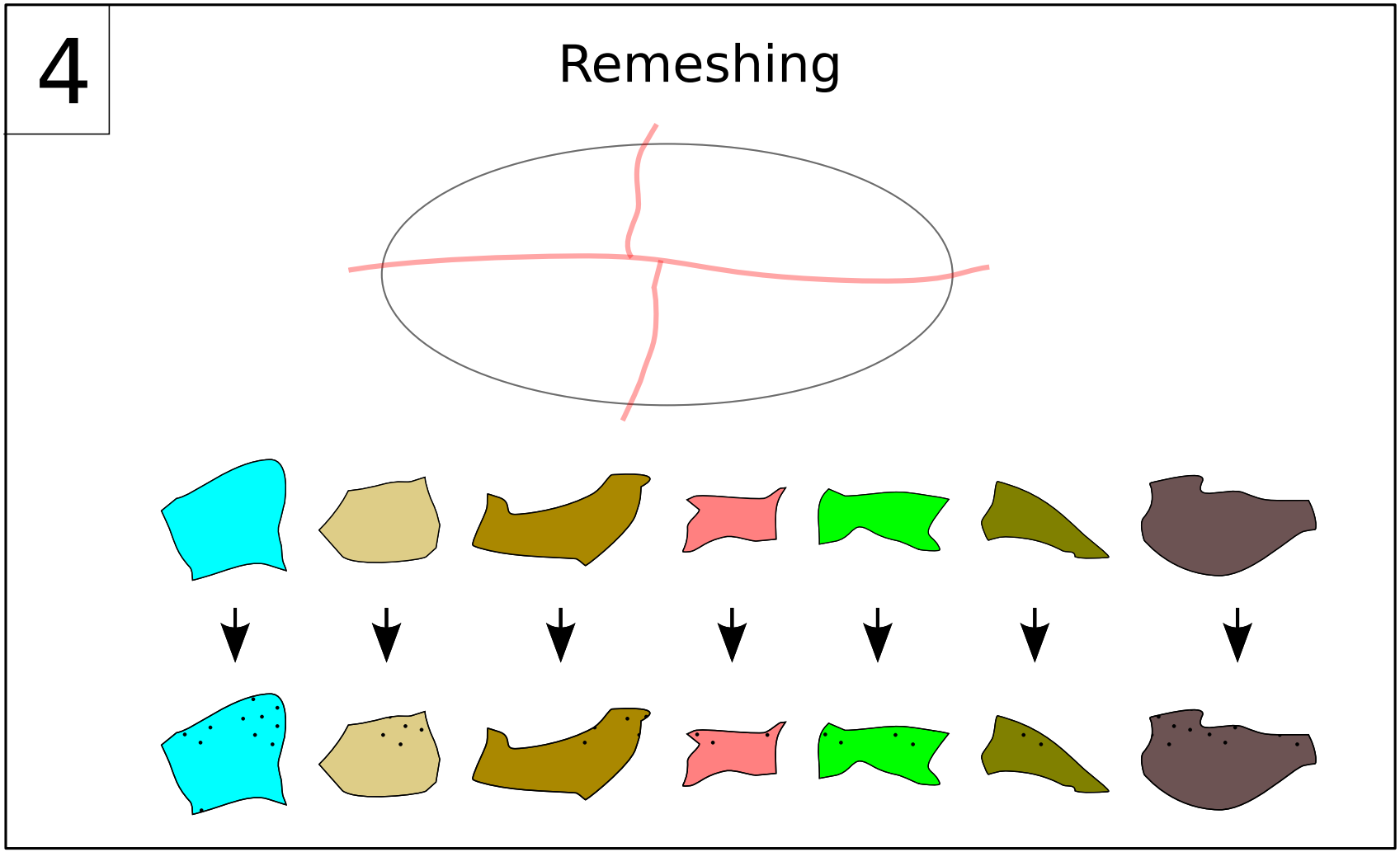


Global scheme:

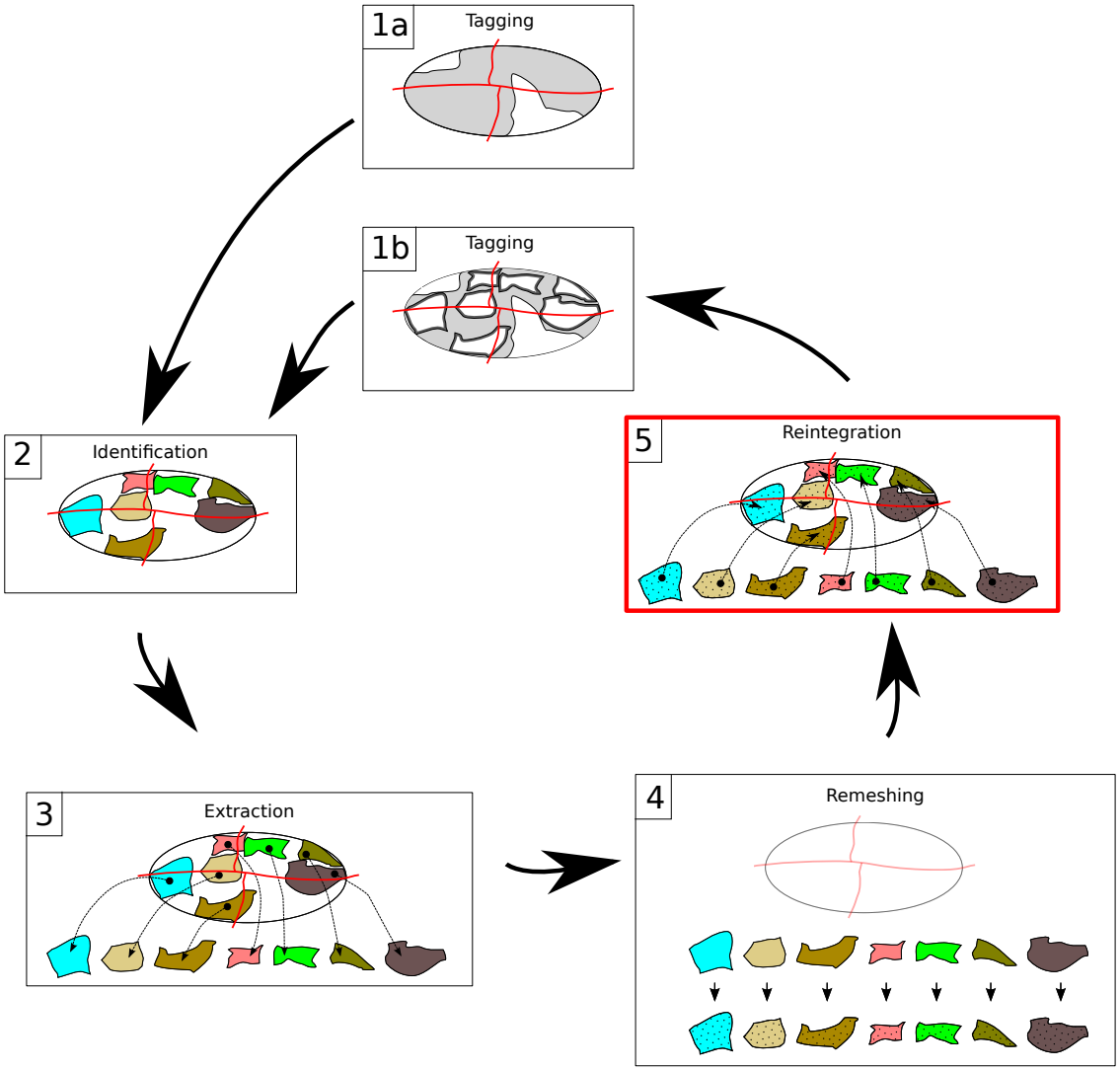


Global scheme:

Remesh in sequential each zone

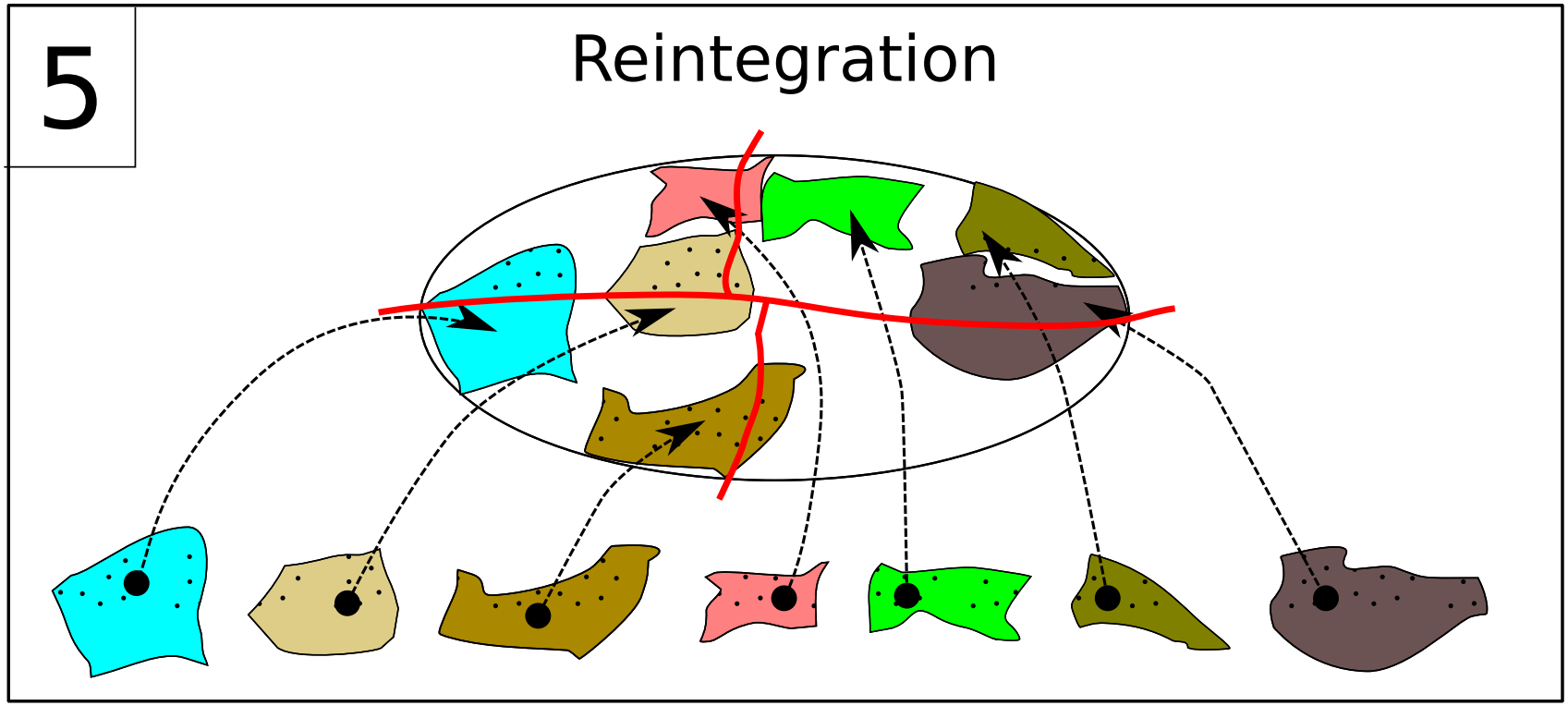


Global scheme:

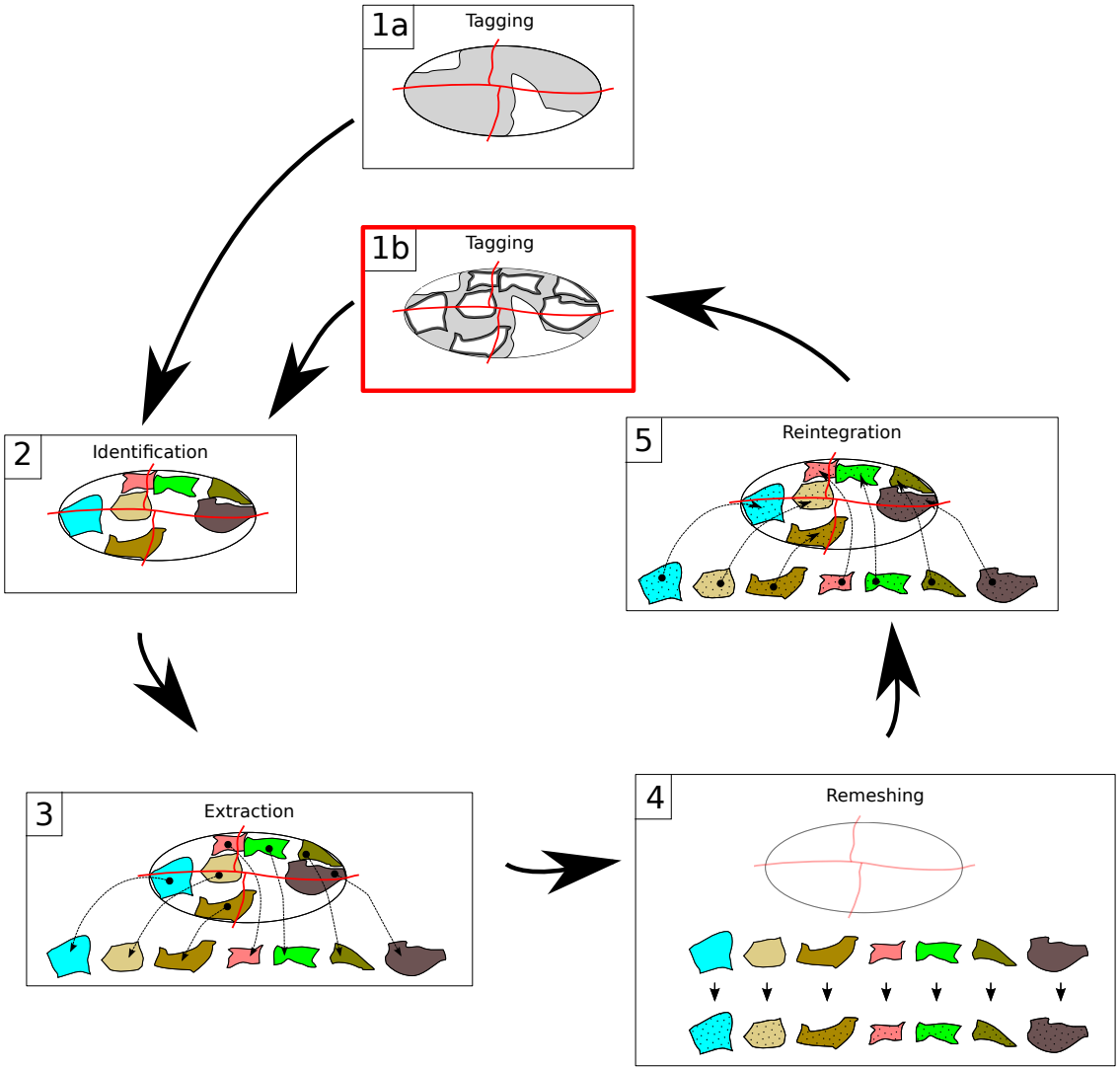


Global scheme:

Reintegrate the remeshed zones in the distributed mesh

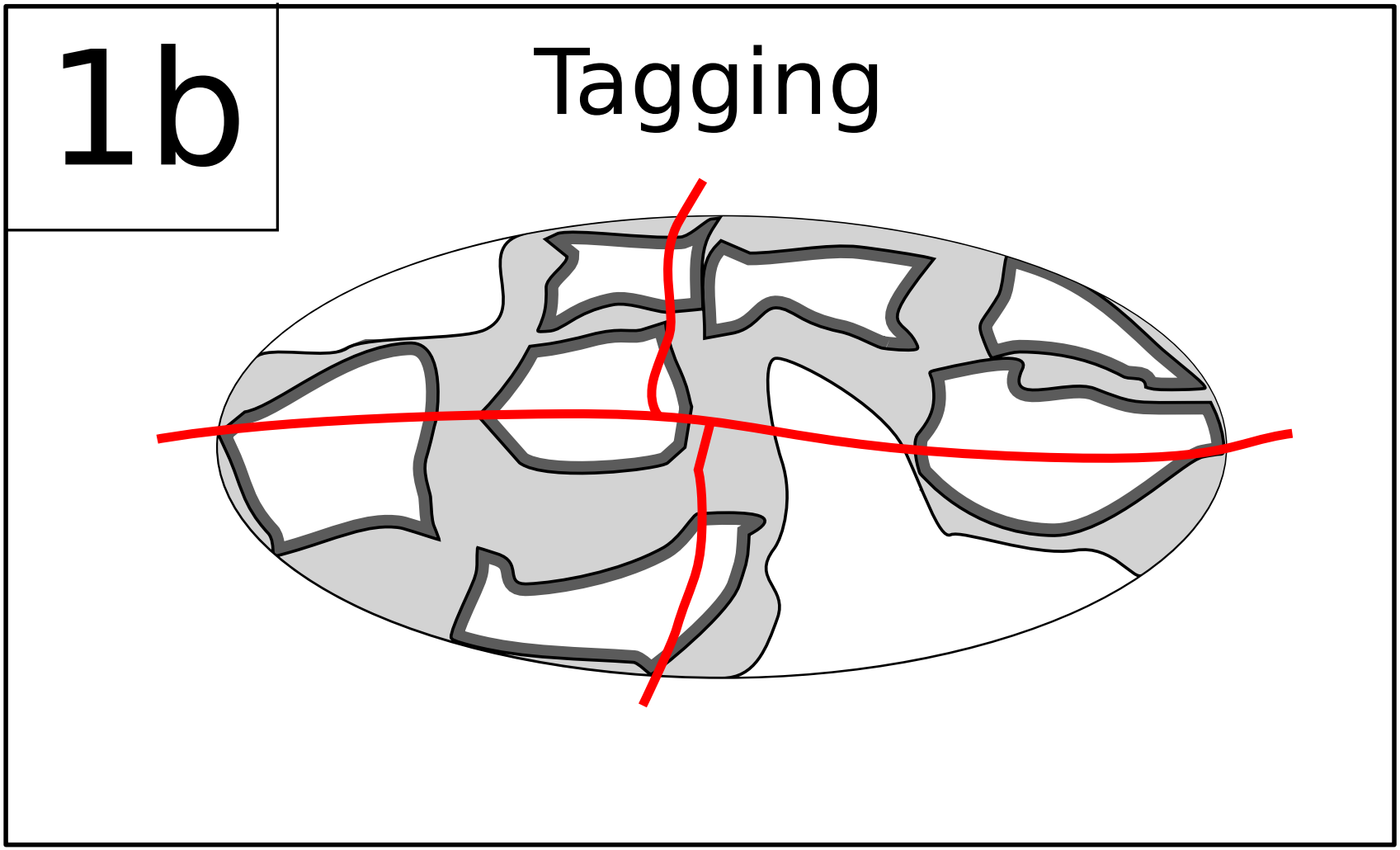


Global scheme:

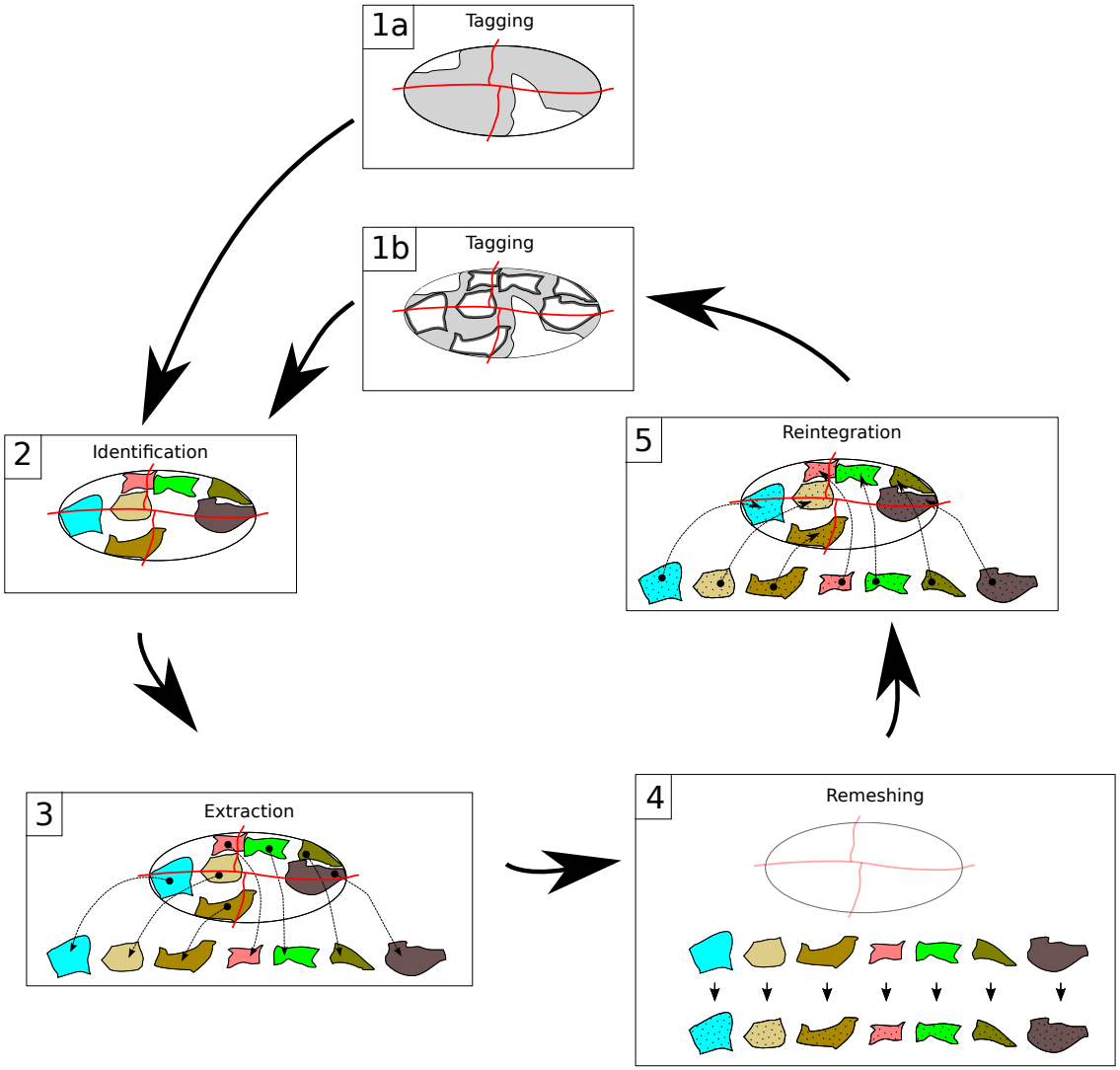


Global scheme:

Tag again elements which need to be remeshed



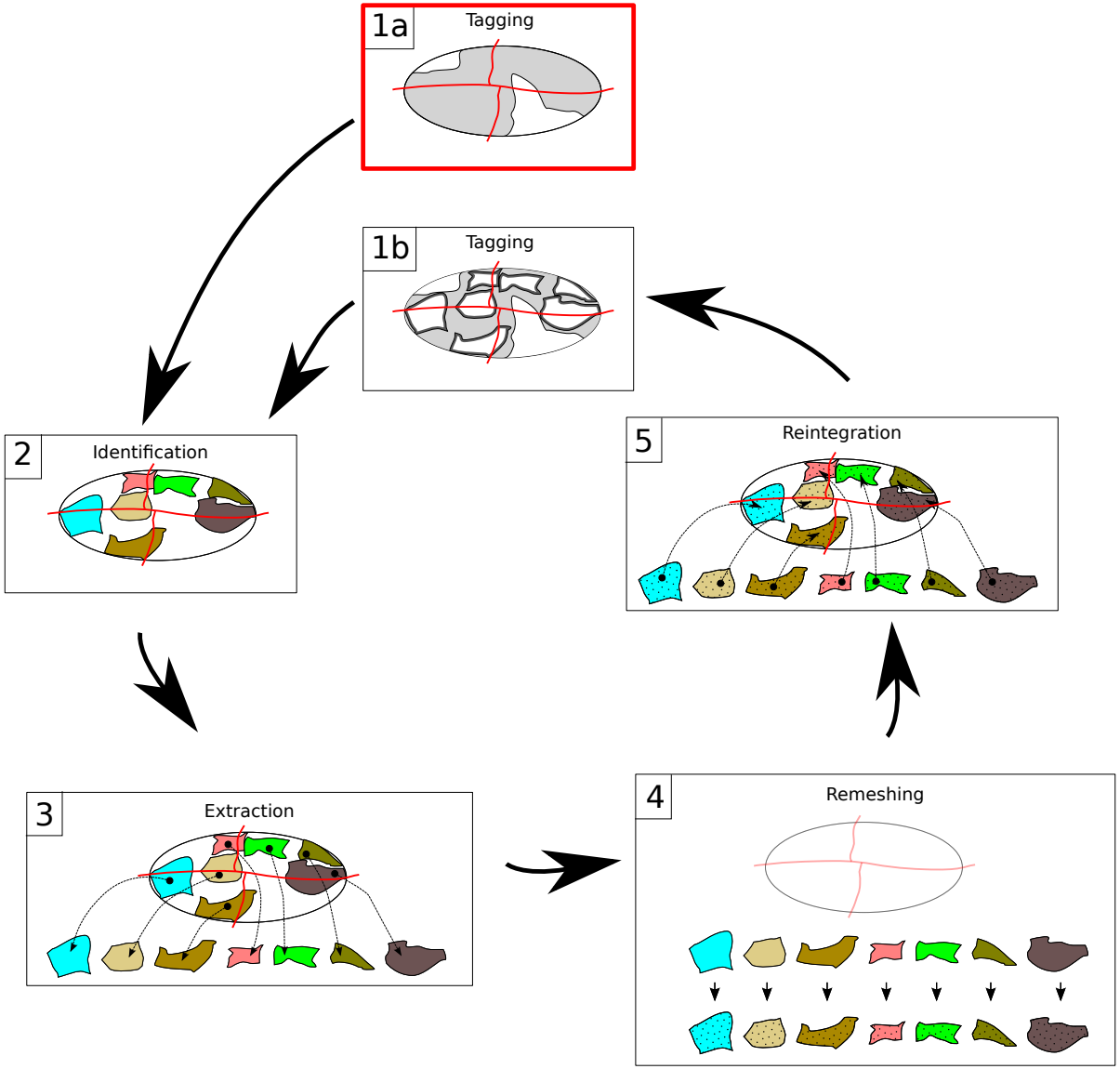
Global scheme:



Requirements (2/2)

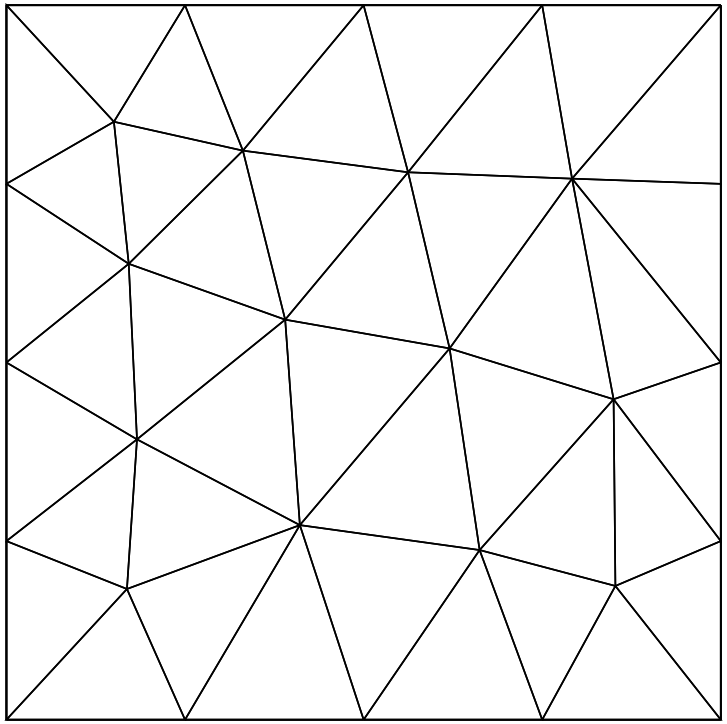
- How to determine which elements need to be remeshed?
  - Isotropic or anisotropic metric on vertices
  - Avoid elements of bad quality
- Which criteria on identified zones?
  - Measure zone skin size compared to zone volume:
    - compute the isoperimetric quotient of the element
    - value between 0 and 1
    - 0: worst quotient possible
    - 1: optimal quotient, in the case of sphere and the corresponding ball

Tagging (1/3)

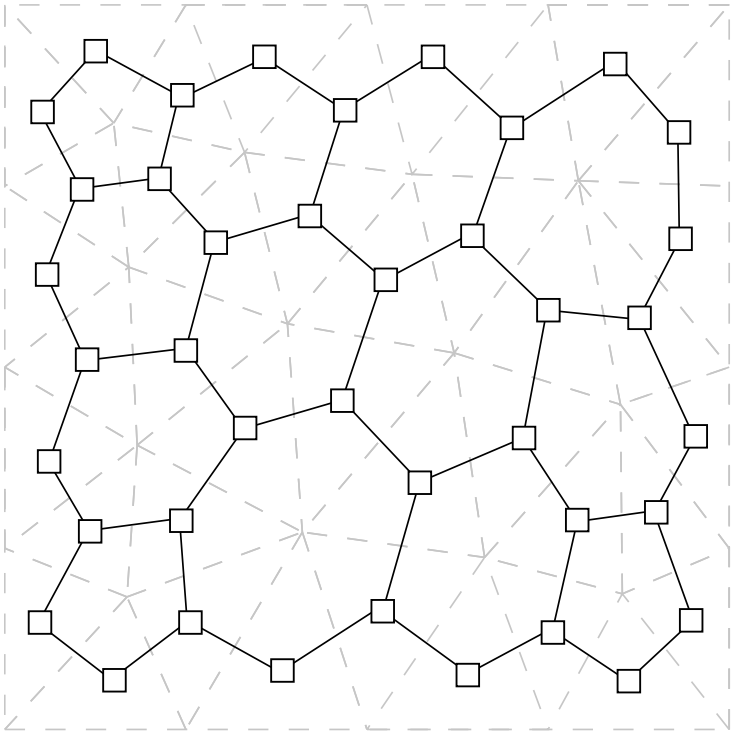

1a
Tagging
1b
Tagging
2
Identification
5
Reintegration
3
Extraction
4
Remeshing

Tagging (2/3)

- User-provided mesh:

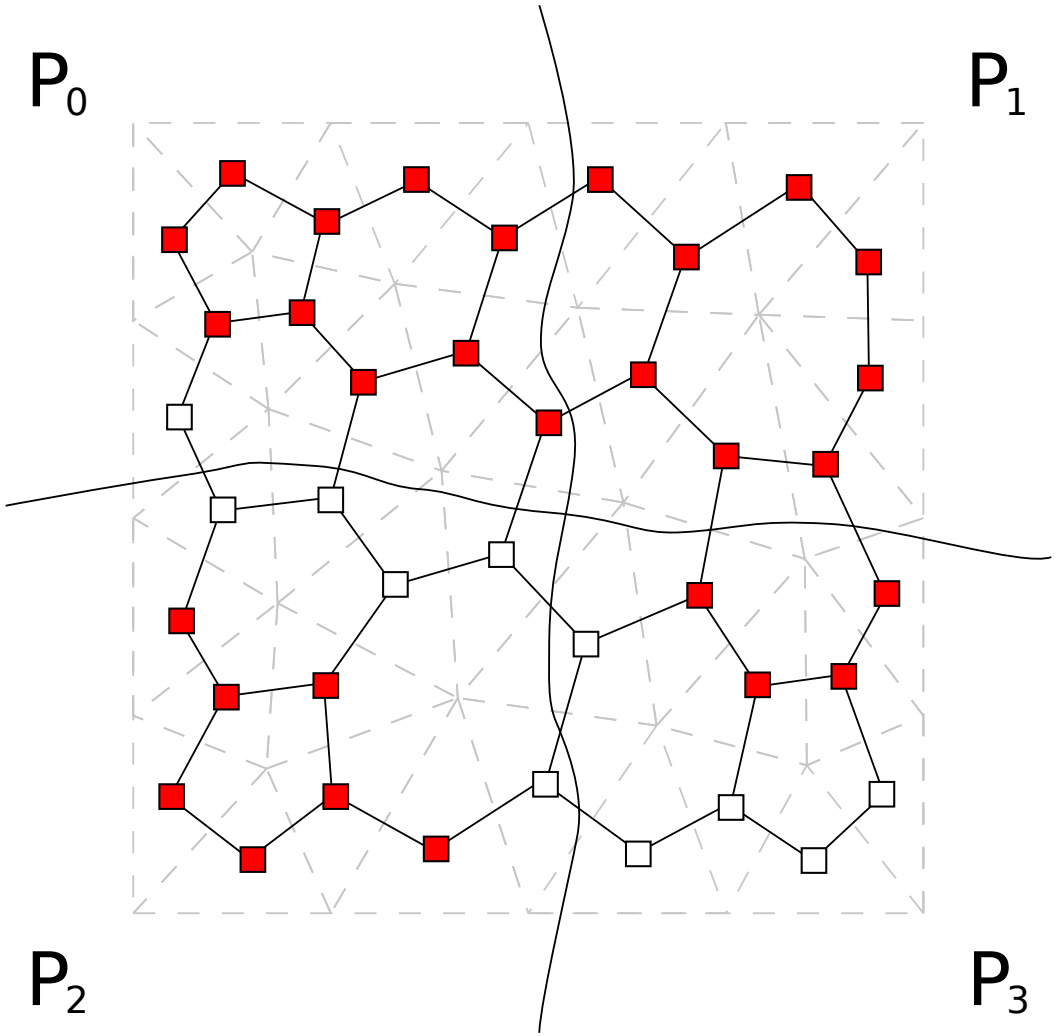


- Corresponding element graph:

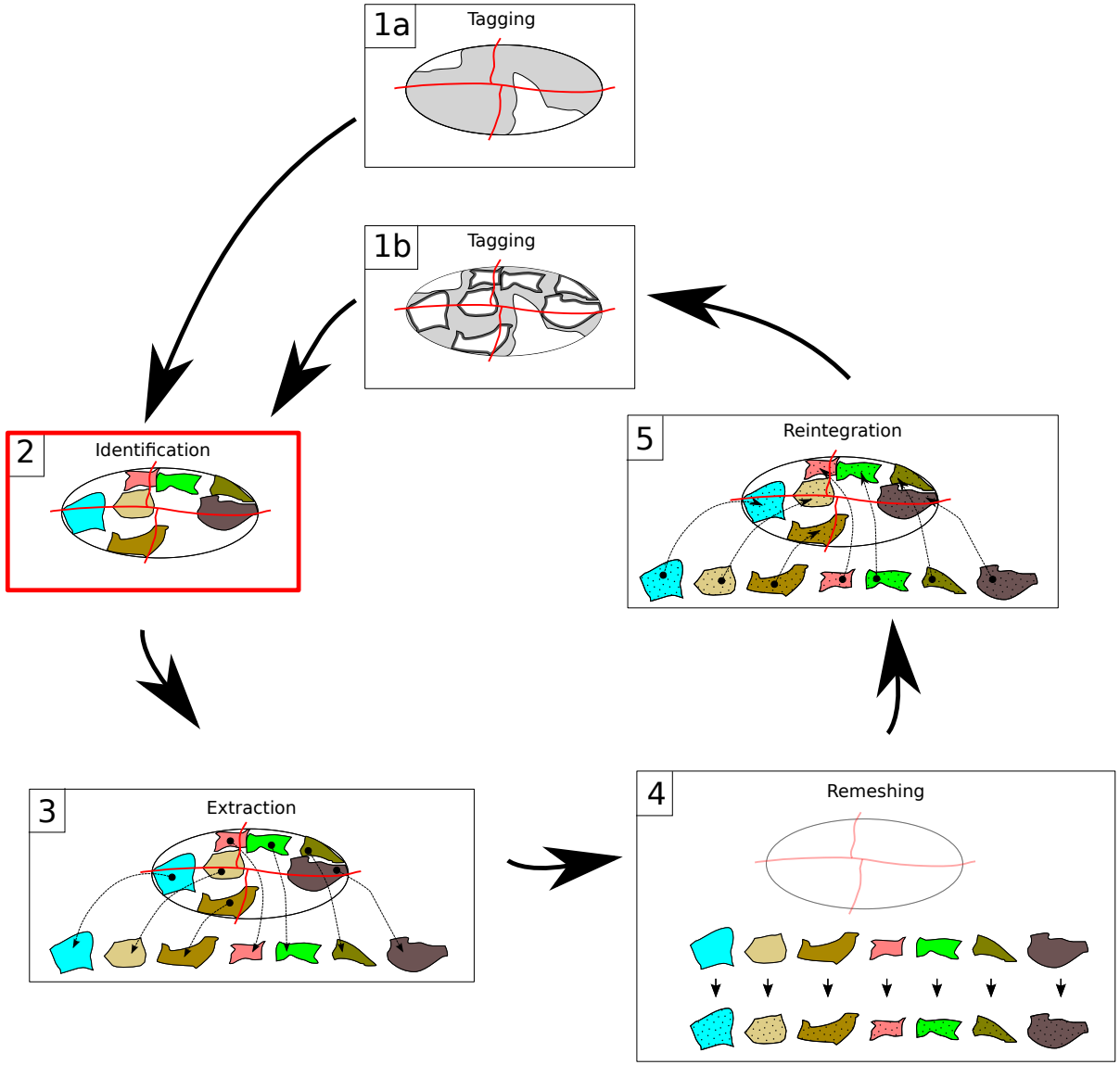


Tagging (3/3)

- Tag elements which need to be remeshed:

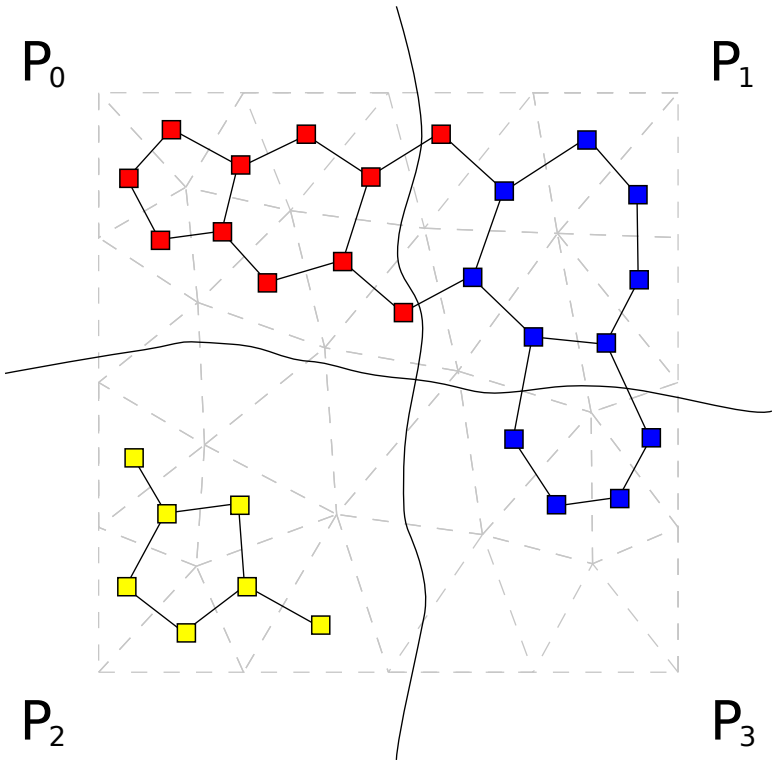


Identification (1/9)



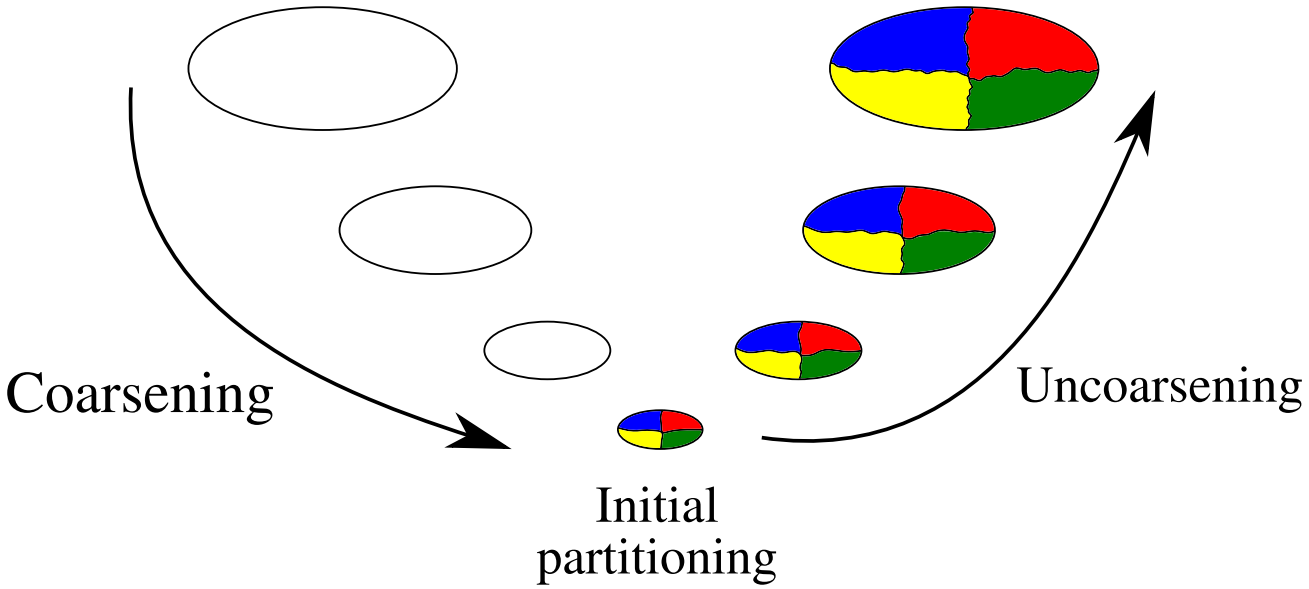
Identification (2/9)

- Identify non-overlapping zones in parallel:
  - each zone will be given to a sequential remesher
  - a weight is determined on each element according to the dedicated work of the remesher
    $\rightarrow$ Want of a good evaluation of the remesher work



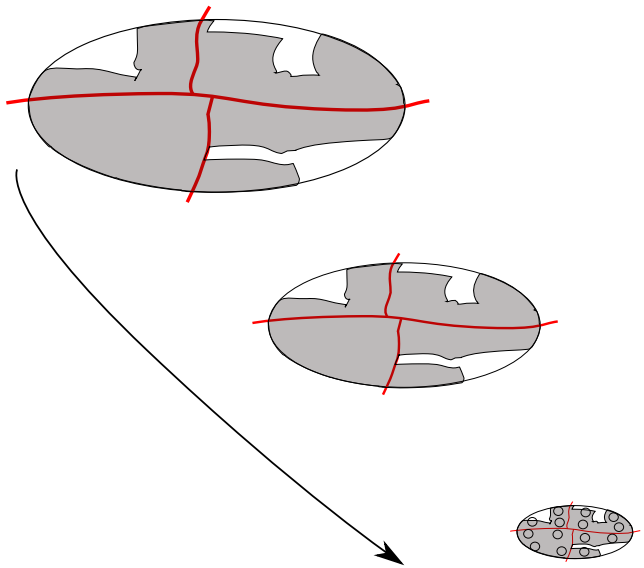
Identification (3/9)

- How to compute efficiently non-overlapping zones in parallel on a distributed mesh?
  - By using a graph multilevel scheme which provides very good partitions when combined to local optimization algorithms



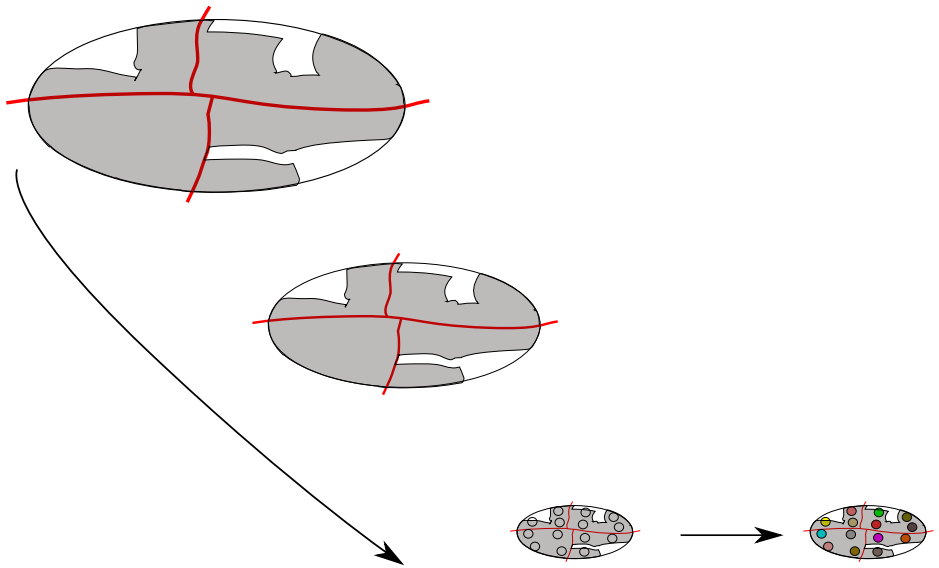
Identification (3/9)

- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



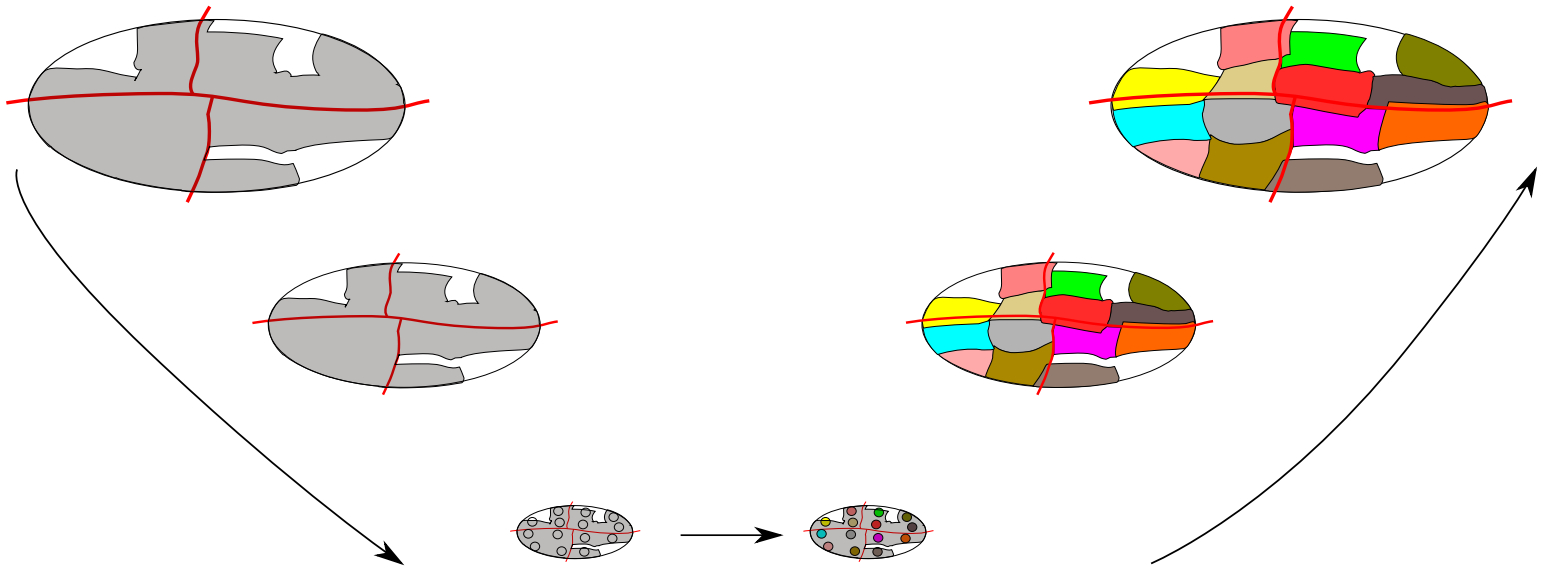
Identification (3/9)

- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



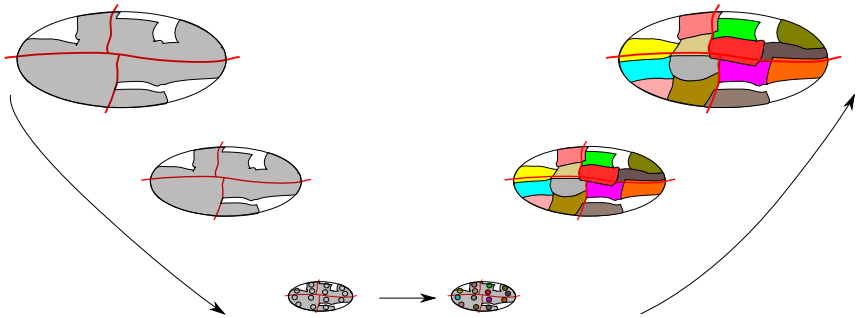
Identification (3/9)

- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



Identification (3/9)

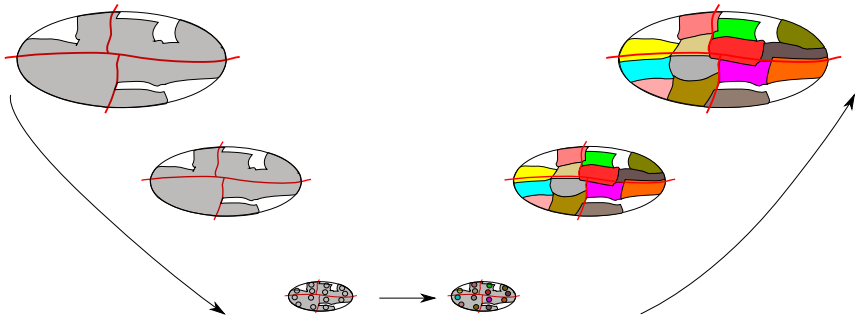
- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



- Pros:

Identification (3/9)

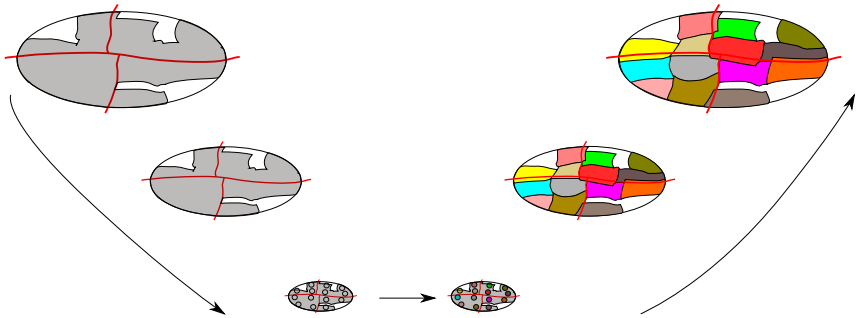
- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



- Pros:
  - Number of zones does not depend on the number of processors

Identification (3/9)

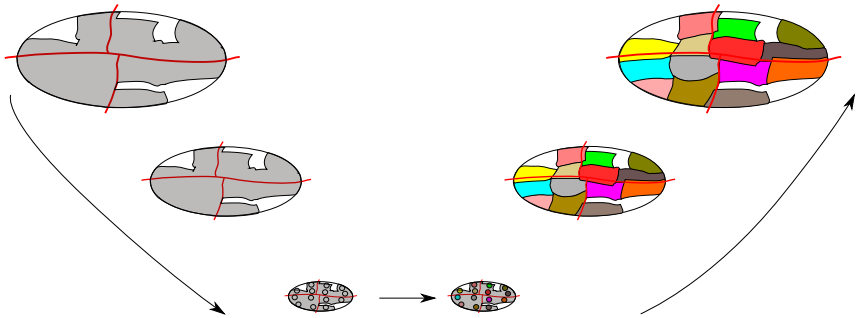
- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



- Pros:
  - Number of zones does not depend on the number of processors
- Cons:

Identification (3/9)

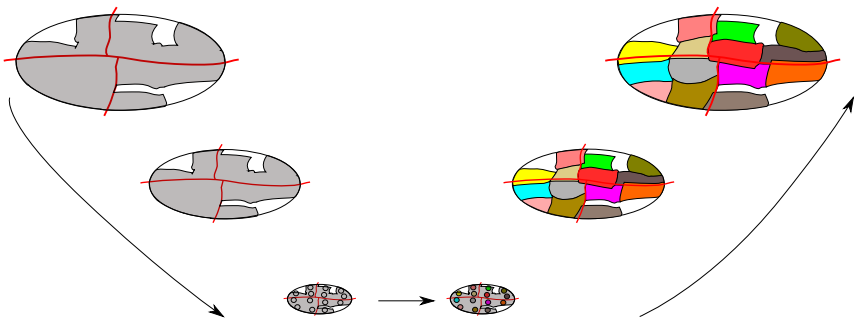
- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time



- Pros:
  - Number of zones does not depend on the number of processors
- Cons:
  - Zones are not compact and have small isoperimetric quotient

Identification (3/9)

- Algorithm 1: parallel graph coarsening
  - without local refinement during uncoarsening so as to save time

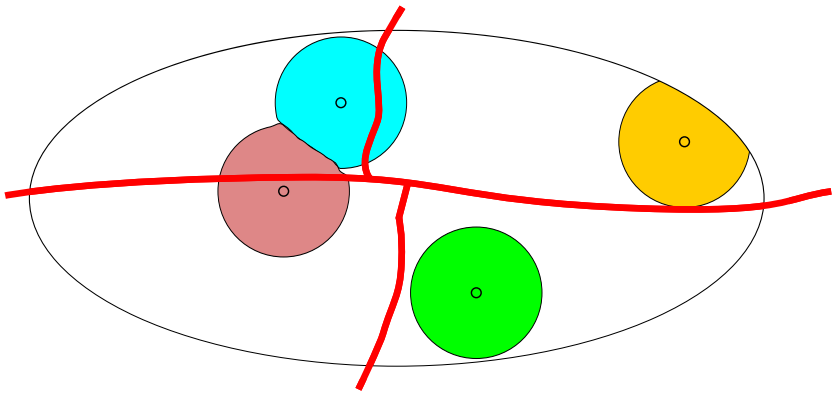


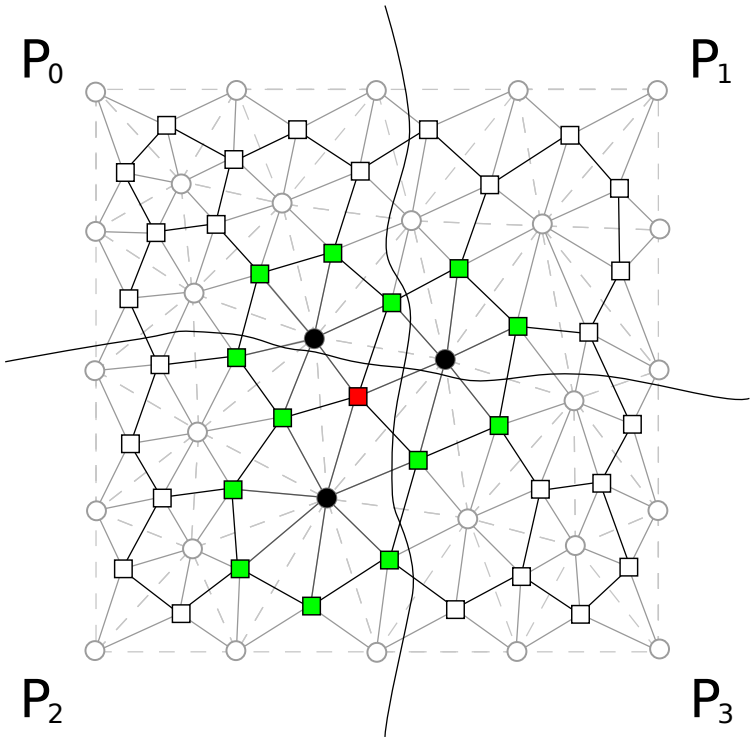
- Pros:
  - Number of zones does not depend on the number of processors
- Cons:
  - Zones are not compact and have small isoperimetric quotient

→ We want to improve isoperimetric quotient

Identification (5/9)

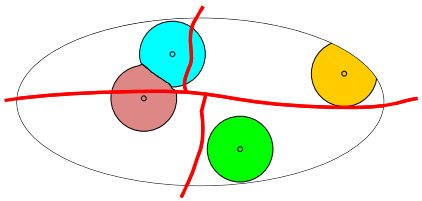
- Algorithm 2: parallel seed growing on a distributed enriched graph
  - How to choose the seeds?
    - random positions can lead to zone conflicts
    - first try: pick one seed by subdomain

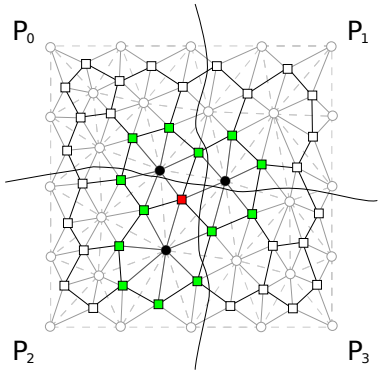




Identification (5/9)

- Algorithm 2: parallel seed growing on a distributed enriched graph
  - How to choose the seeds?
    - random positions can lead to zone conflicts
    - first try: pick one seed by subdomain

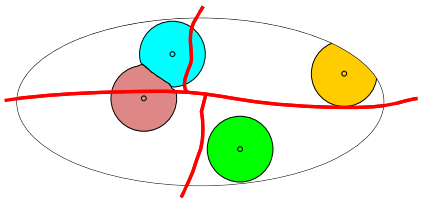


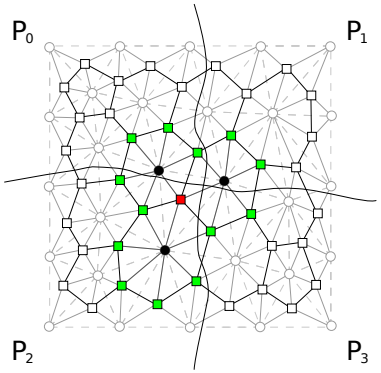


- Pros:
  - Zone isoperimetric quotient becomes reasonably good enough

Identification (5/9)

- Algorithm 2: parallel seed growing on a distributed enriched graph
  - How to choose the seeds?
    - random positions can lead to zone conflicts
    - first try: pick one seed by subdomain

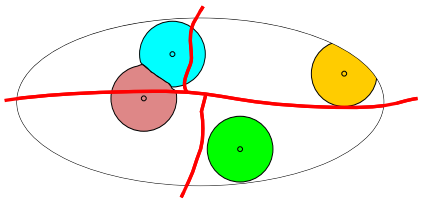


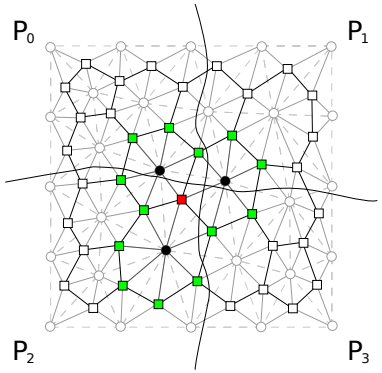


- Pros:
  - Zone isoperimetric quotient becomes reasonably good enough
- Cons:

Identification (5/9)

- Algorithm 2: parallel seed growing on a distributed enriched graph
  - How to choose the seeds?
    - random positions can lead to zone conflicts
    - first try: pick one seed by subdomain

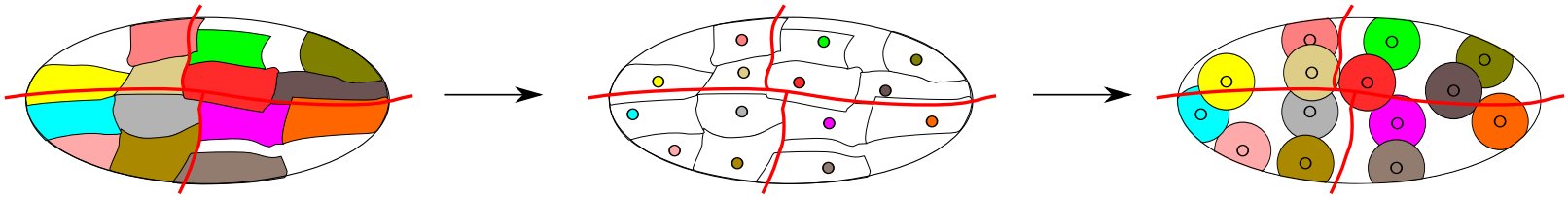




- Pros:
  - Zone isoperimetric quotient becomes reasonably good enough
- Cons:
  - Number of zones corresponds to the number of processors

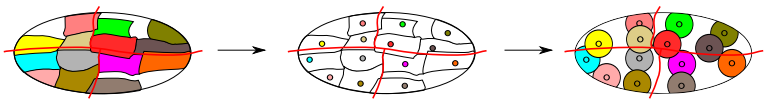
Identification (6/9)

- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



Identification (6/9)

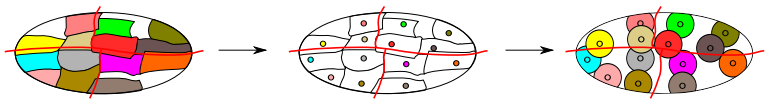
- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



- Pros:

Identification (6/9)

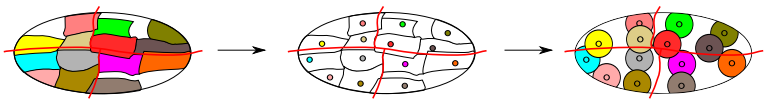
- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



- Pros:
  - Yields satisfactory isoperimetric quotients

Identification (6/9)

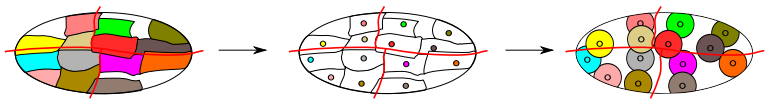
- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



- Pros:
  - Yields satisfactory isoperimetric quotients
  - Number of zones does not depend on the number of processors

Identification (6/9)

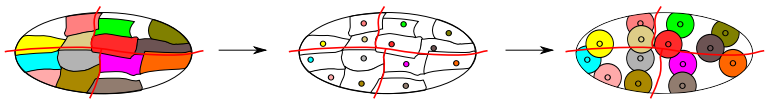
- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



- Pros:
  - Yields satisfactory isoperimetric quotients
  - Number of zones does not depend on the number of processors
- Cons:

Identification (6/9)

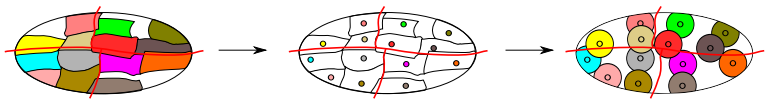
- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



- Pros:
  - Yields satisfactory isoperimetric quotients
  - Number of zones does not depend on the number of processors
- Cons:
  - Two algorithms are used on two different data structures:

Identification (6/9)

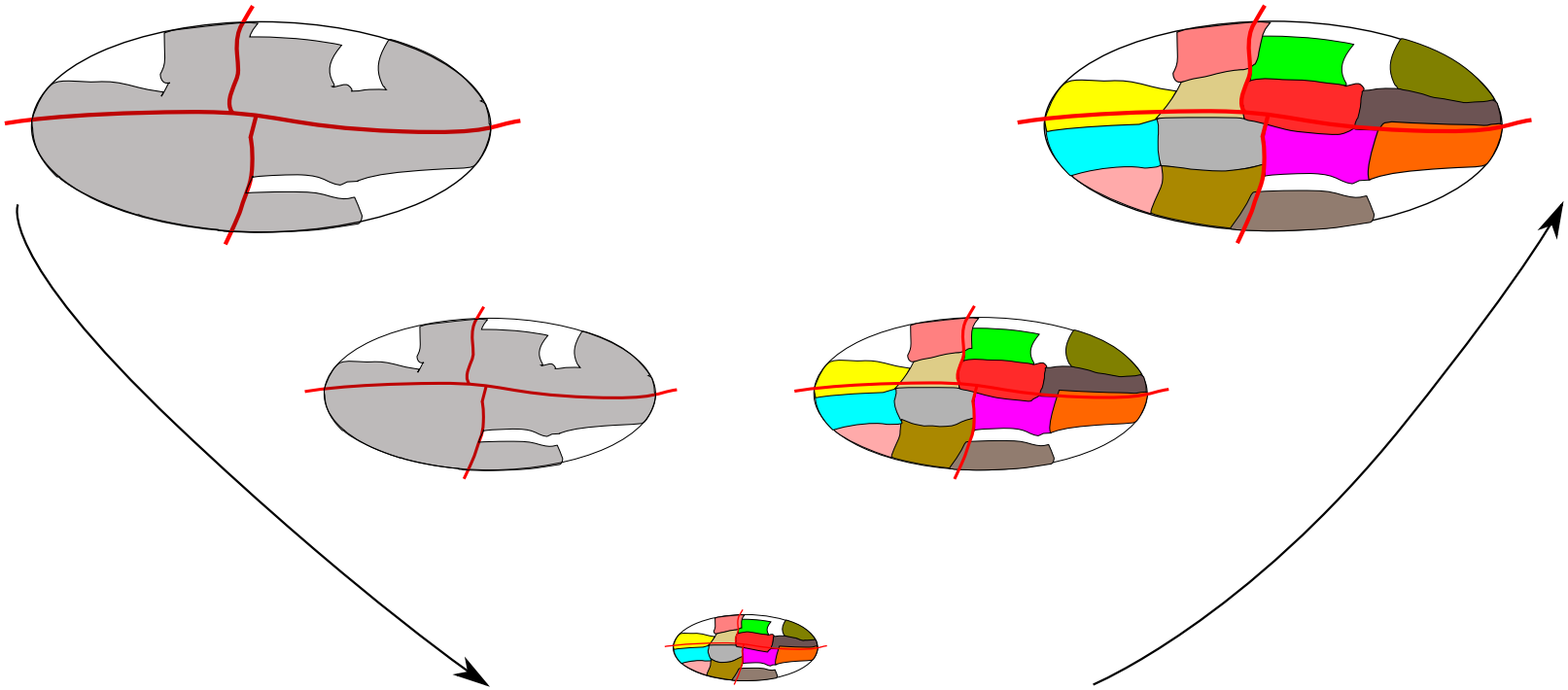
- Algorithm 3: parallel graph coarsening with parallel seed growing on a distributed enriched graph
  - Constrain seed selection within prescribed areas



- Pros:
  - Yields satisfactory isoperimetric quotients
  - Number of zones does not depend on the number of processors
- Cons:
  - Two algorithms are used on two different data structures:
    - element graph and enriched graph
    - consumes more time than other algorithms

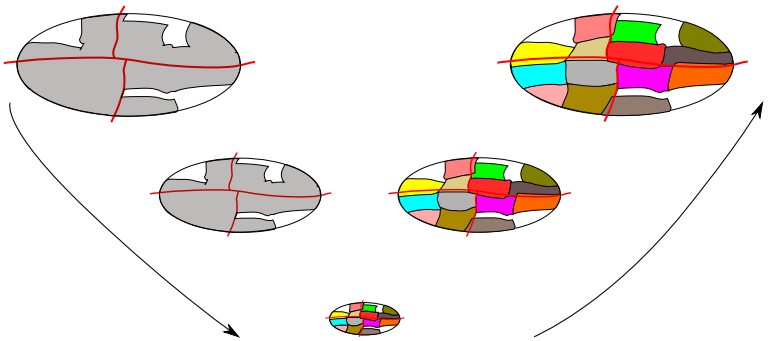
Identification (7/9)

- Algorithm 4: full-featured parallel partitioning
  - Local optimization is mandatory to obtain good isoperimetric quotients



Identification (7/9)

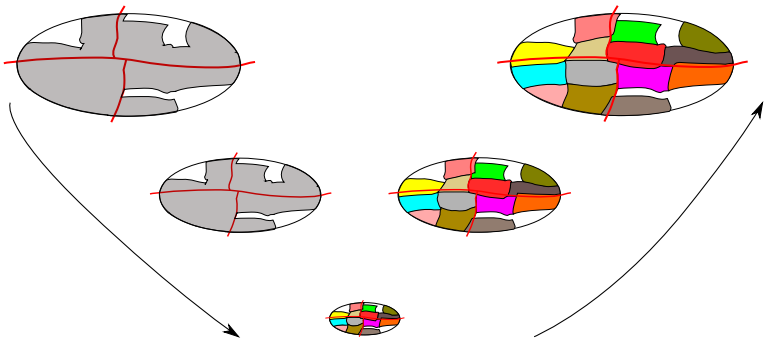
- Algorithm 4: full-featured parallel partitioning
  - Local optimization is mandatory to obtain good isoperimetric quotients



- Pros:

Identification (7/9)

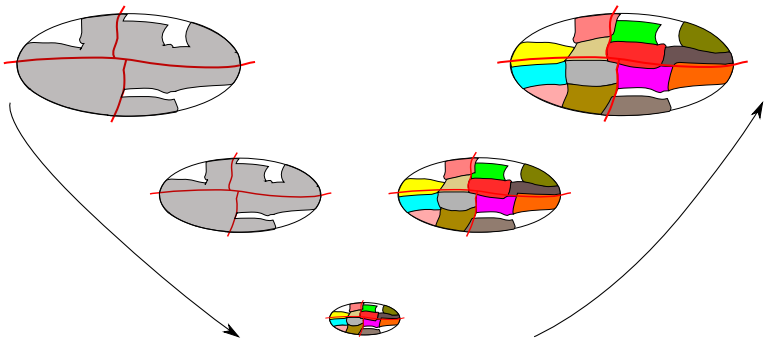
- Algorithm 4: full-featured parallel partitioning
  - Local optimization is mandatory to obtain good isoperimetric quotients



- Pros:
  - Smooth skin which gives good enough isoperimetric quotient

Identification (7/9)

- Algorithm 4: full-featured parallel partitioning
  - Local optimization is mandatory to obtain good isoperimetric quotients



- Pros:
  - Smooth skin which gives good enough isoperimetric quotient
  - Number of zones does not depend on number of processors

Identification (8/9)

Algorithm	Good IC	#Zones	Time spent
Graph coarsening	No	Yes	Yes
Seed growing	Yes	No	Yes
Graph coarsening + seed growing	Yes	Yes	No
Graph partitioning	Yes	Yes	Yes

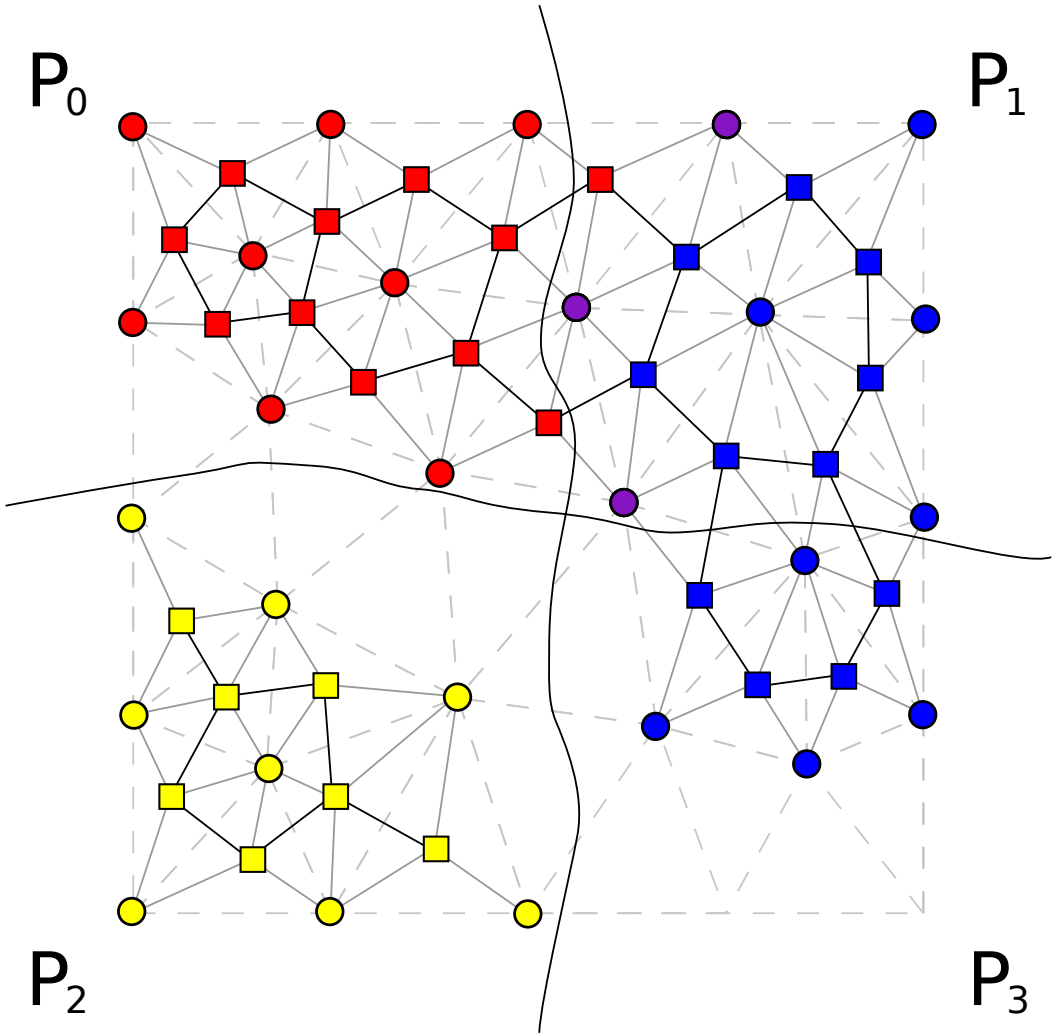
Identification (8/9)

Algorithm	Good IC	#Zones	Time spent
Graph coarsening	No	Yes	Yes
Seed growing	Yes	No	Yes
Graph coarsening + seed growing	Yes	Yes	No
Graph partitioning	Yes	Yes	Yes

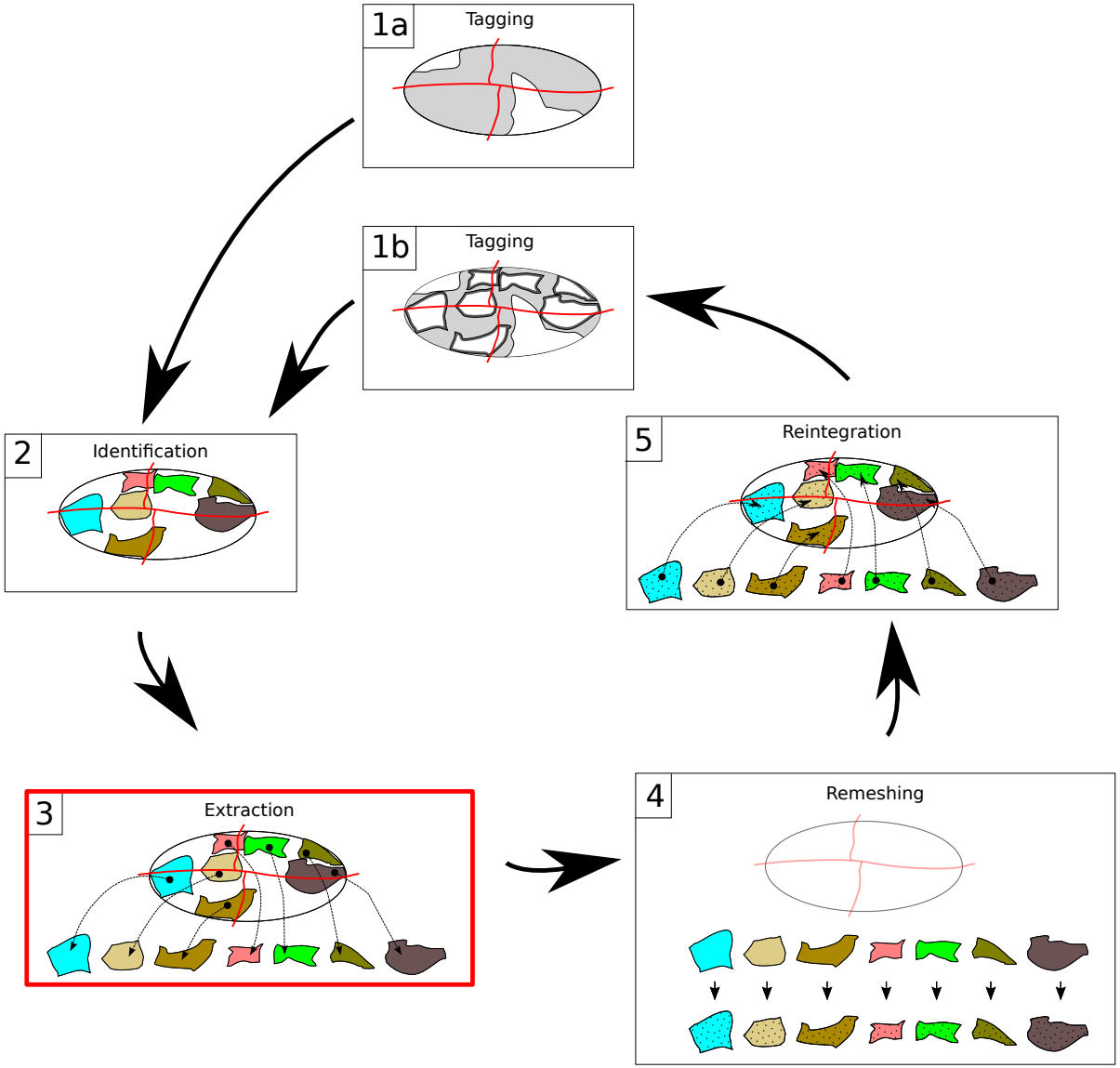
$\rightarrow$ Chosen algorithm is graph partitioning

Identification (9/9)

- Propagate zone color on other vertices through cells:

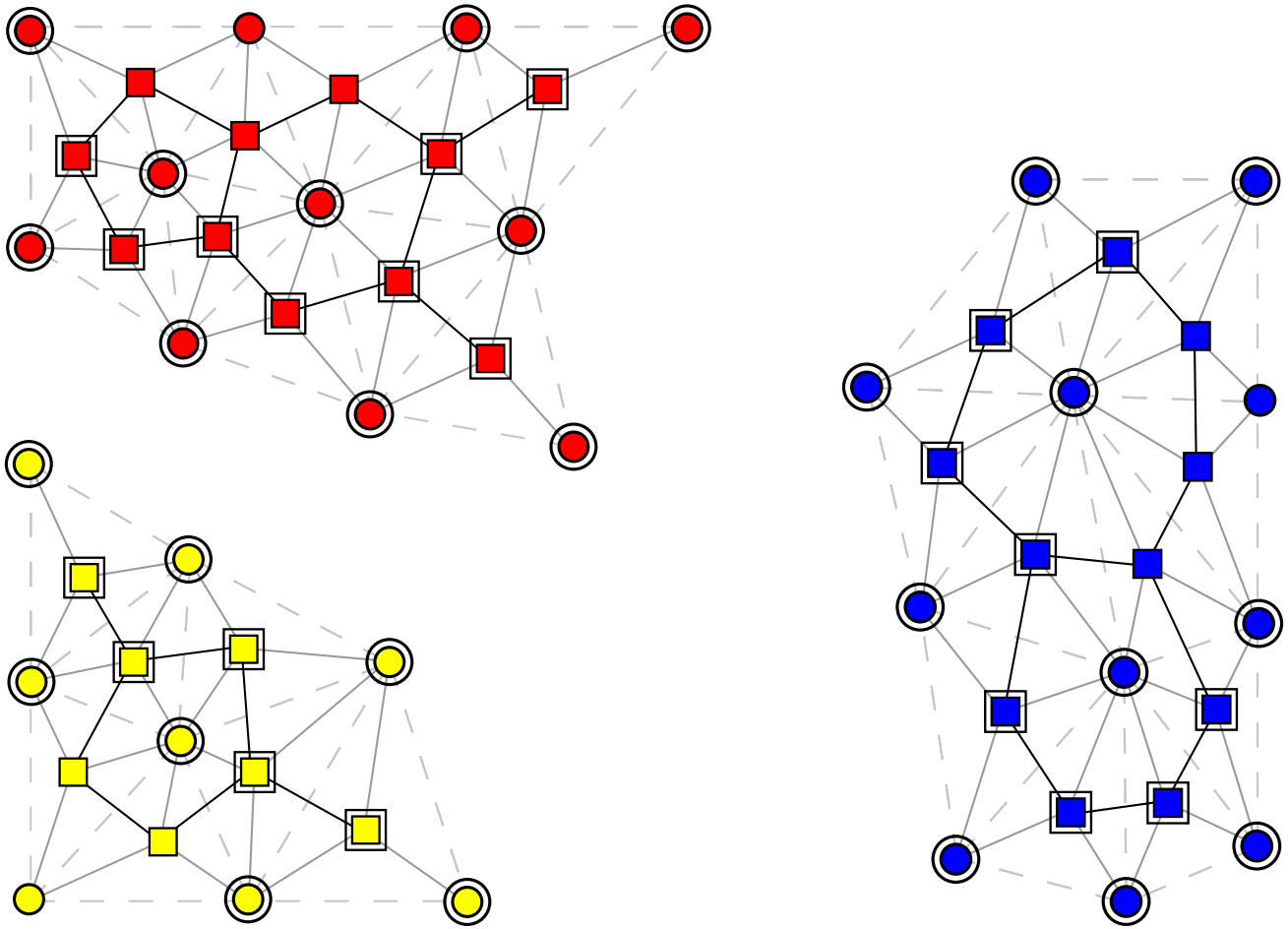


Extraction (1/3)



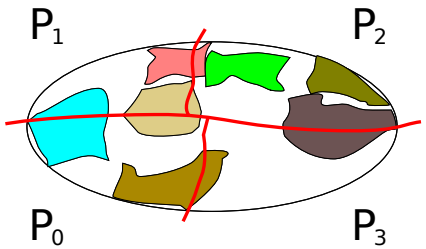
Extraction (2/3)

- Extract zones and distribute them throughout the processors:
  - zone skins will not be remeshed, except zone skins which correspond to mesh skin



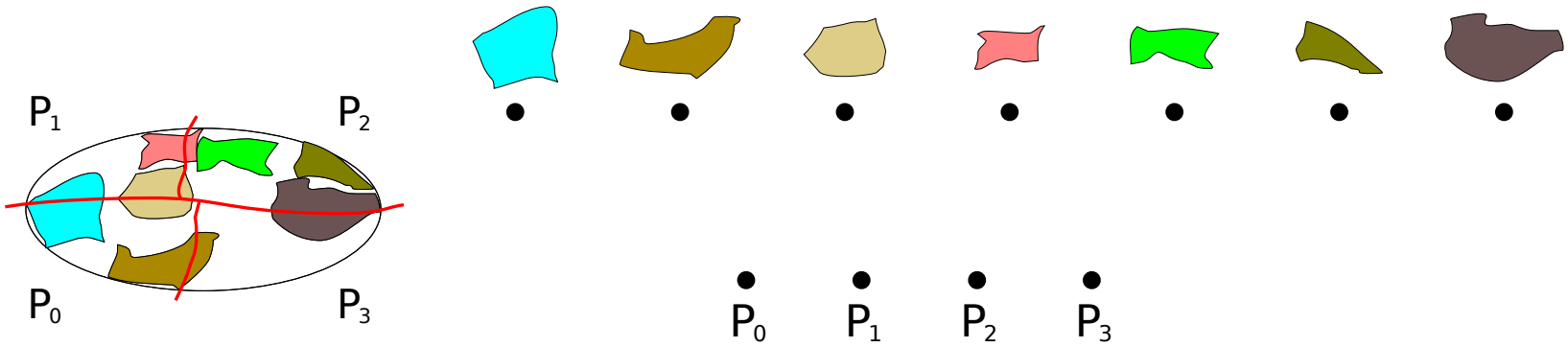
Extraction (3/3)

- Two criteria must be considered:



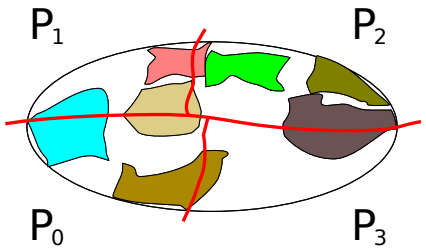
Extraction (3/3)

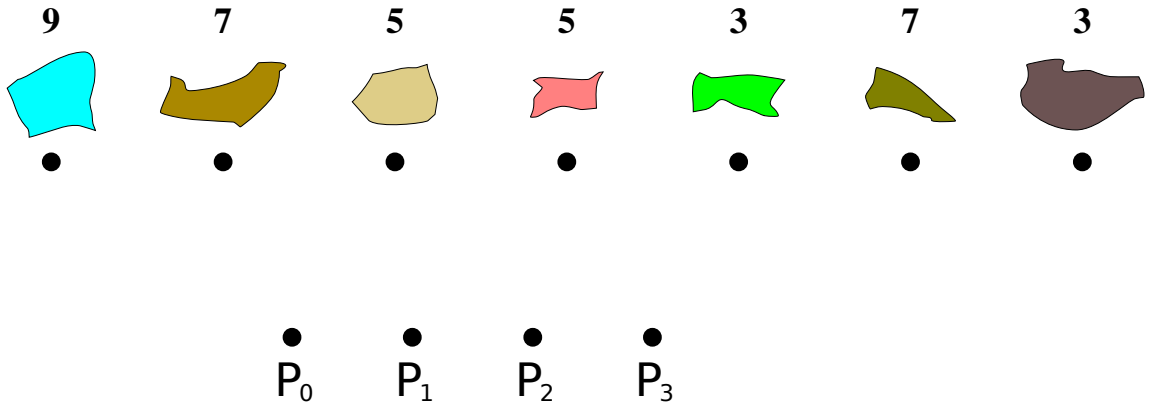
- Two criteria must be considered:
  - load-balance according to the remesher workload



Extraction (3/3)

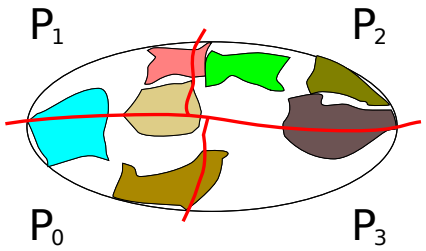
- Two criteria must be considered:
  - load-balance according to the remesher workload

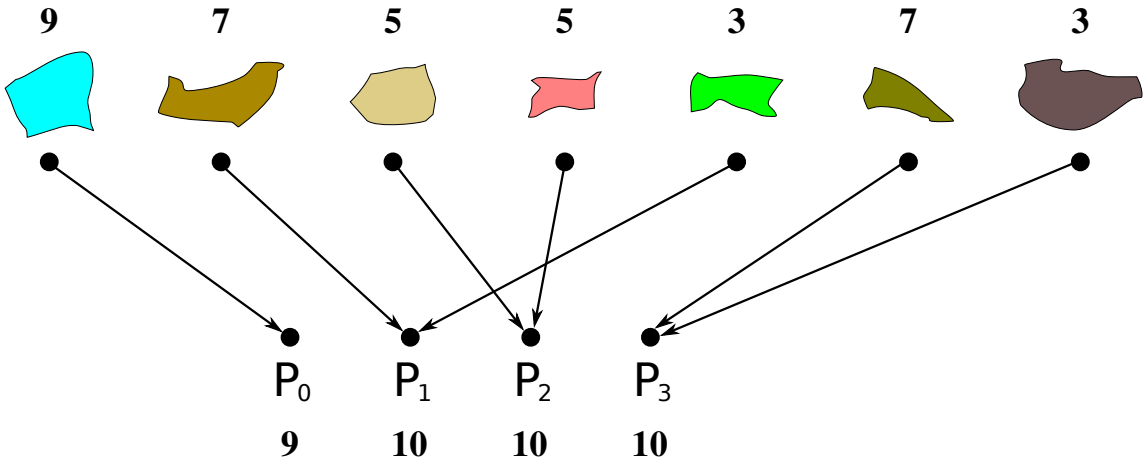




Extraction (3/3)

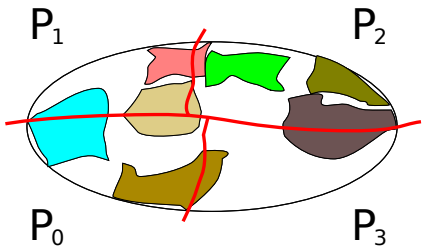
- Two criteria must be considered:
  - load-balance according to the remesher workload

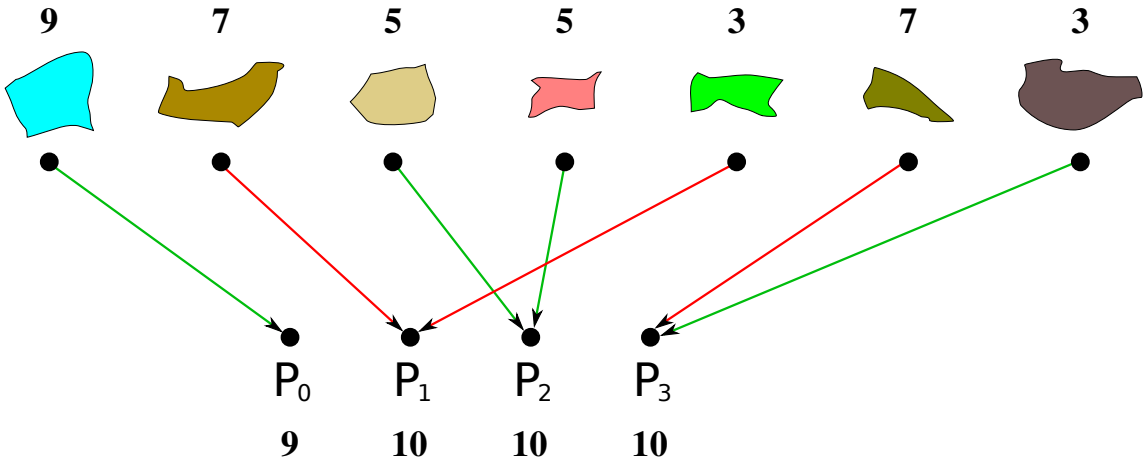




Extraction (3/3)

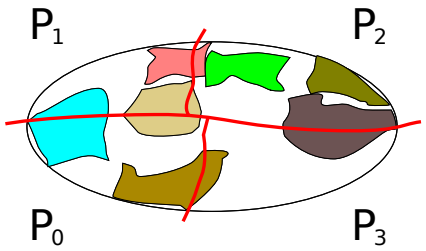
- Two criteria must be considered:
  - load-balance according to the remesher workload

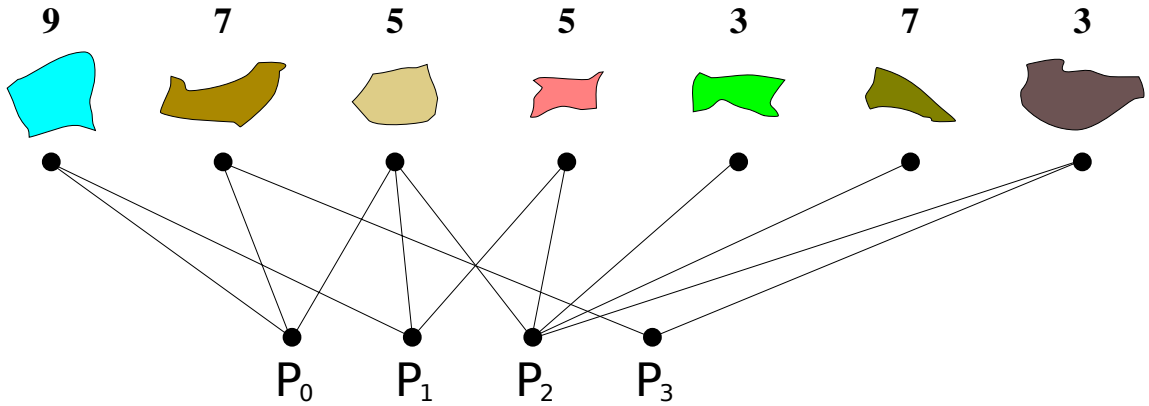




Extraction (3/3)

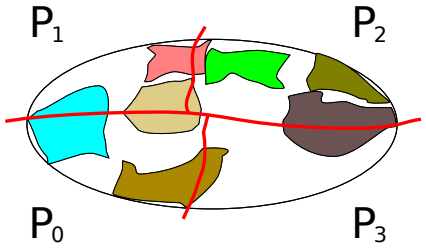
- Two criteria must be considered:
  - load-balance according to the remesher workload
  - minimize communication

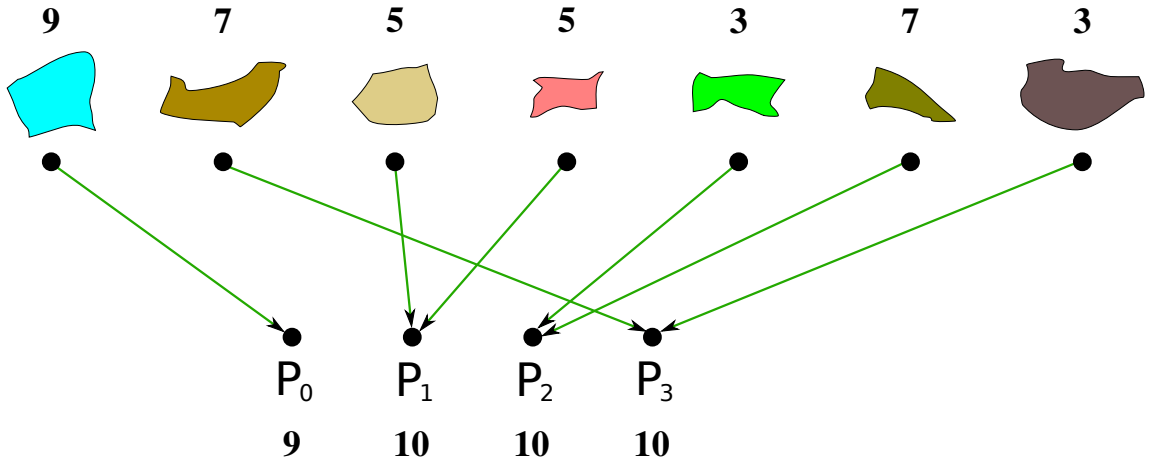




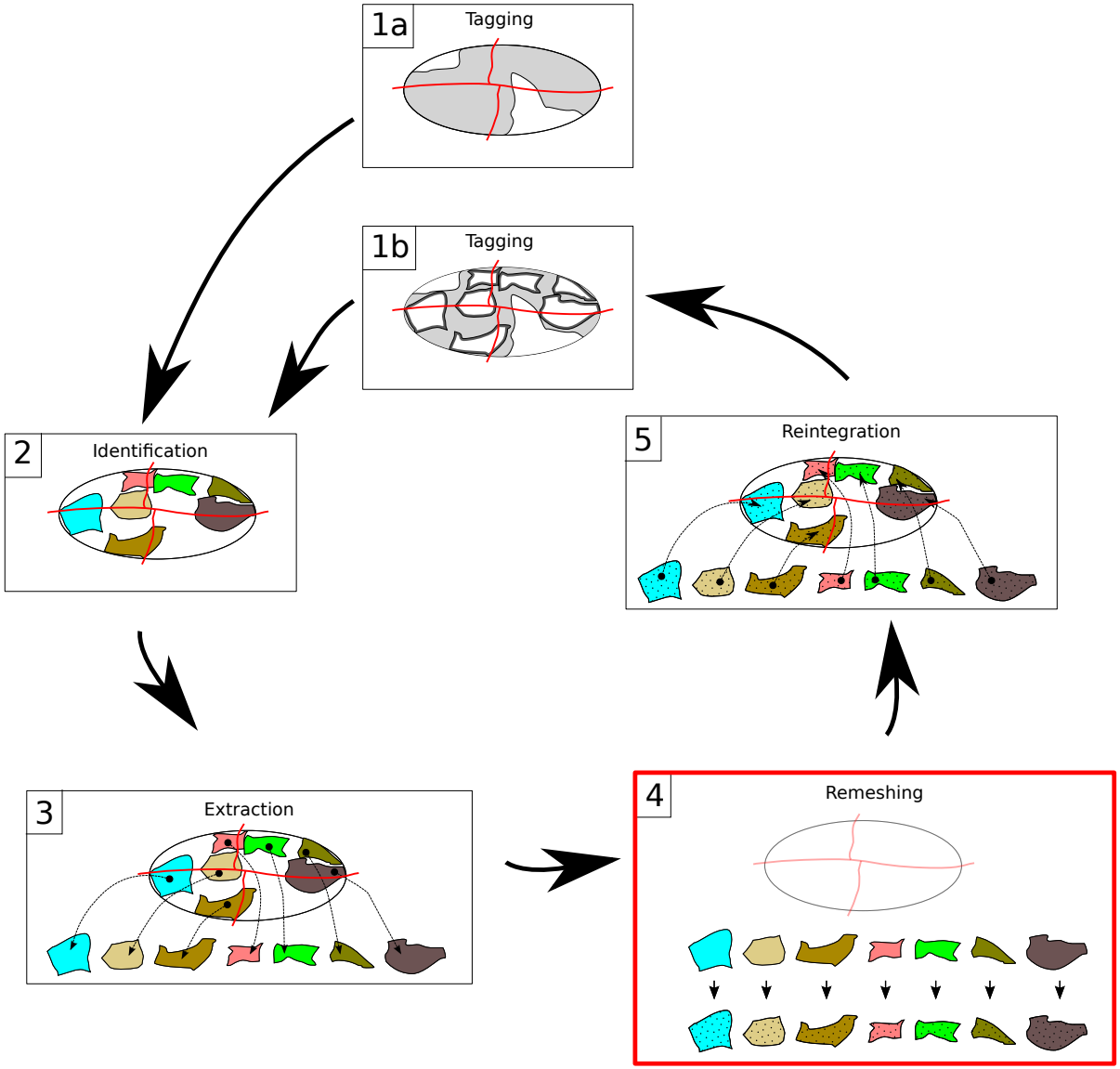
Extraction (3/3)

- Two criteria must be considered:
  - load-balance according to the remesher workload
  - minimize communication



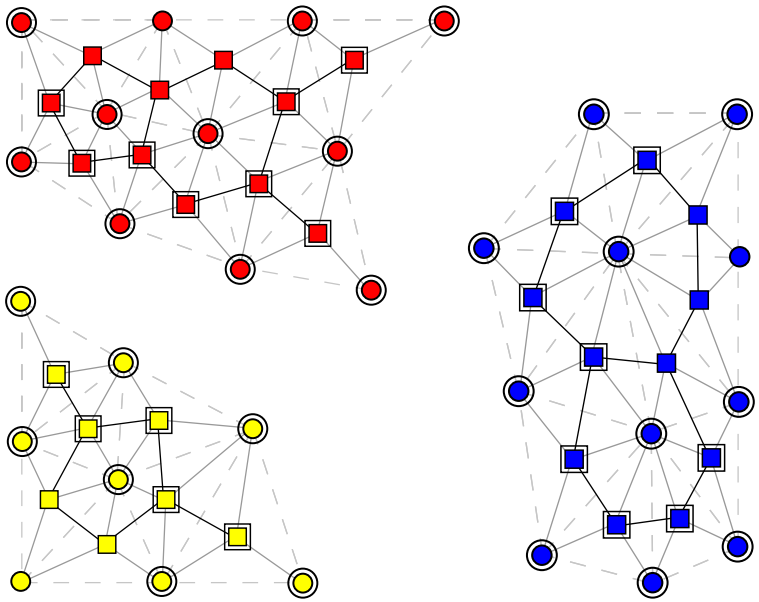


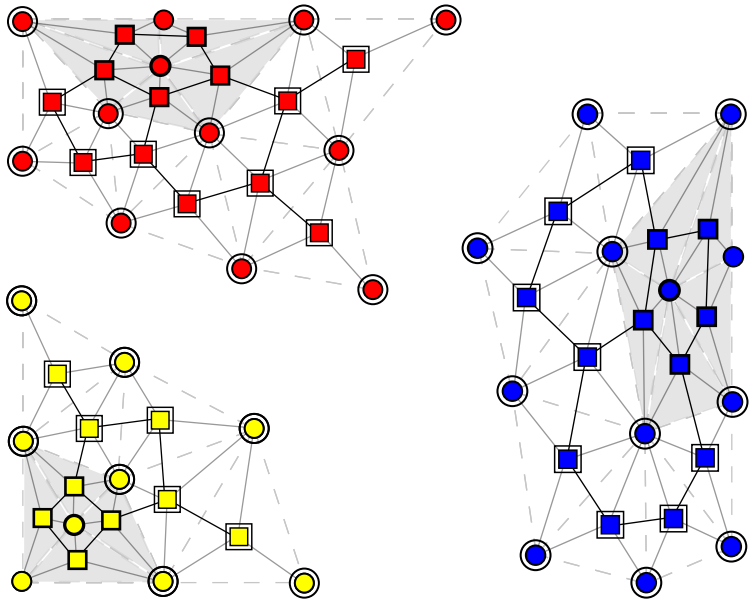
Remeshing (1/2)



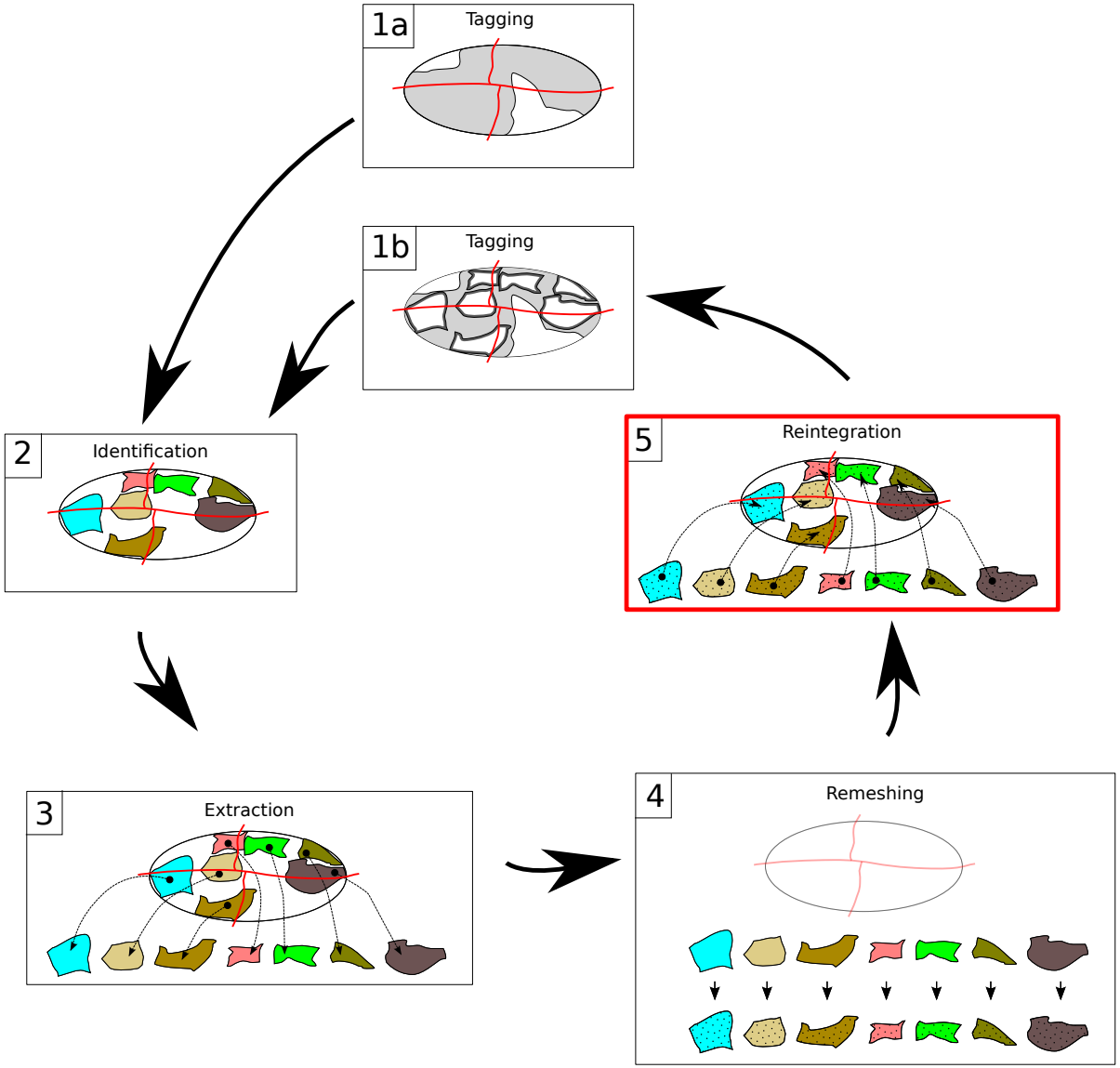
Remeshing (2/2)

- Remesh sequentially each zone on separate processors:
- Before remeshing:
- After remeshing:



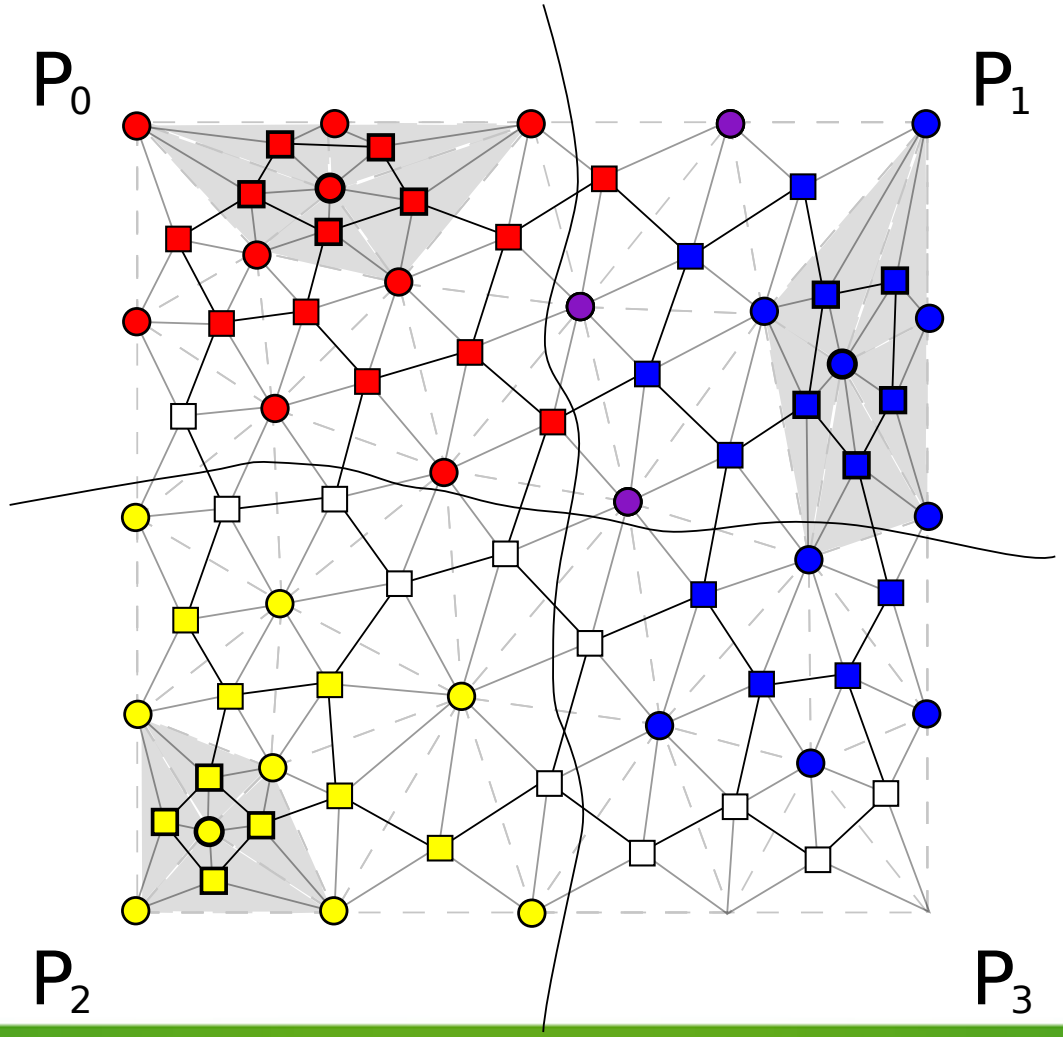


Reintegration (1/2)

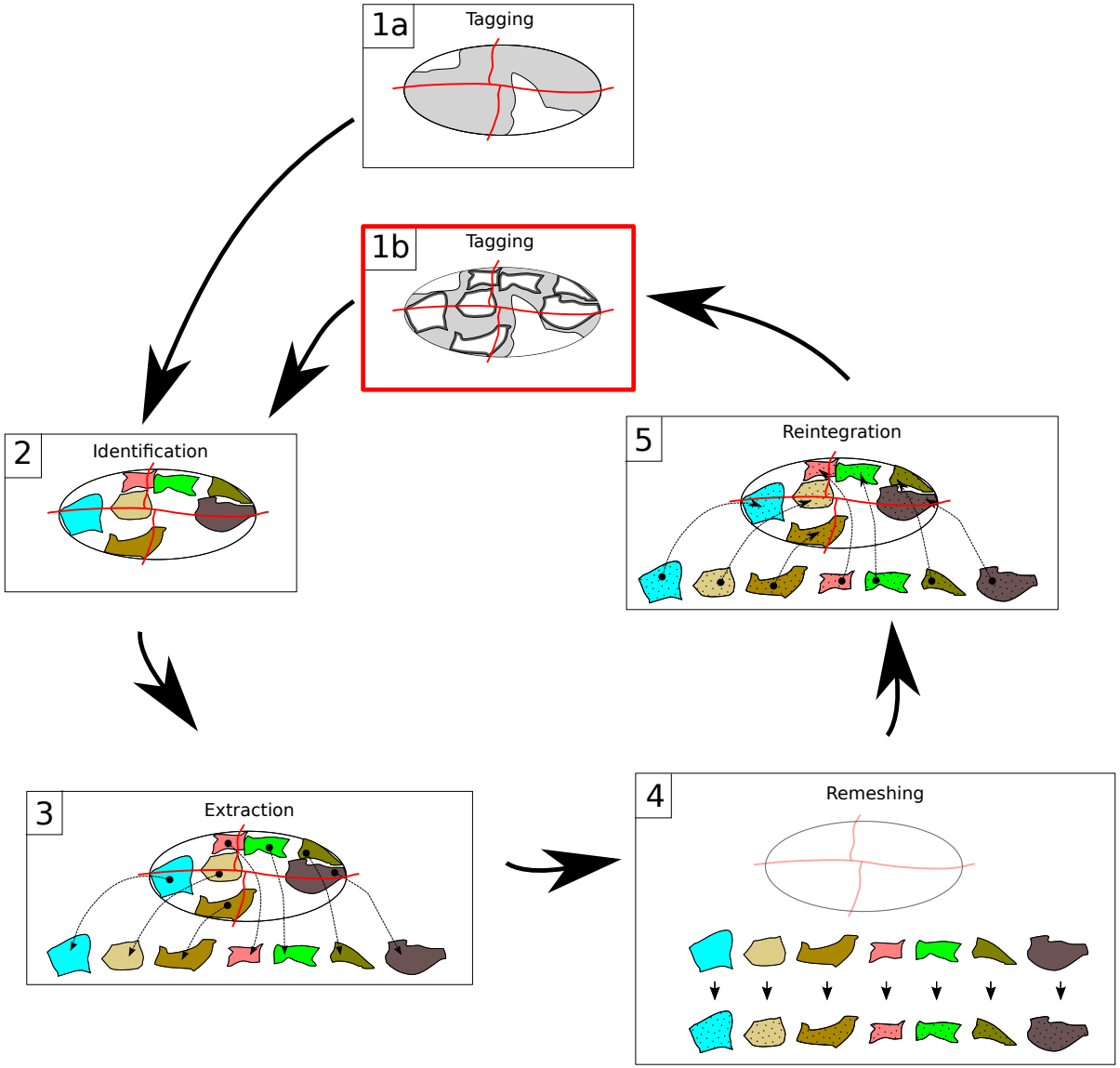


Reintegration (2/2)

- Fixed skin zone permit to reintegrate zones in distributed mesh:

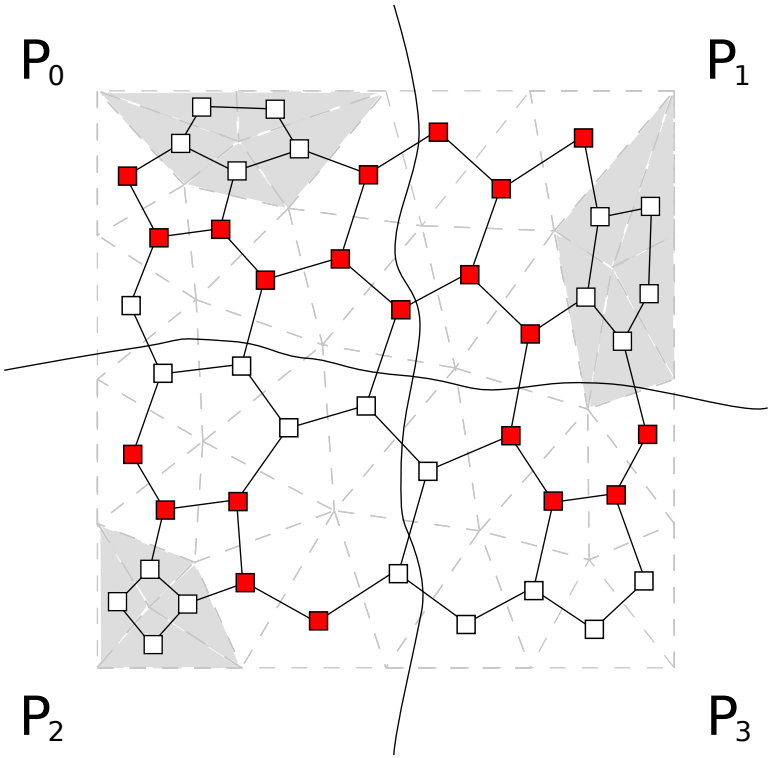


Tagging again (1/2)

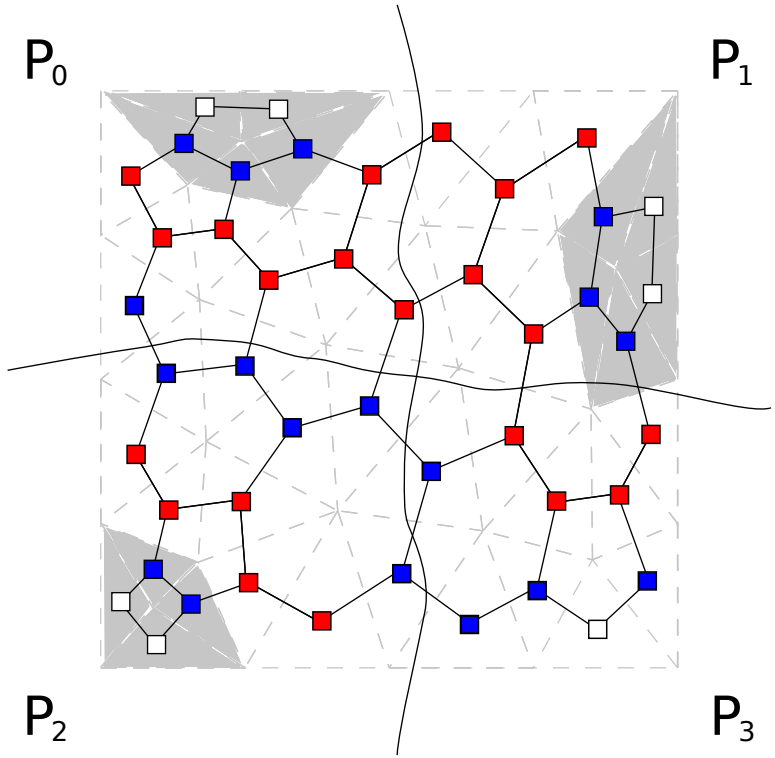


Tagging again (2/2)

- Remove tag on elements which are inside zones
- Yet a band from the skin is needed to remesh zone skin:
- Before tagging:



- After tagging:



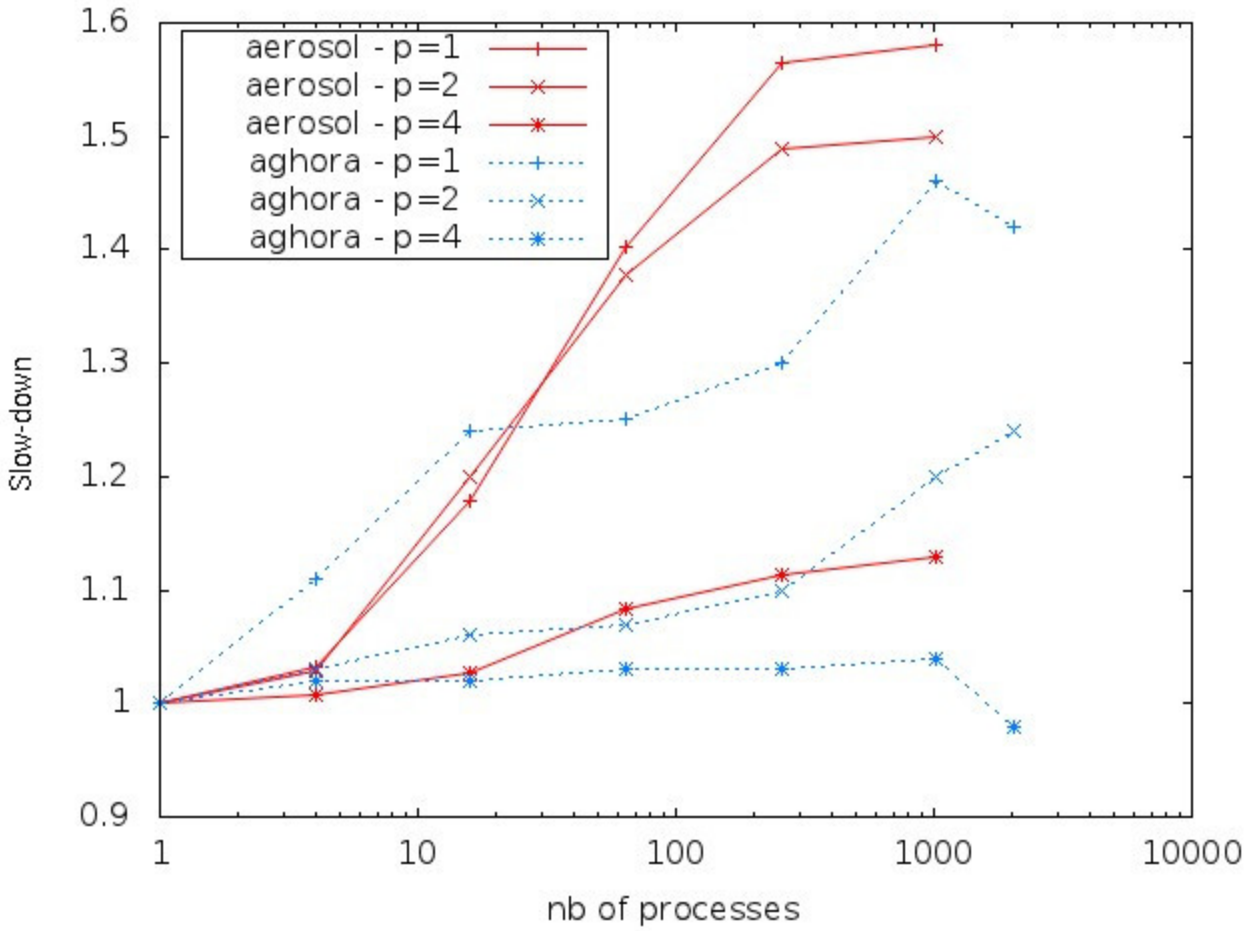
4
Experiments

Functional test case (1/3)

- Our work implemented in a library: PaMPA
- PaMPA integrated in another library: Aerosol (continuous and discontinuous finite elements library on hybrid meshes)
- Comparison of Aerosol with the Aghora library (developed by ONERA) in the Yee vortex test case:
  - implemented into Aerosol by BACCHUS and CAGIRE teams (not including me)
  - isentropic vortex in a 3D uniform and inviscid flow
  - 3D discretization with hexahedra on the unit square $[0, 1]^3$ with periodic boundary conditions
  - discontinuous Galerkin method
  - explicit Runge-Kutta time stepping

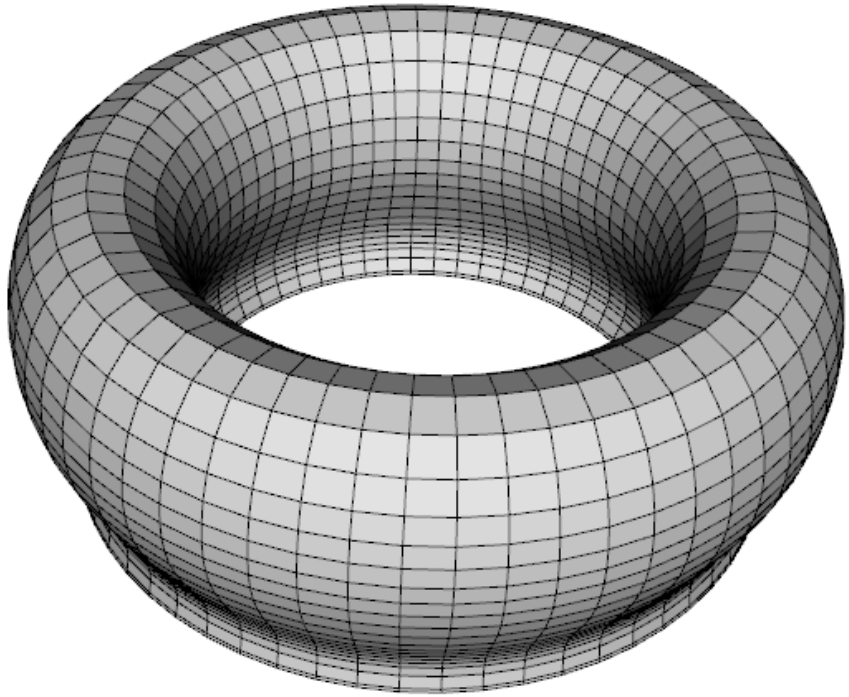
Functional test case (2/3)

- Comparison of the weak scalability
- Varying the polynomial degree p:



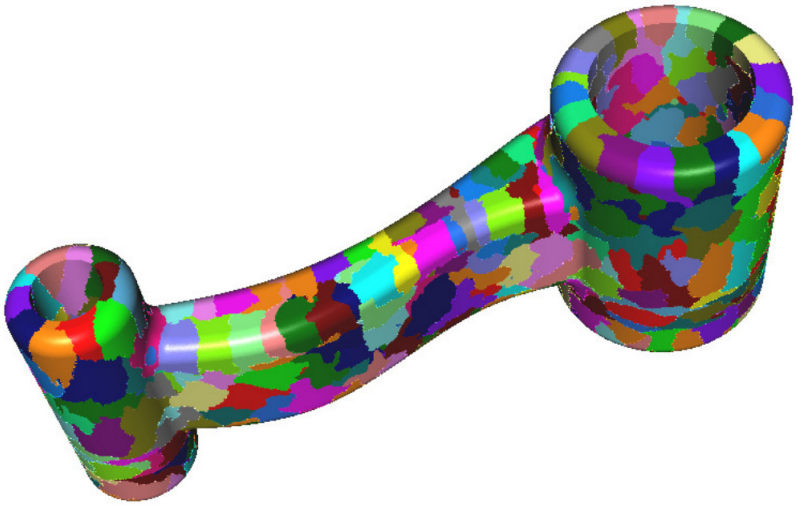
Functional test case (3/3)

- Anisotropic propagation of heat in a tokamak
- Unstationary anisotropic diffusion
- Torus mesh composed of prisms
- Discontinuous Galerkin method
- Implicit scheme
  - Linear system solved with PETsC
- Integrated by CAGIRE team

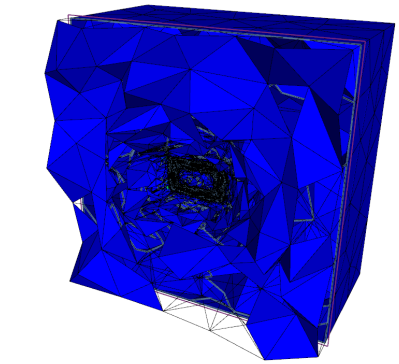


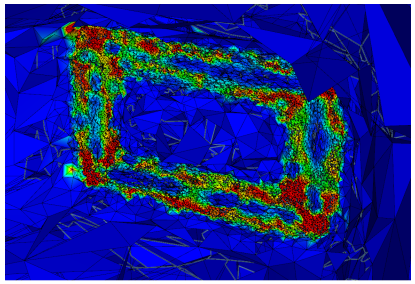
Parallel remeshing (1/5)

Isotropic mesh



Anisotropic mesh

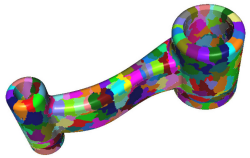




Parallel remeshing (2/5): isotropic mesh

	MMG3D on 1 processor	PaMPA-MMG3D on 24 processors
Number of elements at the start	2 423 029	
Processor frequency (GHz)	2,40	3,06
Used memory (kb)	27 588 940	51 116 044
Elapsed time	17h15m12s	00h21m14s
Elapsed time × number procs	17h15m12s	8h33m36s
Number of elements	108 126 515	115 802 876
Smallest edge length	0.1470	0.1395
Largest edge length	6.3309	11.2415
Worst element quality	294.2669	294.2669
Element quality between 1 and 2	99.65%	99.38%
Edge length between 0.71 and 1.41	97.25%	97.65%

Parallel remeshing (3/5): isotropic mesh

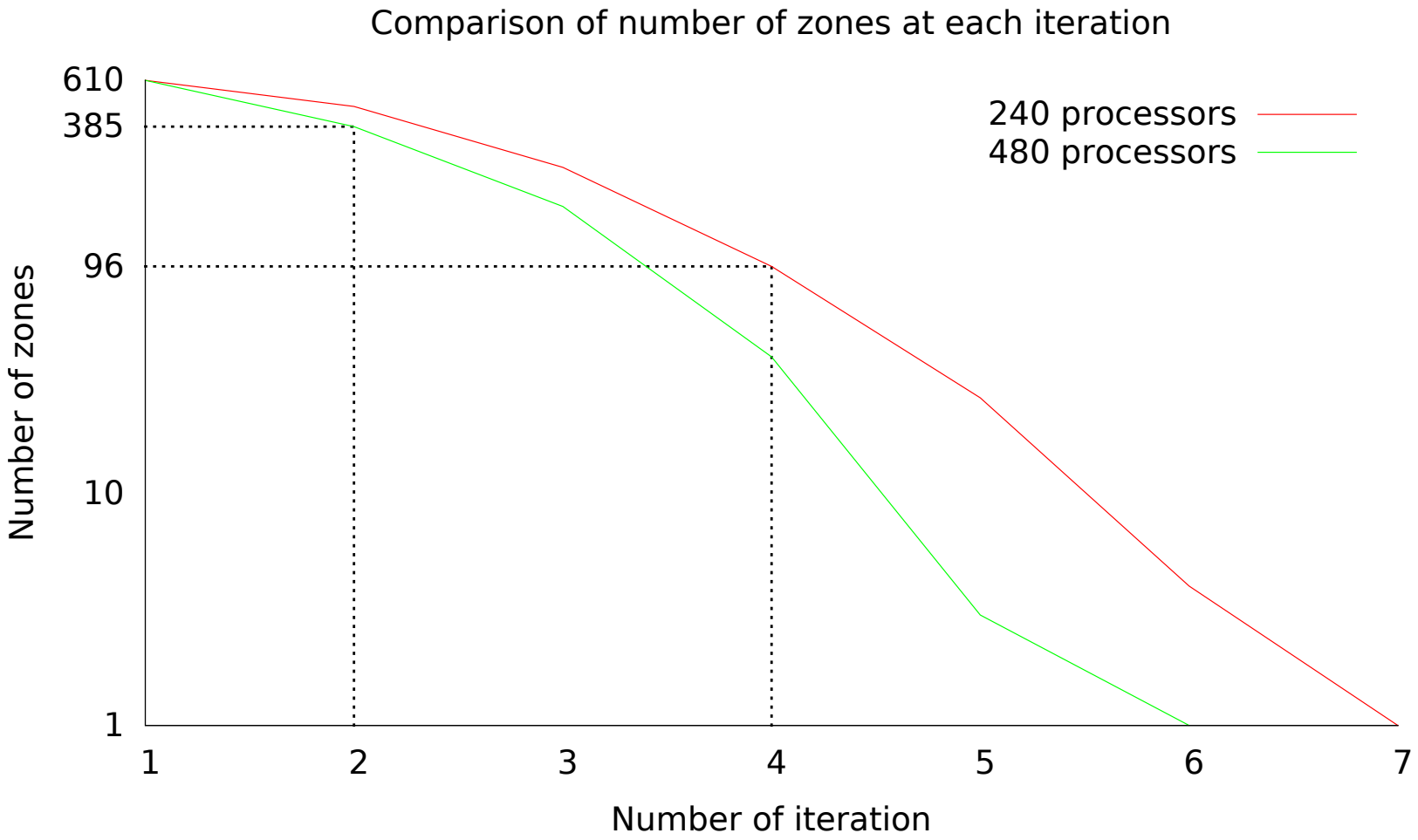


	PaMPA-MMG3D on 120 processors
Number of elements at the start	4 509 655
Elapsed time	00h34m54s
Elapsed time × number procs	69h48m
Number of elements	318 027 812
Smallest edge length	0.2862
Largest edge length	6.2161
Worst element quality	235.6651
Element quality between 1 and 2	99.58%
Edge length between 0.71 and 1.41	97.91%

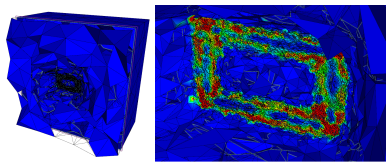
Parallel remeshing (4/5): isotropic mesh

	PaMPA-MMG3D	
	on 240 processors	on 480 processors
Number of elements at the start	27 044 943	
Used memory (kb)	651 185 792	542 832 960
Elapsed time	00h34m59s	00h29m03s
Elapsed time × number procs	139h56m	232h24m
Number of elements	609 671 387	612 426 645
Smallest edge length	0.2911	0.1852
Largest edge length	8.3451	7.3611
Worst element quality	335.7041	190.4122
Element quality between 1 and 2	98.92%	98.97%
Edge length between 0.71 and 1.41	97.20%	97.39%

Parallel remeshing (4/5): isotropic mesh

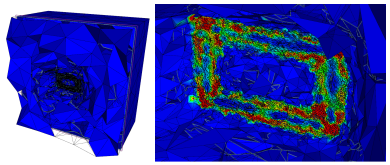


Parallel remeshing (5/5): anisotropic mesh



	MMG3D on 1 processor	PaMPA-MMG3D on 5 processors
Number of elements at the start	254 501	
Elapsed time	00h08m25s	00h06m53s
Elapsed time × number procs	00h08m25s	00h34m25s
Number of elements	11 559 197	11 741 242
Smallest edge length	0.1154	0.1133
Largest edge length	11.0161	10.1912
Worst element quality	37.7373	38.2374
Element quality between 1 and 2	98.41%	98.10%
Edge length between 0.71 and 1.41	93.29%	93.04%

Parallel remeshing (5/5): anisotropic mesh



	MMG3D on 1 processor	PaMPA-MMG3D on 5 processors
Number of elements at the start	254 501	
Elapsed time	00h08m25s	00h06m53s
Elapsed time × number procs	00h08m25s	00h34m25s
Number of elements	11 559 197	11 741 242
Smallest edge length	0.1154	0.1133
Largest edge length	11.0161	10.1912
Worst element quality	37.7373	38.2374
Element quality between 1 and 2	98.41%	98.10%
Edge length between 0.71 and 1.41	93.29%	93.04%

→ The mesh must be bigger in order to cover the overhead induced by the iterative process

5
Conclusion

Conclusion

- We have devised an efficient scheme for parallel remeshing of very large meshes, which can be coupled with any sequential remesher
- We can achieve the same quality than sequential remeshers
- Several algorithms as parallel zone identification have been developed and implemented:
  - graph coarsening
  - seed / "crystal" growing
  - graph partitioning

Prospects

- Short-term prospects:
  - Remeshing large-size anisotropic meshes
  - Testing scalability on parallel remeshing
  - Renumbering entities in parallel remeshing
  - Remeshing mesh skin by plugging-in a sequential remesher which remesh the skin: MMG3D5
- Long-term prospects:
  - parallel repartitioning during parallel remeshing
  - Adding more parallel zone identification algorithms
  - parallel interpolation coupled with parallel remeshing (PhD Thesis: Léo Nouveau)

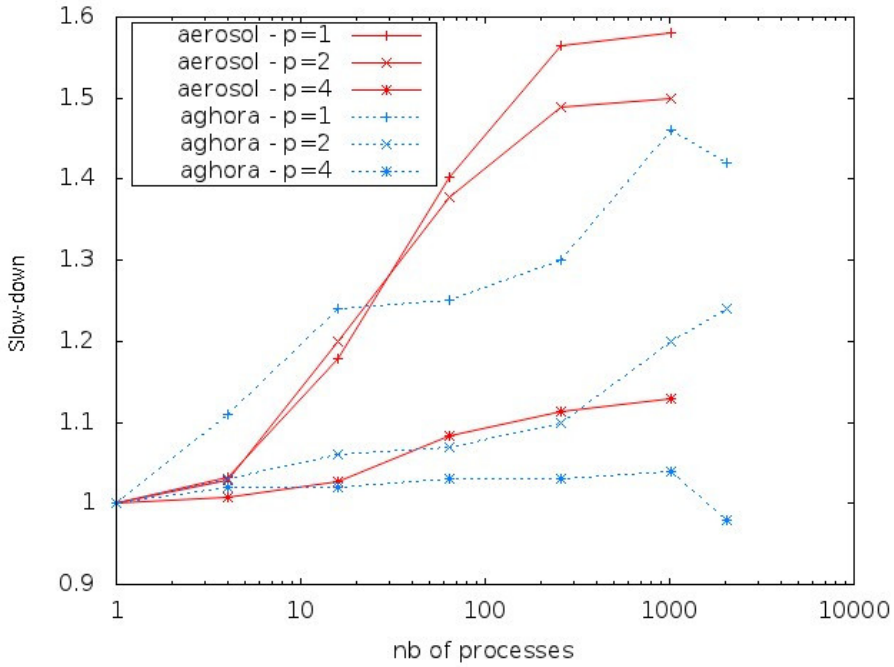
Thank you for your attention

PhD Defense

pampa.bordeaux.inria.fr

About functional test case

- p degree means $5 * (p+1)^3 * 200$ unknowns by sub-domain:
  - $p = 1$: 8000 unknowns
  - $p = 2$: 27000 unknowns
  - $p = 3$: 64000 unknowns
  - $p = 4$: 125000 unknowns



Nina Amenta, Sunghee Choi, and Günter Rote. Incremental constructions con brio. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 211–219. ACM, 2003.

D.A. Burgess and M.B. Giles. Renumbering unstructured grids to improve the performance of codes on hierarchical memory machines. *Advances in Engineering Software*, 28(3):189–201, 1997.

A Casagrande, P Leyland, L Formaggia, and M Sala. Parallel mesh adaptation. *Series on Advances in Mathematics for Applied Sciences*, 69:201, 2005.

Jose G Castanos and John E Savage. The dynamic adaptation of parallel mesh-based computation. 1996.

Peter A Cavallo, Neeraj Sinha, and Gregory M Feldman. Parallel unstructured mesh adaptation method for moving body applications. *AIAA journal*, 43(9):1937–1945, 2005.

Cédric Chevalier. *Conception et mise en oeuvre d'outils efficaces pour le partitionnement et la distribution parallèles de problème numériques de très grande taille*. PhD thesis, Université Sciences et Technologies-Bordeaux I, 2007.

Cédric Chevalier and Ilya Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *Learning and Intelligent Optimization*, pages 191–205. Springer, 2009.

Nikos Chrisochoides and Démian Nave. Parallel delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering*, 58(2):161–176, 2003.

Thierry Coupez, Hugues Digonnet, and Richard Ducloux. Parallel meshing and remeshing. *Applied Mathematical Modelling*, 25 (2):153–175, 2000.

Karen Devine, Bruce Hendrickson, Erik Boman, Matthew St John, Courtenay Vaughan, and WF Mitchell. Zoltan: A dynamic load-balancing library for parallel applications; userś guide. *Sandia National Laboratories Tech. Rep*, 1999.

Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson, and Courtenay Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science & Engineering*, 4(2):90–96, 2002.

Hugues Digonnet, Marc Bernacki, Luisa Silva, and Thierry Coupez. Adaptation de maillage en parallèle, application à la simulation de la mise en forme des matériaux. In *Congrès Français de Mécanique Grenoble-CFM 2007*, page 6 pages, Grenoble, France, 2007. URL http://hal-ensmp.archives-ouvertes.fr/hal-00521844. http://hdl.handle.net/2042/16046.

C Dobrzynski and JF Remacle. Parallel mesh adaptation. *International Journal for Numerical Methods in Engineering*, 2007.

J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 14(1):1–17, March 1988. ISSN 0098-3500. URL http://doi.acm.org/10.1145/42288.42291.

C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181. IEEE, 1982.

Martin Isenburg, Yuanxin Liu, Jonathan Shewchuk, and Jack Snoeyink. Streaming computation of delaunay triangulations. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1049–1056. ACM, 2006.

B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *BELL System Technical Journal*, pages 291–307, February 1970.

Ing Lutz Laemmer. Parallel mesh generation. In *Solving Irregularly Structured Problems in Parallel*, pages 1–12. Springer, 1997.

Orion Lawlor, Sayantan Chakravorty, Terry Wilmarth, Nilesh Choudhury, Isaac Dooley, Gengbin Zheng, and Laxmikant Kalé. Parfum: a parallel framework for unstructured meshes for scalable dynamic physics applications. *Engineering with Computers*, 22:215–235, 2006. ISSN 0177-0667. URL http://dx.doi.org/10.1007/s00366-006-0039-5. 10.1007/s00366-006-0039-5.

Rainald Löhner. Some useful renumbering strategies for unstructured grids. *International Journal for Numerical Methods in Engineering*, 36(19):3259–3270, 1993.

Rainald Löhner. Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines. *Computer methods in applied mechanics and engineering*, 163(1):95–109, 1998.

Adrien Loseille and Frédéric Alauzet. Shrimp User Guide. A Fast Mesh Renumbering and Domain Partionning Method. Technical Report RT-0362, INRIA, 2009. URL http://hal.inria.fr/inria-00362994.

Bart Maerten, Dirk Roose, Achim Basermann, Jochen Fingberg, and Guy Lonsdale. Drama: A library for parallel dynamic load balancing of finite element applications. In Patrick Amestoy, Philippe Berger, Michel Daydé, Daniel Ruiz, Iain Duff, Valérie Frayssé, and Luc Giraud, editors, *Euro-Par'99 Parallel Processing*, volume 1685 of *Lecture Notes in Computer Science*, pages 313–316. Springer Berlin / Heidelberg, 1999. ISBN 978-3-540-66443-7. URL http://dx.doi.org/10.1007/3-540-48311-X_40. 10.1007/3-540-48311-X_40.

Leonid Oliker, Rupak Biswas, and Harold N Gabow. Parallel tetrahedral mesh adaptation with dynamic load balancing. *Parallel Computing*, 26(12):1583–1608, 2000.

Alex Pothen, Horst D Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

M. Ramadan, L. Fourment, and H. Digonnet. A parallel two mesh method for speeding-up processes with localized deformations: application to cogging. *International Journal of Material Forming*, 2:581–584, 2009. ISSN 1960-6206. URL http://dx.doi.org/10.1007/s12289-009-0440-x. 10.1007/s12289-009-0440-x.

U Tremel, KA Sørensen, S Hitzel, H Rieger, Oubay Hassan, and Nigel P Weatherill. Parallel remeshing of unstructured volume grids for cfd applications. *International journal for numerical methods in fluids*, 53(8):1361–1379, 2007.