Aspect Ratio for Mesh Partitioning

Ralf Diekmann[1], Robert Preis[1], Frank Schlimbach[2], and Chris Walshaw[2]

[1] University of Paderborn, Germany,
{diek@,preis@hni.}uni-paderborn.de
[2] University of Greenwich, London, UK,
{sf634,C.Walshaw}@gre.ac.uk

Abstract. This paper deals with the measure of *Aspect Ratio* for mesh partitioning and gives hints why, for certain solvers, the *Aspect Ratio* of partitions plays an important role. We define and rate different kinds of *Aspect Ratio*, present a new center-based partitioning method which optimizes this measure implicitly and rate several existing partitioning methods and tools under the criterion of *Aspect Ratio.*

1 Introduction

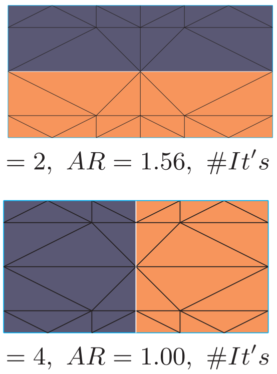
Almost all numerical scientific simulation codes belong to the class of *data-parallel applications*: their parallelization executes the same kind of operations on every processor but on different parts of the data. This requires the partitioning of the mesh into equal-sized subdomains as preprocessing step. Together with the additional constraint of minimizing the number of cut edges (i.e. minimizing the total interface length in the case of FE-mesh partitioning), the problem is NP-complete. Thus, a number of graph (mesh) partitioning heuristics have been developed in the past (e.g. [8]), and used in practical applications.

Most of the existing graph partitioning algorithms optimize the *balance* of subdomains plus the number of cut edges, the *cut size*. This is sufficient for many applications, because an equal balance is necessary for a good utilization of the processors of a parallel system and the *cut size* indicates the amount of data to be transfered between different steps of the algorithms. The efficiency of parallel versions of global iterative methods like (relaxed) Jacobi, Conjugate Gradient (CG) or Multigrid is (mainly) determined by these two measures. But if the decomposition is used to construct pre-conditioners (DD-PCG) [1,2], cut and balance may not longer be the only efficiency determining factors. The *shape* of subdomains heavily influences the quality of pre-conditioning and, thus, the overall execution time [10]. First attempts at optimizing the *Aspect Ratio* of subdomains weigh elements depending on their distance from a subdomain's center (e.g. [5]) and include this weight into the cost function of local iterative search algorithms like the Kernighan-Lin heuristic [6].

In the following section we will define the *Aspect Ratio* of subdomains and give hints why it can improve the overall execution time of domain decomposition methods. Section 3 introduces a new mesh partitioning heuristic which implicitly optimizes the *Aspect Ratio* and Section 4 finishes with experimental results. Further information on this topic can be found in [4].

2 Aspect Ratios

An example that *cut size* is not always the right measure in mesh partitioning can be found in Fig. 1. The sample mesh is partitioned into two parts with different AR's and different cuts. A Poisson problem with homogeneous Dirichlet-0 boundary conditions is solved with DD-PCG. It can be observed that the number of iterations is determined by the Aspect Ratio and that the cut size (number of neighboring elements) would be the wrong measure.



$cut = 2,\ AR = 1.56,\ \#It's = 8$
$cut = 4,\ AR = 1.00,\ \#It's = 5$

Fig. 1. *Aspect Ratio* vs. *Cut Size.*

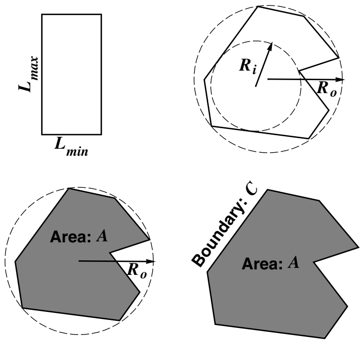


Fig. 2. Different definitions of *Aspect Ratio AR:* $\frac{L_{\max}}{L_{\min}}$, $\frac{R_o^2}{R_i^2}$, $\frac{R_o^2}{A}$ *and* $\frac{C^2}{16A}$.

Possible definitions of AR can be found in Fig. 2. The first two are motivated by common measures in triangular mesh generation where the quality of triangles are expressed in either $\frac{L_{\max}}{L_{\min}}$ (longest to shortest boundary edge) or $\frac{R_o^2}{R_i^2}$ (area of smallest circle containing the domain to area of largest inscribed circle). The definition of $AR = \frac{R_o^2}{R_i^2}$ expresses the fact that circles are perfect shapes. Unfortunately, the circles are quite expensive to find for arbitrary polygons: using Voronoi-diagrams, both can be determined in $O(2n \log n)$ steps where n is number of nodes of the polygon. $AR = \frac{R_o^2}{A}$ (A is the area of the domain) is another measure favoring circle-like shapes. It still requires the determination of the smallest outer circle but turns out to be better in practice. We can do a further step and approximate R_o by the length C of the boundary of the domain (which can be determined fast and updated incrementally in $O(1)$). The definition of $AR = \frac{C^2}{16A}$ additionally assumes that squares are perfect domains. Circles offer a better circumfence/area ratio but force neighboring domains to become non-convex (Fig. 3). For a sub-domain with area A and circumfence C $AR = \frac{C^2}{16A}$ is the ratio between the area of a square with circumfence C and A.

For irregular meshes and partitions, the first definition $AR = \frac{L_{\max}}{L_{\min}}$ does not express the *shape* properly. Fig. 4 shows an example. P_1 is perfectly shaped, but as the boundary towards P_4 is very short, $\frac{L_{\max}}{L_{\min}}$ is large. The circle-based

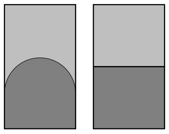


Fig. 3. Circles as perfect shapes?

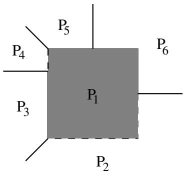


Fig. 4. $\frac{L_{\max}}{L_{\min}}$?

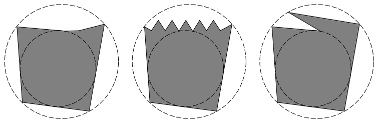


Fig. 5. Problems of $AR := \frac{R_o^2}{R_i^2}$.

measures usually fail to rate complex boundaries like "zig-zags" or inscribed corners. Fig. 5 shows examples which have the same AR each but are very different in shape.

3 Mesh Partitioning

The task of mesh partitioning is to divide the set of elements of a mesh into a given number of parts. The usual optimization criteria are the *balance* and the *cut*, which is the number of crossing edges of the *element graph* (the element graph expresses the data-dependencies between elements in the mesh). The calculation of a balanced partition with minimum cut is NP-complete. We will consider methods included in the tools **JOSTLE** [11], **METIS** [7] and **PARTY** [9].

The *coordinate partitioning* (**COO**) method cuts the mesh by a straight line perpendicular to the axis of the longest elongation such that both parts have an equal number of elements, resulting in a stripe-wise partition. If used recursively (**COO_R**), it usually results in a more box-wise partition. There are many examples for *greedy* methods , in which one element after another is added to a subdomain. We will consider the variations in which the next element to be added is either chosen in a breadth-first manner (**BFS**) or as the one which increases the cut least of all (**CFS**).

The new method *Bubble* (**BUB**) is designed to create compact domains and the idea is to grow subdomains simultaneously from different *seed* elements. A partition is represented by a set of (initial random) seeds, one for each part. Starting from the seeds, the parts are grown in a breadth-first manner until all elements are assigned to a subdomain. Each subdomain then computes its center based on the graph distance measure, i.e. it determines the element which is "nearest" to all others in the subdomain. These center-elements are taken as new seeds and the process is started again. The iteration stops if the seeds do not move any more, or if an already evaluated configuration is visited again. In general, the method stops after few iterations and produces connected and very compact parts, i.e. the shape of the parts is naturally smooth. As drawback, the parts do not have to contain the same number of elements. Therefore we integrated a global load balancing step based on the diffusion method [3] as post-processing trying to optimize either the cut (**BUB+CUT**) or the Aspect Ratio (**BUB+AR**).

We also implemented the meta-heuristic *Simulated Annealing* (**SA**) for the shape optimization and used $AR = \frac{C^2}{16A}$ as optimization function. If the parameters are chosen carefully, SA is able to find very good solutions. Unfortunately, it also requires a very large number of steps.

4 Results

We compare the described approaches with respect to the number of global iterations, the cut size and the Aspect Ratio in Fig. 6 and 7. Methods 1-7 are included in PARTY and 8-11 are default settings of the tools. The partitions are listed with the method numbers with increasing number of iterations.

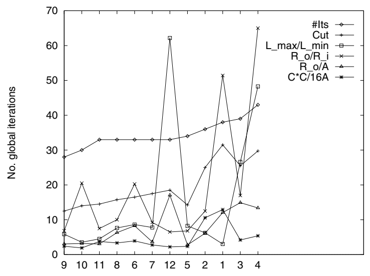


Fig. 6. Results of example *turm* (531 elements) into 8 parts.

1:COO
2:COO_R
3:BFS
4:CFS
5:BUB
6:BUB+CUT
7:BUB+AR
8:PARTY
9:JOSTLE
10:KMETIS
11:PMETIS
12:SA

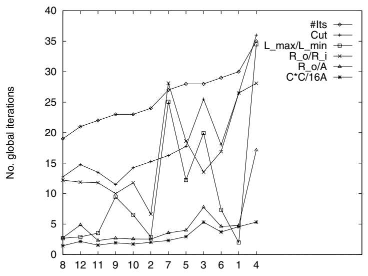


Fig. 7. Results of example *cooler* (749 elements) into 8 parts.

The Aspect Ratios $\frac{L_{max}}{L_{min}}$ and $\frac{R_o}{R_i}$ do not, whereas the values of $\frac{R_o}{A}$ and $\frac{C^2}{16A}$ roughly follow the number of iterations. A low *cut* also leads to a fairly low number of iterations. The coordinate and greedy methods usually result in very high, whereas the bubble variations in adequate iteration numbers. Very low iteration numbers can be observed for the tools and Simulated Annealing.

References

1. S. Blazy, W. Borchers, and U. Dralle. Paralleliziation methods for a characteristic's pressure correction scheme. In *Flow Simulation with High-Perf. Comp. II*, 1995. 347
2. J.H. Bramble, J.E. Pasciac, and A.H. Schatz. The construction of preconditioners for elliptic problems by substructering i.+ii. *Math. Comp.*, 47+49, 1986+87. 347
3. G. Cybenko. Load balancing for distributed memory multiprocessors. *J. Par. Distr. Comp.*, 7:279–301, 1989. 349
4. R. Diekmann, F. Schlimbach, and C. Walshaw. Quality balancing for parallel adaptive fem. In *IRREGULAR*, LNCS. Springer, 1998. 347
5. N. Chrisochoides et.al. Automatic load balanced partitioning strategies for pde computations. In *Int. Conf. on Supercomp.*, pages 99–107, 1989. 347

6. C. Farhat, N. Maman, and G. Brown. Mesh partitioning for implicit computations via iterative domain decomposition. *Int. J. Num. Meth. Engrg.*, 38:989–1000, 1995. 347
7. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, University of Minnesota, 1995. 349
8. B.W. Kernighan and S. Lin. An effective heuristic procedure for partitioning graphs. *The Bell Systems Technical Journal*, pages 291–308, Feb 1970. 347
9. R. Preis and R. Diekmann. The party partitioning-library, user guide, version 1.1. Technical Report TR-RSFB-96-024, Universität–GH Paderborn, Sep 1996. 349
10. D. Vanderstraeten, R. Keunings, and C. Farhat. Beyond conventional mesh partitioning algorithms... In *SIAM Conf. on Par. Proc.*, pages 611–614, 1995. 347
11. C. Walshaw, M. Cross, and M.G. Everett. A localised algorithm for optimising unstructured mesh partitions. *Int. J. Supercomputer Appl.*, 9(4):280–295, 1995. 349