A New Approach to Parallel Dynamic Partitioning for Adaptive Unstructured Meshes

Gerd Heber
CAPSL, 140 Evans Hall
University of Delaware
Newark, DE 19716, USA
heber@capsl.udel.edu

Rupak Biswas
MRJ Technology Solutions
NASA Ames Research Center
Moffett Field, CA 94035, USA
rbiswas@nas.nasa.gov

Guang R. Gao
CAPSL, 140 Evans Hall
University of Delaware
Newark, DE 19716, USA
ggao@capsl.udel.edu

Abstract

Classical mesh partitioning algorithms were designed for rather static situations, and their straightforward application in a dynamical framework may lead to unsatisfactory results, e.g., excessive data migration among processors. Furthermore, special attention should be paid to their amenability to parallelization. In this paper, a novel parallel method for the dynamic partitioning of adaptive unstructured meshes is described. It is based on a linear representation of the mesh using self-avoiding walks.

1. Introduction

The three biggest issues in the parallel implementation of adaptive unstructured grid applications are runtime partitioning, mapping, and data locality. However, it is generally believed that graph partitioning is no longer the bottleneck but data remapping is (see [1]). This is because very powerful general-purpose graph partitioning methods have been developed over the last decade (see [2]). Why then do we propose yet another new partitioner? This is because an intimate relationship exists between the three big issues, and resolving one in an optimal fashion does not automatically imply the same for the others. Determining a new "optimal" partitioning at runtime can be done very rapidly, but what is the cost for moving around a lot of data accordingly? Effective techniques for a better exploitation of data locality based on *space-filling curves* [7] have been successfully applied in the runtime partitioning for N–body simulations and finite element methods (FEM). The expectation that the serialization inherent in such curves would eliminate the partitioning and mapping problem did not completely live up to, primarily because of the difficulties in their construction for non-simply connected domains. On the other hand, partitioners do not necessarily preserve in their repartitioning structures what have to be established for the exploitation of locality. Thus, there is a clear need for a unifying framework.

A key consideration for efficient runtime partitioning is the reuse of existing partition information. We suggest a novel concept which should allow a much faster deformation of partition boundaries than diffusive methods, and which does not require any post-partitioning smoothing operation. Furthermore, it takes into account the linearization of partitions by self-avoiding walks (see [4]). Our approach views the mesh as a graph, but also as determined by the structure of the underlying physical domain, thereby linking the three main issues to mesh generation.

In the following sections, we first sketch the theoretical background and then discuss some algorithmic aspects of the constructions involved. In view of its capabilities for automatic task migration and the very fine-grained synchronization structure of the algorithms, a parallel implementation using a system supporting *multithreading* [5] seems to be the most promising approach.

In Section 2, we sketch a linearization technique based on self-avoiding walks. In Section 3, we develop a new partitioning method that is suitable for runtime purposes and which respects our linear representation of meshes.

2. Self-Avoiding Walks

Consider an arbitrary two-dimensional mesh $\mathcal{M}$. A *self-avoiding walk* (SAW) over $\mathcal{M}$ is an enumeration of the triangles of $\mathcal{M}$ such that two triangles following one another in a walk, share an edge or a vertex. Note that an SAW visits each triangle exactly once. Since we allow consecutive triangles to share either an edge or a vertex, this is *not* a Hamiltonian cycle problem on the dual graph. In fact, the non-Hamiltonicity of the duals of simple meshes forces us to relax certain assumptions. An SAW "enters" a given triangle over an edge or a vertex, and "leaves" it over another

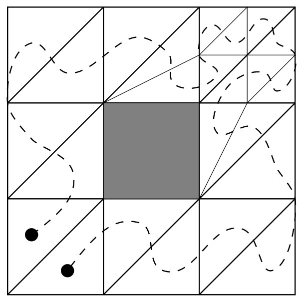

Figure 1. An example of an SAW.

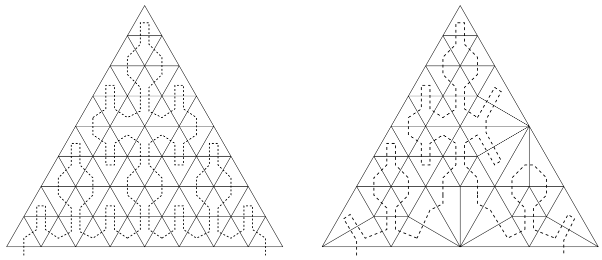

Figure 2. Space-filling curves for uniformly-refined and adaptively-refined triangulations.

edge or vertex. In the cases where the SAW jumps over vertices, we imply that the triangles following one another do not share an edge.

Intuitively, walks going only over edges show better locality. Since we want to stay away from the problem of Hamiltonicity, we make a weaker assumption about the behavior of the SAWs which however excludes cases of extreme non-locality. In the following, we consider a special class of SAWs which we call *proper self-avoiding walks* (PSAW). A PSAW is an SAW where for any three triangles following one another in the walk, we do not allow jumping twice over the same vertex. (The SAW shown in Fig. 1 is proper.) The precise statement of the problem, the overall machinery, and proofs for the proposition formulated below can be found in [4]. The reference also contains some performance results for a simple implementation of the underlying algorithm.

Proposition 2.1. *There exists a proper self-avoiding walk for an arbitrary triangular mesh $\mathcal{M}$ with prescribed initial and final triangles.* □

The proof of Proposition 2.1 consists of two parts. In the first part, we prove the existence of PSAWs (without boundary conditions). This is done by induction over the number of triangles in the mesh, and extends an existing PSAW over to a larger mesh. This step of the proof also provides a set of elementary rules which can be used to formulate an $\mathbf{O}(n \log n)$ algorithm for constructing PSAWs with a given initial triangle over arbitrary unstructured meshes. The value of such rules becomes apparent if practical questions such as the proper extension of existing "incomplete" walks are addressed (and most of the difficulty of the proof is in this part). In the second part of the proof, we show that there is an incomplete PSAW from the initial to the final triangle, and then use the techniques from the first part to complete it [4].

For the hierarchical refinement of unstructured grids, the construction of PSAWs can be considerably simplified. In [4], we gave a construction of PSAWs for such cases. It turns out that this process is equivalent to an intermediate stage of a modified Sierpiński curve [7] (see Fig. 2), if we interpret the (geometric) curve visiting a triangle twice as a jump over a vertex. Therefore, the PSAW "inherits" almost all of the nice locality properties of such a curve. As a consequence, in the case of hierarchical adaptation, we have to modify an existing walk only in the regions affected by the adaptation.

3. Linearization, Adaptation, and Partitioning

Consider the following application scenario: start with a pre-generated mesh which will be subject to dynamic and *hierarchical* coarsening and refinement, i.e., all triangles generated at runtime are contained within the triangles of the initial mesh. Within the partitions to be constructed, we would like to achieve good data locality, represented by a PSAW. On the other hand, a runtime deformation of partition boundaries should not require a complete rebuilding of the linearization. Although PSAWs exist for arbitrary meshes (Proposition 2.1), some "guidance" or at least a few "warning signs" on the mesh would be helpful. Greedyu partitioners tend to ignore these "warning signs", occasionally resulting in disconnected partitions and a sensitive dependence on the starting point. The *level* approach behind greedy partitioning, taken from the Cuthill-McKee algorithm [2, 3], tends to fail in regions where the mesh or the physical domain undergoes structural changes (e.g., sudden shrinking or expansion, or the presence of corners).

In the next section, we briefly describe our novel boundary contour coloring strategy which provides some important information about the mesh. We assume that the underlying physical domain has a non-empty boundary.

3.1. Boundary Contour Coloring

We call a mapping $\mathcal{M} \to \mathbb{N}$, which assigns to each triangle of $\mathcal{M}$ a natural number, a *coloring*. Given a mesh $\mathcal{M}$, we consider the coloring that can be obtained by iteratively applying the following rules:

- All triangles which share an edge or a vertex with the boundary of $\mathcal{M}$ have color 0.
- All uncolored triangles which share an edge or a vertex with a triangle of color i have color $i+1$.

We call this coloring a *boundary contour coloring*. Interpreting color as height, the boundary contour coloring of a mesh gives a mountainous terrain over the mesh with (local) maxima and saddle points. Figure 3 shows an example of a mesh containing 1286 triangles (thanks to J. Shewchuck for his Triangle tool), that produces a sufficiently interesting boundary contour coloring.

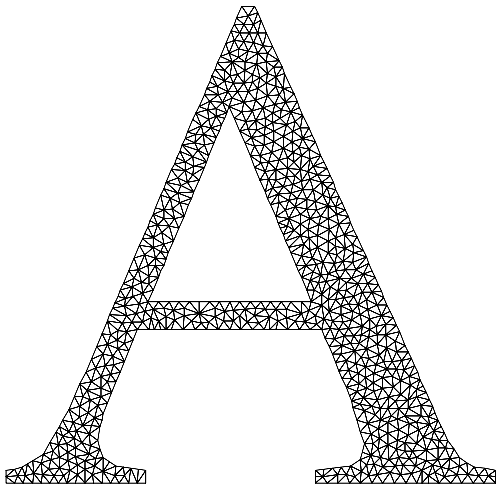

Figure 3. The letter A mesh.

Unfortunately, a black-and-white document does not effectively reproduce the coloring. We thus describe the coloring in words. The boundary contour coloring makes the mesh appear as separated into leafs or layers which can be "peeled off". The coloring of the mesh in Fig. 3 contains five different layers. The left leg of the A mesh is thinner and has colors 0 and 1, except for a cluster of ten triangles of color 2 in the lower left. The color 1 layer branches near the middle bar into two parts, one going further up and the other going to the right. The thicker right leg has four layers of colors 0, 1, 2, and 3, as well as a cluster of three triangles of color 4 in the lower right. All the layers up to and including color 2 are connected whereas the layer of color 3, as the ridge of a mountain, is disconnected (three components). The contour of color 1 branching from the left joins up with the corresponding contour from the right, where it shows the same branching behavior as the one on the left.

Denote by T_c the set of triangles of color c. In general, the color contour of T_c does not have a proper interior in the following sense: for a triangle $t \in T_c,\ 0 < c < c_{\max}$, at least one of the three neighbors sharing an edge with t usually has a different color while the other two are of the same color. If all three neighbors have the same color as t, it indicates a local maximum, a saddle point, or an "untying" or "branching" of contours. Only the last possibility is critical, as we shall see below. The color foliation has a nontrivial structure, but apart from the branches, looks like a rectangular region with colored fibers. If we wish to exploit this structure, we will have to determine the boundaries of such regions.

The boundary contour coloring induces an edge coloring as follows:

- An edge on the domain boundary or on the boundary between two color layers is assigned the color -1.
- An edge whose two adjacent triangles are of the same color i is assigned the color i.

Thus, edges of non-negative color cut through the boundary contour coloring in a "transverse" manner.

So far, we have only considered walks consisting of triangles. In the following, we consider walks of edges. A sequence of edges $\eta_i,\ 1 \le i \le k$ is called a walk, if every two edges following one another in the walk share a vertex and no edge appears more than once in the walk. A walk (of edges) $\eta = (\eta_i)_{1 \le i \le k}$ is called a *section* if it has the following properties:

- All edges in η have non-negative color, except for those corresponding to triangles of (locally) maximal color having three adjacent (over an edge) triangles of lower color.
- The first and last edges are of color 0.
- The color of the edges in the walk is strictly increasing or decreasing, except for at most two adjacent edges of maximal color.

The first property ensures that sections cut transversally through the color layers and allows isolated maxima to be bypassed. The second property states that sections are "anchored" to the domain boundary, while the last property gives the walk the shape of an arc (color = height) and excludes certain pathological cases. Sections model the behavior of levels in the Cuthill-McKee algorithm. This is obvious for structured grids. For unstructured grids, sections have the advantage that, via the coloring, they are aware of structural changes in the mesh, and avoid the unpleasant divergence and separation of the levels. We state this more precisely in the next section.

3.2. Deformations and Critical Sections

Let η_a, η_b be two sections. We call η_b an *elementary deformation* of η_a (and vice-versa) if the following holds:

- There exist two triangles on the domain boundary such that the initial and final edges of both η_a and η_b are adjacent to them.
- Two vertices where η_a and η_b cut through the same boundary between two color contours are either identical or separated by an edge of color -1.

The first requirement ensures that an elementary deformation of a section is not too far away from the given section. The second requirement controls the "smoothness" of the transition. Two sections η_a and η_b are called *deformations* of one another if they can be deformed into each other by a sequence of elementary deformations. In other words, deformation is an equivalence relation on the set of sections. From an algorithmic point of view, this definition appears rather unsatisfactory since it might be hard to determine in practice if two sections can be deformed into each other. However, proposition 3.1 shows that this issue can be resolved for practical purposes. The mathematical theory was established about 60 years ago, and served as our primary motivation.

We call a section *critical* if it has non-trivial deformations to one side only. (The meaning of "side" is obvious for simply connected domains. We save the reader from the rather lengthy discussion for the non-simply connected case.)

3.2.1. An Aside to Morse Theory

We refer the reader to [6, 8] for an introduction to Morse theory. Consider the following real-valued function on the Torus T^2. Let $f : T^2 \to \mathbb{R}$ be the height above the plane V (see Fig. 4). A point $x_0 \in T^2$ is called a *critical point* of f if the gradient of f vanishes at x_0. $f(x_0)$ is then called a *critical value*. Points p, q, r, and s in Fig. 4 are critical. One of the central results in Morse theory can be stated as follows:

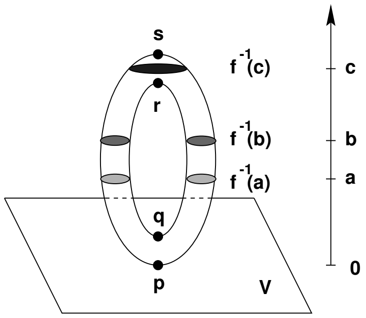

Figure 4. Critical Points.

Theorem 3.1. *Let f be a smooth real-valued function on a manifold M. Let $a < b$ and suppose that the set $f^{-1}[a, b]$, consisting of all $p \in M$ with $a \le f(p) \le b$, is compact and contains no critical points of f. Then, $f^{-1}(a)$ and $f^{-1}(b)$ are isomorphic.* □

In other words, given a Morse function, its critical points serve as "sentinels" for structural changes. Apart from these sentinels, undisturbed deformation is possible.

Returning to our discussion in Sec. 3.2, we would like to have a result similar to Theorem 3.1 which would allow us to determine if a region bounded by sections is free of critical sections that are deformations of one of the bounding sections. Consider a simply connected region R of the mesh whose boundary consists of two disconnected components of the boundary and two (disjoint) sections. If there is no critical section between the two sections, which is a deformation of one of them, they can be deformed into each other. We can prove the following sufficient condition for a region to have this property.

Proposition 3.1. *Let R be a region as described above. Assume that there are no triangles in a connected contour of R of non-maximal color which have three neighbors (adjacent over an edge) of the same color. Furthermore, assume that if a triangle t in R is of (locally) maximal color and two neighbors of smaller color are adjacent with t over an edge, then they belong to different contours of R. Under these assumptions, there are no critical sections in R which are deformations of one of the sections on the boundary of R.* □

The assumptions in Proposition 3.1 ensure that contours of nonmaximal color do not have an interior (which they might have in the neighborhood of branches or nodes of very high degree), and that clusters of maximal color have a smooth boundary, rather than be fragmented. Note that the assumptions of Proposition 3.1 are sufficient, not necessary. Thus, there might be regions which do not satisfy these assumptions, but the assertion is still true. These conditions can be easily verified, which makes them applicable for practical purposes. They are satisfied, except for two triangles (which can be fixed easily), by the mesh in Fig. 3 for the relevant nine regions (see Figure 5). As a conclusion of Proposition 3.1, we find that a region reaches it maximal extension if both of the bounding sections turn into a critical section.

Unfortunately, we are not experts in mesh generation. However, we strongly believe that there is an intimate connection between Morse functions and what is considered to be an adequate mesh for a given physical domain.

3.3. The Partitioning Problem

Based on our previous discussions, a mesh can be *decomposed* into foliated regions, leaving its core and rather small critical regions where the foliations undergo structural changes. To a large extent, the success of modern graph partitioners relies on a multilevel approach, their ability to contract or coarsen a given graph into smaller graph(s) and propagate partitioning information from coarser levels back to the initial problem. With our decomposition approach we "virtually" accomplish something similar. Figure 5 shows a

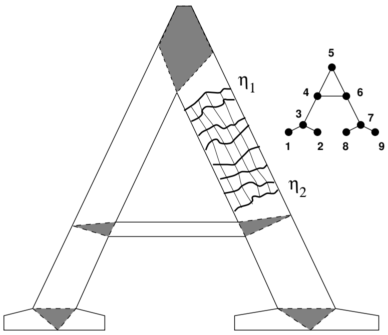

Figure 5. Critical regions of the A mesh.

schematic representation of the decomposition of the letter A mesh, and a problem graph for the partitioning problem.

The white regions are bounded by critical sections which can be freely deformed inside the region. The shaded regions mark areas where the boundary contour coloring undergoes essential changes and the structure is not as simple as that of the white regions. With each critical region, we associate a vertex weighted by the number of triangles in it. This region should be considered to be rather complex and stiff, and we want to avoid those regions in the process of determining partition boundaries. With each white region we associate an edge, weighted in the same manner by the number of triangles in it.

A *partitioning problem* can now be formulated as follows: find n sections, with n depending on the connectedness of the domain (holes?) and the number of partitions desired, such that each partition contains about the same number of triangles and the total length of the sections is minimal.

The reduced A mesh graph is two-connected. For two partitions, we would need two sections in the mesh. Obviously, the sections would be made in the regions corresponding to the edges (4,6) and (5,6). For sections, we generally have good control over their length. The length of the two sections for the A mesh would be 3 or 4 and 9 or 10, respectively, with about 640 triangles in each partition.

The proposed partitioning method can be applied in a recursive manner. In this process, a critical section usually remains critical. On the one hand, this simplifies the task of finding critical sections, but on the other hand, imposes certain requirements for mesh generation: if the mesh practically consists of critical regions only, this method is going to fail.

The practical (parallel) search for sections is based on the heuristic that sufficiently deformable sections are likely to be found in the neighborhood of local maxima or saddle points of the boundary contour coloring. Notice, that we do not have to search for critical sections: we find them naturally in the process of determining the "elasticity" of a region.

4. Summary

In this paper, we proposed a new approach to runtime partitioning of adaptive unstructured grids. Its inherent parallelism stems from a natural decompositon of the underlying mesh. A comprehensive comparison with multilevel diffusion techniques needs to be performed in order to better understand the strengths and weaknesses of the method. Furthermore, we must achieve a better comprehension of the implications for mesh generation.

Acknowledgements

This work is supported by NASA under Contract Numbers NAS 2-14303 with MRJ Technology Solutions and partially by NSF under Grant Numbers NSF-CISE-9726388 and NSF-MIPS-9707125.

The work of the first and third authors, was performed for the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the National Security Agency (NSA) through an agreement with the National Aeronautics and Space Administration.

References

[1] R. Biswas and L. Oliker. Experiments with repartitioning and load balancing adaptive meshes. Technical Report NAS-97-021, NASA Ames Research Center, 1997.

[2] U. Elsner. Graph partitioning – a survey. Technical Report SFB393/97-27, SFB393, Technische Universität Chemnitz, 1997.

[3] N. E. Gibbs, W. G. Poole, and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Analysis*, 13:236–250, 1976.

[4] G. Heber, R. Biswas, and G. R. Gao. Self-avoiding walks over two-dimensional adaptive unstructured grids. Technical Report NAS-98-007, NASA Ames Research Center, 1998.

[5] G. Heber, R. Biswas, and G. R. Gao. Using multithreading for the automatic load balancing of adaptive finite element meshes. In H. D. Simon, editor, *Proceedings 5th Symp. on Solving Irregularly Structured Problems in Parallel*, volume LNCS 1457, NERSC, LBNL Berkley, California, 1998. Springer Verlag.

[6] J. Milnor. *Morse Theory*. Annals of Mathematics Studies 51. Princeton University Press, 1963.

[7] H. Sagan. *Space-Filling Curves*. Universitext. Springer Verlag, 1994.

[8] H. Seifert and W. Threlfall. *The calculus of variations in the large*. (in German) published in the USA by Chelsea, New York, 1951.