Self-Avoiding Walks over Adaptive Unstructured Grids

Gerd Heber[1], Rupak Biswas[2], and Guang R. Gao[1]

[1] CAPSL, University of Delaware, 140 Evans Hall, Newark, DE 19716, USA
{heber,ggao}@capsl.udel.edu
http://www.capsl.udel.edu

[2] MRJ/NASA Ames Research Center, MS T27A-1, Moffett Field, CA 94035, USA
rbiswas@nas.nasa.gov
http://science.nas.nasa.gov/~rbiswas

Abstract. In this paper, we present self-avoiding walks as a novel technique to "linearize" a triangular mesh. Unlike space-filling curves which are based on a geometric embedding, our algorithm is combinatorial since it uses the mesh connectivity only. We also show how the concept can be easily modified for adaptive grids that are generated in a hierarchical manner based on a set of simple rules, and made amenable for efficient parallelization. The proposed approach should be very useful in the runtime partitioning and load balancing of adaptive unstructured grids.

1 Introduction

Advances in adaptive software and methodology notwithstanding, parallel computational strategies will be an essential ingredient in solving complex, real-life problems. However, parallel computers are easily programmed with regular data structures; so the development of efficient parallel adaptive algorithms for unstructured grids poses a serious challenge. An efficient parallelization of these unstructured adaptive methods is rather difficult, primarily due to the load imbalance created by the dynamically-changing nonuniform grid and the irregular data access pattern. Nonetheless, it is generally believed that unstructured adaptive-grid techniques will constitute a significant fraction of future high-performance supercomputing. Mesh adaptation and dynamic load balancing must therefore be accomplished rapidly and efficiently, so as not to cause a significant overhead to the numerical simulation.

Serialization techniques play an important role when using a finite element method (FEM) over adaptive unstructured grids. A numbering of the unknowns allows for a matrix-vector notation of the underlying algebraic equations. Special numbering techniques (Cuthill-McKee, frontal methods) have been developed to optimize memory usage and locality of the algorithms. In addition, the runtime support for decomposing dynamic adaptive grids is often based on a linear representation of the grid hierarchy in the form of a *space-filling curve*. Several researchers have demonstrated the successful application of techniques based on space-filling curves to N-body simulations, graph partitioning, and other graph-related problems [4, 6–9].

The general idea of a space-filling curve is a serialization (or linearization) of visiting points in a higher-dimensional space. A standard method for the construction is to embed the object under study into a regular environment (where the standard space-filling curves live). However, this approach introduces an artificial structure in that the entire construction depends on the embedding. Furthermore, this type of a linear representation ignores the combinatorial structure of the mesh which drives the formulation of operators between the finite element spaces. The question arises whether one could reduce the different requirements for a serialization in an adaptive FEM over unstructured grids to a common denominator.

In this paper, we present a new approach to constructing a *self-avoiding walk* through a triangular mesh. Unlike space-filling curves which are based on a geometric embedding, our algorithm is combinatorial since it uses the mesh connectivity only. We also show how the concept can be easily modified for adaptive grids (that are generated in a hierarchical manner based on a set of simple rules) and made amenable for efficient parallelization (using a novel indexing scheme).

2 Self-Avoiding Walks

A self-avoiding walk (SAW) over an arbitrary unstructured two-dimensional mesh $\mathcal{M}$ is an enumeration of all the triangles of $\mathcal{M}$ such that two successive triangles share an edge or a vertex. Note that a SAW visits each triangle exactly once, entering it over an edge or a vertex, and exiting over another edge or vertex. In the cases where the SAW jumps over vertices, we imply that the triangles following one another in the enumeration do not share an edge. As simple examples show, the requirement that triangles following one another in a SAW must share an edge is too strong. In general, SAWs do not exist under such conditions (see Fig. 1 for a simple example where jumping over a vertex is necessary to visit every triangle). It is important to note that a SAW is not a Hamiltonian path.

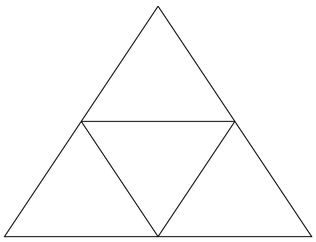


Fig. 1. A simple example that shows the need for SAWs to jump over vertices

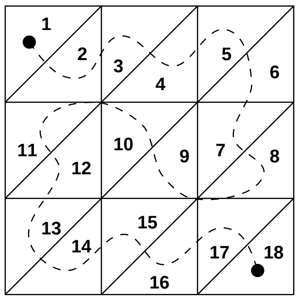


Fig. 2. An example of a PSAW over a triangular mesh

SAWs can be used to improve the parallel efficiency of several irregular algorithms, in particular issues related to locality and load balancing. However, we need special classes of SAWs that will facilitate dynamic load balancing with good locality in terms of runtime memory access and interprocessor communication. We thus consider a special class of SAWs called *proper self-avoiding walks* (PSAW) where jumping twice over the same vertex is forbidden (see Fig. 2 for an example).

2.1 Definitions

Let us first fix some common notation. For a SAW ω over a triangular mesh $\mathcal{M}$, we denote the i-th triangle in ω by $\omega(i)$ and the length of ω by $\#\omega$. Note that $\#\omega$ is also the number of triangles in $\mathcal{M}$. Formally, the properness of a SAW can be stated as follows:

Definition 1. *A self-avoiding walk (SAW) ω is called* <u>proper</u> $\Longleftrightarrow_{\text{def}}$

$$\omega(i-1) \cap \omega(i) \neq \omega(i) \cap \omega(i+1) \qquad \forall i \in \{2, \ldots, \#\omega - 1\}. \tag{1}$$

If two triangles t_1, t_2 share an edge, we write $t_1 \mid t_2$. If $\omega(i) \mid \omega(i+1)$, we write $\omega(i) \vdash \omega(i+1)$ to indicate that ω enters $\omega(i+1)$ from $\omega(i)$ over an edge. If $\omega(i) \nmid \omega(i+1)$, we write $\omega(i) \curvearrowright \omega(i+1)$ to indicate that ω jumps into $\omega(i+1)$ from $\omega(i)$ over a vertex.

In the following, we make use of two technical results.

Lemma 1. *Let t, t_1, t_2, t_3 be triangles in a mesh $\mathcal{M}$ and*

$$t_i \mid t \qquad \forall i \in \{1, 2, 3\}. \tag{2}$$

Then, either

$$t_1 \cap t_2 \cap t_3 = \emptyset \tag{3}$$

or $\mathcal{M}$ is isomorphic to a tetrahedron. $\square$

Note that if two triangles in a mesh share two vertices, they *must* share the corresponding edge.

Lemma 2. *Let $\mathcal{M}$ be a triangular mesh. Then, there exists a triangle τ in $\mathcal{M}$ such that $\mathcal{M} - \tau$, the complex obtained by removing τ from $\mathcal{M}$, is still a triangular mesh.* $\square$

2.2 Existence of Proper Self-Avoiding Walks

We can prove that a PSAW exists for any arbitrary triangular mesh $\mathcal{M}$. The proof is inductive over the number of triangles in the mesh and extends an existing PSAW over to a larger mesh.

Proposition 1. *There exists a proper self-avoiding walk (PSAW) for an arbitrary triangular mesh $\mathcal{M}$.*

Proof. Assume that PSAWs exist for meshes with n triangles. Let $\mathcal{M}$ be a mesh with $n+1$ triangles.

Let τ be a triangle in $\mathcal{M}$ such that $\mathcal{M}-\tau$, the mesh consisting of all the triangles in $\mathcal{M}$ except τ, is a mesh (see Lemma 2). This mesh has n triangles and hence, by our induction assumption, a PSAW $\omega_{\mathcal{M}-\tau}$ exists for $\mathcal{M}-\tau$. We show that $\omega_{\mathcal{M}-\tau}$ can be extended to another PSAW $\omega_{\mathcal{M}}$ for $\mathcal{M}$. We call $\omega_{\mathcal{M}}$ a *proper extension* of $\omega_{\mathcal{M}-\tau}$.

Let t be an triangle of $\mathcal{M}-\tau$ sharing an edge with τ (i.e. $t \mid \tau$ holds). For an appropriate $i \in \{1,\ldots,n\}$, there holds

$$t = \omega_{\mathcal{M}-\tau}(i). \tag{4}$$

The discussion can be naturally split up into four cases, depending on whether $\omega_{\mathcal{M}-\tau}$ enters t through an edge or a vertex and leaves through an edge or a vertex, respectively.

We omit the subscript $\mathcal{M}-\tau$ from $\omega_{\mathcal{M}-\tau}$ in the following for the sake of clarity. Also, if we do not know whether the transition from $\omega(i)$ to $\omega(i+1)$ goes over an edge or a vertex, we write $\omega(i) \to \omega(i+1)$.

In the figures below, parts of $\omega_{\mathcal{M}-\tau}$ are drawn as solid lines whereas the modifications leading to $\omega_{\mathcal{M}}$ are drawn as dashed lines. Triangles are indexed by their position in the walk (e.g. $\omega(i)$ is denoted by i).

Case I: $\boxed{\omega(i-1) \vdash \omega(i) \vdash \omega(i+1)}$

Figure 3 illustrates the modifications necessary if $\omega_{\mathcal{M}-\tau}$ enters and leaves t through edges.

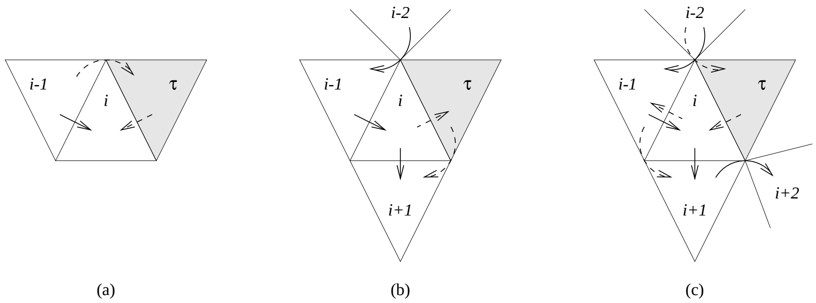


Fig. 3. Walk entering and exiting the triangle adjacent to τ via edges

- If $\omega(i-1) \cap \tau \neq \omega(i-2) \cap \omega(i-1)$, then $\omega(i-1) \to \tau \vdash \omega(i)$ is a proper extension of ω (Fig. 3(a)).
- If $\omega(i-1) \cap \tau = \omega(i-2) \cap \omega(i-1)$, then $\omega(i-2) \curvearrowright \omega(i-1)$.
  - If $\omega(i+1) \cap \tau \neq \omega(i+1) \cap \omega(i+2)$, then $\omega(i) \vdash \tau \to \omega(i+1)$ is a proper extension of ω (Fig. 3(b)).
  - If $\omega(i+1) \cap \tau = \omega(i+1) \cap \omega(i+2)$, then $\omega(i+1) \curvearrowright \omega(i+2)$.
    - If $\omega(i-1) \cap \omega(i+1) \neq \omega(i+1) \cap \omega(i+2)$, then $\omega(i-2) \to \tau \vdash \omega(i) \vdash \omega(i-1) \to \omega(i+1)$ is a proper extension of ω (Fig. 3(c)).

* Let $\omega(i-1) \cap \omega(i+1) = \omega(i+1) \cap \omega(i+2)$. According to Lemma 1, either $\mathcal{M}$ is tetrahedral or $\omega(i-1) \cap \omega(i+1) \cap \tau = \emptyset$. If $\mathcal{M}$ is tetrahedral, there is nothing to prove. For the other case, we have $\emptyset = \omega(i-1) \cap \omega(i+1) \cap \tau = \omega(i+1) \cap \omega(i+2) \cap \tau = \omega(i+1) \cap \tau = \omega(i+1) \cap \omega(i+2)$. This is contradicts our assumption $\omega(i+1) \curvearrowright \omega(i+2)$ which means $\omega(i+1) \cap \omega(i+2) \neq \emptyset$. Therefore, if $\mathcal{M}$ is not tetrahedral, the case $\omega(i-1) \cap \omega(i+1) = \omega(i+1) \cap \omega(i+2)$ is not possible.

Case II: $\boxed{\omega(i-1) \vdash \omega(i) \curvearrowright \omega(i+1)}$

Figure 4 illustrates the modifications necessary if $\omega_{\mathcal{M}-\tau}$ enters t through an edge and leaves it through a vertex.

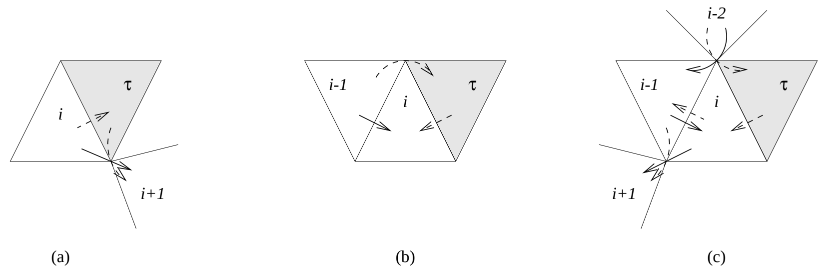


Fig. 4. Walk entering the triangle adjacent to τ via an edge and exiting via a vertex

- If $\omega(i) \cap \omega(i+1) \subset \omega(i) \cap \tau$, then $\omega(i) \vdash \tau \rightarrow \omega(i+1)$ is a proper extension of ω (Fig. 4(a)).
- Suppose $\omega(i) \cap \omega(i+1) \notin \omega(i) \cap \tau$. (Note that $\omega(i) \cap \omega(i+1)$ definitely consists of a single vertex!)
  - If $\omega(i-1) \cap \tau \neq \omega(i-2) \cap \omega(i-1)$, then $\omega(i-1) \rightarrow \tau \vdash \omega(i)$ is a proper extension of ω (Fig. 4(b)).
  - If $\omega(i-1) \cap \tau = \omega(i-2) \cap \omega(i-1)$, then $\omega(i-2) \curvearrowright \omega(i-1)$. In that case, $\omega(i-2) \rightarrow \tau \vdash \omega(i) \vdash \omega(i-1) \rightarrow \omega(i+1)$ is a proper extension of ω (Fig. 4(c)).

Case III: $\boxed{\omega(i-1) \curvearrowright \omega(i) \vdash \omega(i+1)}$

Figure 5 illustrates the modifications necessary if $\omega_{\mathcal{M}-\tau}$ enters t through a vertex and leaves it through an edge.

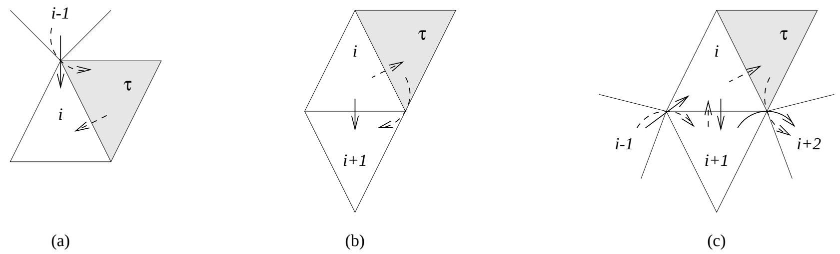
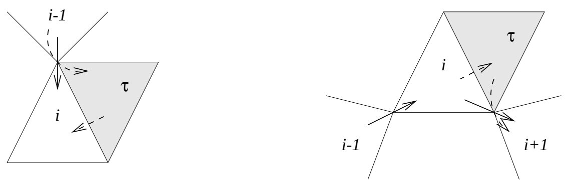


Fig. 5. Walk entering the triangle adjacent to τ via a vertex and exiting via an edge

- If $\omega(i-1) \cap \omega(i) \subset \omega(i) \cap \tau$, then $\omega(i-1) \to \tau \vdash \omega(i)$ is a proper extension of ω (Fig. 5(a)).
- Suppose $\omega(i-1) \cap \omega(i) \not\subset \omega(i) \cap \tau$.
  - If $\omega(i+1) \cap \tau \neq \omega(i+1) \cap \omega(i+2)$, then $\omega(i) \vdash \tau \to \omega(i+1)$ is a proper extension of ω (Fig. 5(b)).
  - If $\omega(i+1) \cap \tau = \omega(i+1) \cap \omega(i+2)$, then $\omega(i+1) \curvearrowright \omega(i+2)$. In that case, $\omega(i-1) \to \omega(i+1) \vdash \omega(i) \vdash \tau \to \omega(i+2)$ is a proper extension of ω (Fig. 5(c)).

Case IV: $\boxed{\omega(i-1) \curvearrowright \omega(i) \curvearrowright \omega(i+1)}$

Figure 6 illustrates the modifications necessary if $\omega_{\mathcal{M}-\tau}$ enters and leaves t through vertices.



(a) (b)

Fig. 6. Walk entering and exiting the triangle adjacent to τ via vertices

- If $\omega(i-1) \cap \omega(i) \subset \omega(i) \cap \tau$, then $\omega(i-1) \to \tau \vdash \omega(i)$ is a proper extension of ω (Fig. 6(a)).
- If $\omega(i-1) \cap \omega(i) \not\subset \omega(i) \cap \tau$, then $\omega(i) \vdash \tau \to \omega(i+1)$ is a proper extension of ω (Fig. 6(b)).

Since we have been able to properly extend $\omega_{\mathcal{M}-\tau}$ in all cases, our proposition is proved. □

A closer look at the proof of Proposition 1 and an auxiliary construction allow us to prove the following stronger result:

Proposition 2. *Let $\mathcal{M}$ a triangular mesh and $t_1, t_2 \in \mathcal{M}$ be two arbitrary triangles. Then, there exists a proper self-avoiding walk (PSAW) ω with $\omega(1) = t_1$ and $\omega(\#\mathcal{M}) = t_2$.* □

The proof of Proposition 1 actually provides a set of elementary rules which can be used to construct a PSAW: starting from any random triangle of $\mathcal{M}$, choose new triangles sharing an edge with the current submesh and extend the existing PSAW over the new triangle. In the process of deriving the existence of PSAWs, a natural mathematical framework emerges which can be easily adapted for other special classes of SAWs.

The complexity of this algorithm is $O(n \log n)$, where n is the number of triangles in the mesh. This complexity is obtained as follows. Since the triangles are organized in a red-black tree, a search requires $O(\log n)$ time (the edge-triangle incidence relation can also be organized in a red-black tree). Since a search has to be performed for each triangle in the mesh, the overall complexity of our algorithm is $O(n \log n)$.

We have implemented a sequential algorithm for the generation of a PSAW using the `set` and `map` containers from the C++ Standard Template Library. Both containers are implemented with red-black trees; however, one may also think about an implementation using hash tables. The results presented in Tables 1 and 2 were obtained for some sample meshes on a 110 MHz microSPARC II using GNU's `g++` compiler (version 2.7.2.3) with the `-O` flag. Note that the times in the tables below include the setup time for the mesh and the edge-triangle incidence structure.

Table 1. Runtimes for various mesh sizes

Triangles	*Edges*	*Time* (s)
2048	3136	0.30
4096	6240	0.57
8192	12416	1.31
16384	24768	2.71
32768	49408	5.54
65536	98688	11.82
131072	197120	24.07

Table 2. Runtimes for various triangle/edge ratios

Triangles	*Edges*	*Ratio*	*Time* (s)
65536	131073	0.500	12.85
65536	114690	0.571	12.27
65536	106500	0.615	12.21
65536	102408	0.640	12.15
65536	100368	0.653	12.13
65536	99360	0.660	12.02
65536	98880	0.663	11.99
65536	98688	0.664	11.82

The results in Table 1 indicate the dependence of the runtime on the number of triangles (or edges). According to our description of the algorithm, one would suspect that the runtime may also depend on the triangle/edge ratio. The results in Table 2 explore this possibility; however, the numbers clearly show that the dependence of the execution time on the triangle/edge ratio is negligible and confirm our complexity estimate for the algorithm.

A standard parallelization of the deterministic construction algorithm would be based on the observation that if the extensions of an existing PSAW did not interfere with one another, then they could be done in parallel. Such an approach is feasible in dealing with single meshes when no additional structural information is available, or with families of meshes without any hierarchical structure, although scalability would be a major concern.

3 Adaptive Triangular Grids

As is evident from the discussion in the previous section, a parallel algorithm for constructing a PSAW is non-trivial, and the walk has to be completely rebuilt after each mesh adaptation. However, PSAWs provide enough flexibility to design a *local* algorithm for *hierarchical* adaptive changes in an unstructured mesh. In contrast to a general adaptation scheme that produces just another unstructured mesh, a hierarchical strategy pays particular attention to the initial triangulation and uses a set of simple coarsening and refinement rules:

- When a triangle is refined, it is either subdivided into two (green) or four (red) smaller triangles (Fig. 7).
- All triangles with exactly two bisected edges have their third edge also bisected; thus, such triangles are red.
- A green triangle cannot be further subdivided; instead, the previous subdivision is discarded and red subdivision is first applied to the ancestor triangle (Fig. 8).

Such a policy also assures that, with repeated refinement, the quality of the adapted mesh will not deteriorate drastically from that of the initial mesh. This strategy is similar to that described in [1] for tetrahedral meshes.

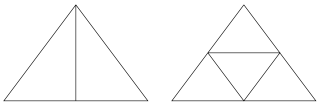


Fig. 7. Green (left) and red (right) refinement of a triangle

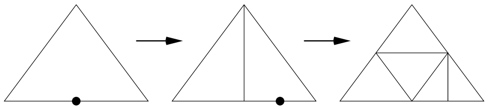


Fig. 8. Special policy for refining a green triangle

Such a scheme exploits the structure of the adaptation hierarchy to simplify the mesh labeling process. Our goal is to modify (or adapt) the labeling only in regions where the mesh has been altered but not to rebuild the entire indexing. This indexing idea has been extensively developed in [3]. It is a combination of coarse and local schemes. The coarse scheme labels the objects of the initial mesh so that the incidence relations can be easily derived from the labels. The local scheme exploits the regularity and the finiteness of the refinement rules to produce names for the mesh objects at subsequent refinement levels. The coarse and local schemes are combined by taking the union of the Cartesian products of the coarse mesh objects with their corresponding local schemes. Ambiguities are resolved by using a normal form of the index.

The key features of this scheme are:

- Each object is assigned a global name that is independent of any architectural considerations or implementation choices.
- Combinatorial information is translated into simple arithmetic.
- It is well-behaved under adaptive refinement and no artificial synchronization/serialization is introduced.
- It can be extended (with appropriate modifications) to three dimensions [2].

Our basic strategy for such hierarchical adaptive meshes is to apply the general algorithm only once to the initial mesh to construct a coarse PSAW, and then to reuse the walk after each adaptation. This requirement leads us to the concept of *constrained proper self-avoiding walks* (CPSAW) that exploits the regularity of the refinement rules. In a way, a PSAW is global since it is related to the initial coarse mesh, while a CPSAW is local since it is restricted to a triangle of the coarse mesh. A PSAW leaves a trace or footprint on each coarse triangle which is then translated into a boundary value problem for extending the walk over the adapted mesh. In addition, the extensions for different coarse triangles can be decoupled; thus, the construction process of a CPSAW is inherently parallel. Formal mathematical definitions of a constraint and the compatibility of PSAWs with constraints are given in [5].

Figure 9 depicts an example of how a CPSAW is generated from an existing PSAW when the underlying mesh is adapted. Figure 9(a) shows a portion of a PSAW over four triangles of an initial coarse mesh. The entry and exit points are respectively marked as α and β. These four triangles are then refined as shown in Fig. 9(b). The entry and exit constraints for each of the four original triangles are shown in Fig. 9(c) (an exploded view is used for clarity). Figure 9(d) shows the extension of the original PSAW to the higher level of refinement. Note that it is a direct extension dictated by the constraints. This CPSAW can often be improved by replacing certain jumps over vertices by jumps over edges when adjacent triangles in the walk share an edge (Fig. 9(e)).

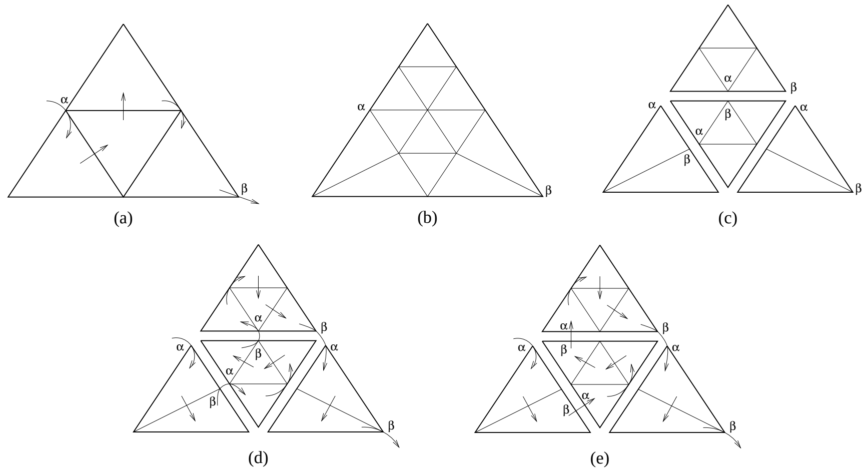


Fig. 9. Extending a PSAW to a CPSAW over an adapted mesh

The regularity of the local picture allows us to prove the existence of CP-SAWs; full details are given in [5]. Moreover, CPSAWs are also appropriate for multigrid methods since they "know" the refinement level of the adapted mesh. The process of generating a CPSAW from an initial PSAW after mesh adaptation is embarrassingly parallel, since each processor is responsible for only its own chunk of the walk. However, load balancing is still an important issue that must be addressed when the coarse mesh is locally adapted.

4 Conclusions

In this paper, we have developed a theoretical basis for the study of self-avoiding walks over two-dimensional adaptive unstructured grids. We described an algorithm of complexity $O(n \log n)$ for the construction of proper self-avoiding walks over arbitrary triangular meshes and reported results of a sequential implementation. We discussed parallelization issues and suggested a significant improvement for hierarchical adaptive unstructured grids. For this situation, we have also proved the existence of constrained proper self-avoiding walks.

The algorithms presented allow a straightforward generalization to tetrahedral meshes; however, there is some flexibility because of additional freedom in three dimensions. The application of this methodology to partitioning and load balancing holds a lot of promise and is currently being investigated. However, modifications to improve locality in the sense of a generalized frontal method requires a more detailed study of the relationships between more specialized constraints on walks and cache memory behavior.

Note that our basic algorithm can be improved in many ways. For instance, we completely ignored additional information like nodal degrees. Also, given the existence of PSAWs, an acceleration technique might be favorable. For example, walks have a long tradition in the application of Monte Carlo methods to study long-chain polymer molecules. An investigation of these alternate approaches is beyond the scope of this paper but could be the focus of future work.

Acknowledgements

The work of the first and third authors was partially supported by NSF under Grant Numbers NSF-CISE-9726388 and NSF-MIPS-9707125. The work of the second author was supported by NASA under Contract Number NAS 2-14303 with MRJ Technology Solutions.

References

1. Biswas, R., Strawn, R.C.: Mesh Quality Control for Multiply-Refined Tetrahedral Grids. Appl. Numer. Math. **20** (1996) 337–348
2. Gerlach, J.: Application of Natural Indexing to Adaptive Multilevel Methods for Linear Triangular Elements. RWCP Technical Report 97-010 (1997)
3. Gerlach, J., Heber, G.: Fundamentals of Natural Indexing for Simplex Finite Elements in Two and Three Dimensions. RWCP Technical Report 97-008 (1997)
4. Griebel, M., Zumbusch, G.: Hash-Storage Techniques for Adaptive Multilevel Solvers and their Domain Decomposition Parallelization. AMS Contemp. Math. Series **218** (1998) 279–286
5. Heber, G., Biswas, R., Gao, G.R.: Self-Avoiding Walks over Two-Dimensional Adaptive Unstructured Grids. NAS Technical Report NAS-98-007 (1998)
6. Ou, C.-W., Ranka, S., Fox, G.: Fast and Parallel Mapping Algorithms for Irregular Problems. J. of Supercomp. **10** (1995) 119–140
7. Parashar, M., Browne, J.C.: On Partitioning Dynamic Adaptive Grid Hierarchies. 29th Hawaii Intl. Conf. on System Sciences (1996) 604–613
8. Pilkington, J.R., Baden, S.B.: Dynamic Partitioning of Non-Uniform Structured Workloads with Spacefilling Curves. IEEE Trans. Par. Dist. Sys. **7** (1996) 288–300
9. Salmon, J., Warren, M.S.: Parallel, Out-of-Core Methods for Fast Evaluation of Long-Range Interactions. 8th SIAM Conf. on Par. Proc. for Sci. Comput. (1997)