AUTOMATIC MESH PARTITIONING

GARY L. MILLER *, SHANG-HUA TENG †, WILLIAM THURSTON ‡, AND STEPHEN A. VAVASIS §

Abstract This paper describes an efficient approach to partitioning unstructured meshes that occur naturally in the finite element and finite difference methods. This approach makes use of the underlying geometric structure of a given mesh and finds a provably good partition in random $O(n)$ time. It applies to meshes in both two and three dimensions. The new method has applications in efficient sequential and parallel algorithms for large-scale problems in scientific computing. This is an overview paper written with emphasis on the algorithmic aspects of the approach. Many detailed proofs can be found in companion papers.

Keywords: Center points, domain decomposition, finite element and finite difference meshes, geometric sampling, mesh partitioning, nested dissection, radon points, overlap graphs, separators, stereographic projections.

1. Introduction. Many large-scale problems in scientific computing are based on unstructured meshes in two or three dimensions. Examples of such meshes are the underlying graphs of finite volume methods in computational fluid dynamics or graphs of the finite element and finite difference methods in structural analysis. These meshes may have millions of nodes. Quite often the mesh sizes used are determined by the memory available on the machine rather than the physics of the problem to be solved. Thus, the larger the memory the larger the mesh used and, hopefully, the better the simulation produced.

The main goal of this paper is to describe our work on how and under what conditions unstructured meshes will have partitions into two roughly equal sized pieces with a small boundary (called small separators to be defined later). When these partitions exist they have several important applications to the finite element and finite difference methods. We list some of them here.

One approach to achieving the large memory and computation power requirements for large-scale computational problems is to use massively parallel distributed-memory machines. In such an approach, the underlying computational mesh is divided into submeshes, inducing a subproblem to be stored on each processor in the parallel system and boundary information to communicated [67]. To fully utilize a massively parallel machine, we need a subdivision in which subproblems have approximately equal size and the amount of communication between subproblems is relatively small. This approach will decrease the time spent per iteration. There are also methods

* School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. Supported in part by National Science Foundation grant CCR-9016641.

† Xerox Corporation, Palo Alto Research Center, Palo Alto, CA 94304. Part of the work was done while the author was at Carnegie Mellon University. Current address: Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139.

‡ Department of Mathematics, University of California, Berkeley CA 94720.

§ Department of Computer Science, Cornell University, Ithaca, NY 14853. Supported by an NSF Presidential Young Investigator award. Revision work on this paper was supported by the Applied Mathematical Sciences program of the U.S. Department of Energy under contract DE-AC04-76DP00789 while the author was visiting Sandia National Laboratories.

which use good partitioning to either decrease the number of iterations used or the time used by direct methods.

Several numerical techniques have been developed using the partitioning method to solve problems on a parallel system. Examples include *domain decomposition* and *nested dissection.* Domain decomposition divides the nodes among processors of a parallel computer. An iterative method is formulated that allows each processor to operate independently. See Bramble, Pasciak and Schatz [11], Chan and Resasco [13], and Bjørstad and Widlund [9]. *Nested dissection* is a divide-and-conquer node ordering for sparse Gaussian elimination, proposed by George [34] and generalized by George and Liu [36] and Lipton, Rose and Tarjan [49]. Nested dissection was originally a sequential algorithm, pivoting on a single element at a time, but it is an attractive parallel ordering as well because it produces blocks of pivots that can be eliminated independently in parallel. Parallel nested dissection was suggested by Birkhoff and George [8] and has been implemented in several settings [12, 21, 35, 84]; its complexity was analyzed by Liu [52] (for the regular square grid) and Pan and Reif [63] (in the general case).

Vaidya has produced results which indicate that the quality of good preconditioners may also be linked to the existence of good partitions [78].

Therefore, one of the key problems in solving large-scale computational problems on a parallel machine is the question of how to partition the underlying meshes in order to reduce the total communication cost and to achieve load balance.

If a mesh has a sufficiently regular structure, then it is easy to decide in advance how to distribute it among the processors of a parallel machine. However, meshes of many applications are irregular and unstructured, making the partition problem much more difficult. In general, there are meshes in three dimensions which have no small partition [59]. These examples are not the type that would naturally arise in the finite element methods, but they are meshes. One important goal is to understand which meshes do and which do not have small partitions.

Various heuristics have been developed and implemented [65, 68, 82]. However, none of the prior mesh partitioning algorithms is both efficient in practice and provably good, especially for meshes from three dimensional problems. Leighton and Rao [46] have designed a partitioning algorithm based on multicommodity flow problems, which finds a separator that is optimal within logarithmic factors. But their algorithm runs in superlinear time and it remains to be seen if it could be used in practice for large-scale problems.

1.1. A new method. In a series of papers, the authors (Vavasis [81]; Miller and Thurston [59]; Miller and Vavasis [60]; Miller and Teng [55]; Miller, Teng, and Vavasis [56]) have developed an efficient and provably good mesh partitioning method. This overview paper describes this new approach. It is written with emphasis on the algorithmic aspects of the approach. Many detailed proofs can be found in companion papers [57, 58].

This method applies to meshes in both two and three dimensions. It is based on the following important observation: graphs from large-scale problems in scientific computing are often defined geometrically. They are meshes of elements in a fixed

dimension (typically two and three dimensions), that are *well shaped* in some sense, such as having elements of bounded aspect ratio or having elements with angles that are not too small. In other words, they are graphs embedded in two or three dimensions that come with natural geometric coordinates and with structures.

Our approach makes use of the underlying geometric structure of a given mesh and finds a provably good partition efficiently. The main ingredient of this approach is a novel geometrical characterization of graphs embedded in a fixed dimension that have a small *separator*, which is a relatively small subset of vertices whose removal divides the rest of the graph into two pieces of approximately equal size. By taking advantage of the underlying geometric structure, we also develop an efficient algorithm for finding such a small separator.

In contrast, all previous separator results (see Section 1.2) are combinatorial in nature. They not only characterize the small separator property combinatorially, but also find a small separator based only on the combinatorial structure of the given graph. When applied to unstructured geometric meshes, they simply discard the valuable geometric information. The result has been that they are either too costly to use or they do not find a separator as good as it should be. Worst of all, none of the earlier separator results is useful for graphs in three dimensions.

1.2. Separators and earlier work. DEFINITION 1.1 (SEPARATORS). *A subset of vertices C of a graph G with n vertices is an* $f(n)$**-separator** *that* δ**-splits** *if $|C| \leq f(n)$ and the vertices of $G - C$ can be partitioned into two sets A and B such that there are no edges from A to B, $|A|, |B| \leq \delta n$, where f is a function and $0 < \delta < 1$.*

Two of the most well-known families of graphs that have small separators are trees and planar graphs. Every tree has a single vertex separator that 2/3-splits [44]. Lipton and Tarjan [50] proved that every planar graph has a $\sqrt{8n}$-separator that 2/3-splits. Their result improved an earlier one by Ungar [77]. Some extensions of their work have been made [19, 31, 32, 54], and separator theorems have also been obtained for graphs with bounded genus [38, 43] and graphs with bounded excluded minor [2]. In particular, Gilbert, Hutchinson, and Tarjan showed that all graphs with genus bounded by g have an $O(\sqrt{gn})$-separator, and Alon, Seymour, and Thomas proved that all graphs with an excluded minor isomorphic to the h-clique have an $O(h^{3/2}\sqrt{n})$-separator.

Interestingly, all the characterizations above are combinatorial, not geometric, as are their proofs!

Separator results for families of graphs closed under the subgraph operation immediately lead to divide-and-conquer recursive algorithms for many applications. In general, the efficiency of such algorithms depends on δ being bounded away from 1 and $f(n)$ being a slowly-growing function.

Perhaps the most classical application of small separator results is nested dissection, a widely used technique for solving a large class of sparse linear systems. This approach was pioneered by George [34], who designed the first $O(n^{1.5})$-time nested dissection algorithm for linear systems on regular grids using the fact that the $\sqrt{n} \times \sqrt{n}$ grid has a $\sqrt{n}$-separator. His result was extended to planar linear systems by Lip-

ton, Rose, and Tarjan [49]. Gilbert and Tarjan [40] examined several variants of the nested dissection algorithms. It has been demonstrated, in theory and in practice, that nested dissection can be implemented efficiently in parallel.

In the analysis of sparse matrix algorithms, *a priori* upper bounds on operation counts are rare in the literature (aside from the trivial dense-matrix upper bounds). The major exception is nested dissection. The *a priori* bounds attained by nested dissection, which in many cases are asymptotically the best possible, always depend on the associated bounds of the underlying graph-separator algorithm. This means that a careful analysis of separator sizes is an important aspect of nested dissection.

Small separator results have found fruitful applications in VLSI design (Leiserson [47]; Leighton [45]; Valiant [79]) and efficient message routing (Fredrickson and Janardan [28]). They have also been used in proving several complexity-theoretic results (Paterson [62]; Lipton and Tarjan [51]), and have been used to design efficient graph algorithms such as parallel construction of breadth-first-search trees (Pan and Reif [63]), testing graph isomorphism (Gazit [33]), and approximating NP-complete problems (Lipton and Tarjan [51]).

1.3. Outline of the paper. Section 2 defines a new class of geometric graphs, the overlap graphs, and describes our main separator theorem. This class has a simple definition and contains many important classes of graphs as special cases. Section 3 studies meshes from the finite element and finite difference methods. We show that overlap graphs include "well shaped" meshes. We also show that planar graphs are a special case of overlap graphs in two dimensions. Section 4 presents a partitioning algorithm for overlap graphs. The algorithm first uses the geometric information of the input graph to find a "continuous" separator, then uses the combinatorial structure to compute a "discrete" separator from its continuous counterpart. The central step of the algorithm is to find a *center point* of a point set in a fixed dimensions, where a center point is a point such that every hyperplane passing through it about evenly divides the point set. We show that center points always exist and can be computed in polynomial time using linear programming. We also show that the step of computing a "discrete" separator from a continuous one can be performed in linear time. Section 5 introduces *geometric sampling*, a technique that reduces the problem size and simultaneously guarantees a provably good approximation of the larger problem. Using geometric sampling, we can compute an approximate center point in random constant time and find a "good" separator of an overlap graph in random linear time. We further give a practical heuristic for approximating a center point. We then extend the partitioning algorithm for unstructured meshes. Section 6 gives the proof outline of the main separator theorem. We demonstrate how to use geometric arguments to prove separator properties for graphs embedded in fixed dimension. Section 7 summarizes the paper and gives open questions.

2. Neighborhood systems and overlap graphs. Our geometric characterization of graphs that have small separators is based on the following elementary concept.

2.1. Neighborhood systems. DEFINITION 2.1. *Let $P = \{p_1, \ldots, p_n\}$ be points in $\mathbb{R}^d$. A* k**-ply neighborhood system** *for P is a set, $\{B_1, \ldots, B_n\}$, of closed balls*

such that (1) B_i is centered at p_i and (2) no point $p \in \mathbb{R}^d$ is strictly interior to more than k balls from B.

A 3-ply neighborhood system in two dimensions is illustrated in Figure 1.

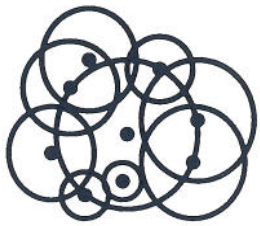

FIG. 1. *A 3-ply neighborhood system*

The following notation will be used throughout this paper. For each positive real α, if B is a ball of radius r in $\mathbb{R}^d$, then $\alpha \cdot B$ denotes the ball with the same center as B but radius αr.

We now state an important property of neighborhood systems [58].

LEMMA 2.2 (BALL INTERSECTION). *Suppose $\{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. For each d-dimensional ball B with radius r, for all constant $\beta : 0 < \beta \leq 1$,*

$$|\{i : B_i \cap B \neq \emptyset \text{ and } r_i \geq \beta r\}| \leq \beta^{-d} 3^d k,$$

where r_i is the radius of B_i.

2.2. Overlap graphs. DEFINITION 2.3. *Let $\alpha \geq 1$ and let $\{B_1, \ldots, B_n\}$ be a k-ply neighborhood system for $P = \{p_1, \ldots, p_n\}$. The (α, k)-**overlap graph** for the k-ply neighborhood system $\{B_1, \ldots, B_n\}$ is the undirected graph with vertices $V = \{1, \ldots, n\}$ and edges*

$$E = \{(i,j) : (B_i \cap (\alpha \cdot B_j) \neq \emptyset) \textbf{ and } ((\alpha \cdot B_i) \cap B_j \neq \emptyset)\}.$$

For simplicity, we call a $(1,k)$-overlap graph a *k-intersection graph.* In the case that $\alpha = 1$ and $k = 1$, and no two balls in the neighborhood system have a common point in their interior, we have the family of graphs known as *sphere-packings*; this interesting class of graphs will be discussed in the next section.

2.3. Main separator theorem. THEOREM 2.4 (MAIN). *Let G be an (α, k)-overlap graph for some fixed d. Then G has an*

$$O\left(\alpha \cdot k^{\frac{1}{d}} \cdot n^{\frac{(d-1)}{d}} + q(\alpha, k, d)\right) \textit{-separator}$$

that $(d+1)/(d+2)$-splits. Furthermore, such a separator that $(d+1+\epsilon)/(d+2)$-splits can be computed in random linear time sequentially and in random constant time, using n processors, for any $1/n^{1/2d} < \epsilon < 1$.

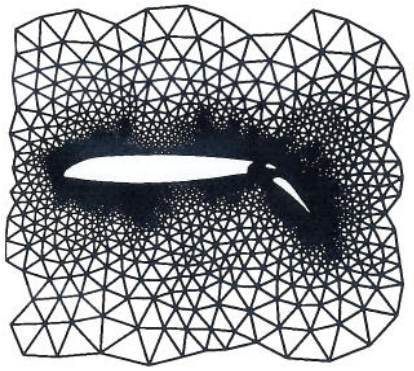

FIG. 2. *Airplane wing (Barth and Jespersen)*

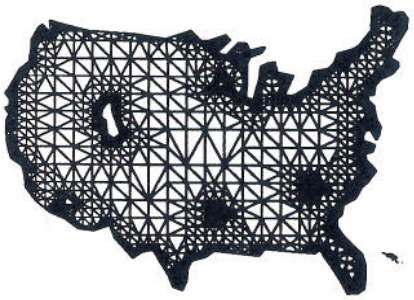

FIG. 3. *US map*

The function $q(\alpha, k, d)$ depends exponentially on d but is independent of n. Since the interesting cases are when $d = 2$ or $d = 3$ and when n is large, this term should be considered low order.

It has been shown in the companion paper [58] that the bound of Theorem 2.4 is tight up to a constant factor. Section 6 will outline the proof of this theorem.

3. Finite element and finite difference meshes. One important aspect that distinguishes a finite element or finite difference mesh from a regular graph is that it has two structures: the combinatorial structure and the geometric structure. In general, it can be represented by a pair (G, xyz) where G describes the combinatorial structure of the mesh and xyz gives the geometric information.

3.1. Meshes from the finite element method. The finite element method is a collection of numerical techniques for approximating a continuous problem by a finite structure [69]. To approximate a continuous function, the finite element method subdivides the domain (a subset of $\mathbb{R}^d$) into a *mesh* of polyhedral *elements* (Figures 2 and 3), and then approximates the continuous function by a piecewise polynomial on the elements.

A common choice for an element in the finite element method is a d-dimensional *simplex*, which is the convex hull of $(d+1)$ affinely independent points in $\mathbb{R}^d$, e.g., a triangle in two dimensions and a tetrahedron in three dimensions. A d-dimensional *simplicial complex* is defined to be a collection of d-dimensional simplices that meet only at shared faces [6, 7, 59]. So a 2-dimensional simplicial complex is a collection of triangles that intersect only at shared edges and vertices.

For most applications, a mesh is given as a list of its elements, where each element is given by the information describing the hierarchical structure of the elements, its lower dimensional structures such as its faces, edges, and vertices. Moreover, each vertex has geometric coordinates in two or three dimensions.

Associated with each simplicial complex is a natural graph, its 1-skeleton. For example, the 1-skeleton of a 2-dimensional simplicial complex is a planar graph. Conversely, every planar graph can be embedded in the plane such that each edge is mapped to a straight line segment (Fáry [25]; Tutte [74, 75]; Thomassen [71]; Fraysseix, Pach, and Pollack [27]).

In the finite element method, a linear system is defined over a mesh, with variables representing physical quantities at the nodes. Let *finite element graph* refer to the nonzero structure of the coefficient matrix of such a linear system. In the case of linear finite elements based on a triangulation, such as in Figures 2 and 3, the nodes of the finite element graph are exactly the nodes of the mesh, and hence the finite element graph is the same as the 1-skeleton of the simplicial complex. In the case of higher-order elements, the finite element graph usually contains the 1-skeleton as a proper subset. It can be obtained from the finite element mesh as follows: Identify certain points (vertices, points on edges, points in faces, and points in elements) as "nodes." Add edges between every pair of nodes that share an element.

To properly approximate a continuous function, in addition to the conditions that a mesh must conform to the boundaries of the region and be fine enough, each individual element of the mesh must be *well shaped.* A common shape criterion for elements is the condition that the angles of each element are not too small, or the aspect ratio of each element is bounded [6, 29].

Several definitions of the *aspect ratio* have been used in literature. We list some of them.

1. The ratio of the longest dimension to the shortest dimension of the simplex S, denoted by $A_1(S)$. For a triangle in $\mathbb{R}^2$, it is the ratio of the longest side divided by the altitude from the longest side.
2. The ratio of the radius of the smallest containing sphere to the radius of the inscribed sphere of S, denoted by $A_2(S)$.
3. The ratio of the radius of the circumscribing sphere to the radius of the

FIG. 4. *Meshes derived from quadtrees (S. Mitchell)*

inscribed sphere of S, denoted by $A_3(S)$.

4. The ratio of the diameter to the dth root of the volume of the simplex S, denoted by $A_4(S)$, where the *diameter* of a d-simplex S is the maximum distance between any pair of points in S.

Examples of a simplicial complex with bounded aspect ratio are illustrated in Figure 4 as well as in Figures 2 and 3. The above definitions of the aspect ratio are polynomially related to each other. They are also closely related to the *smallest angle* of the simplex, which is the smallest angle among all the angles between pairs of supporting hyperplanes of the simplex. Using elementary geometric arguments, one can prove the following set of inequalities: There are constants $c_1 < c_2$, c_3, $c_4 < c_5$, depending only on d, such that if the smallest angle of S is θ, then

$$\begin{aligned}
\frac{1}{|\sin\theta|} &\leq A_1(S) \leq \frac{2}{|\sin\theta|} \\
c_1 A_1(S) &\leq A_2(S) \leq c_2 A_1(S) \\
A_1(S) &\leq A_3(S) \leq c_3 (A_1(S))^2 \\
c_4 A_1(S) &\leq (A_4(S))^d \leq c_5 (A_1(S))^{d-1}.
\end{aligned}$$

Therefore, if one of the above parameters is bounded by a constant, then all of them are bounded.

3.2. Graphs from the finite difference method. The finite difference method is another useful technique for solving computational problems in scientific computing. It also uses a finite and discrete structure, *a finite difference mesh*, to approximate a continuous problem.

Finite difference meshes are often produced by inserting a uniform grid of $\mathbb{R}^2$ or $\mathbb{R}^3$ into the domain via a boundary-matching conformal mapping. In general, the derivative of the conformal transformation must be slowly varying with respect to the mesh size in order to produce good results. See, for example [72]. This means that the mesh will probably satisfy a *density condition* [5, 60].

Let G be an undirected graph and let π be an embedding of its nodes in $\mathbb{R}^d$. We say π is an embedding of *density* α if the following inequality holds for all vertices v in G. Let u be the closest node to v. Let w be the farthest node from v that is

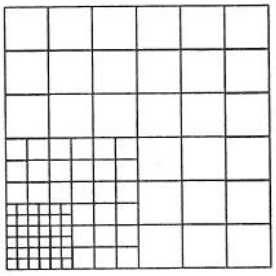

FIG. 5. *Berger and Bokhari's example of a density graph.*

connected to v by an edge. Then

$$\frac{||\pi(w) - \pi(v)||}{||\pi(u) - \pi(v)||} \leq \alpha.$$

In general, G is an *α-density graph* in $\mathbb{R}^d$ if there exist an embedding of G in $\mathbb{R}^d$ with density α. It can be easily shown that there is a $\Delta(\alpha, d)$ depending only on α and d such that the maximum degree of an α-density graph is bounded by $\Delta(\alpha, d)$.

Furthermore, a finite difference mesh may not be a collection of simplices or elements as a finite element mesh, so we can not analyze it as a triangulation. Finite difference meshes are often locally refined by further subdividing some mesh cells. See Figure 5. This means that nodes occur on the sides of some mesh cells and that interpolation must be used in the finite difference approximation. For numerical accuracy of the interpolation, the usual practice is that mesh cells are refined to a level no more than a constant factor smaller than their neighboring cells [5]. In the presence of such refinement, the finite difference mesh will still satisfy a constant density condition.

3.3. Overlap graphs and well shaped meshes. One of the most valuable aspects of the class of overlap graphs is that it enables us to give a unified geometric characterization of graphs with the small separator property. The set of overlap graphs in $\mathbb{R}^d$ contains all finite subgraphs of infinite grids, planar graphs and sphere packing graphs. Moreover, overlap graphs include graphs associated with finite element and finite difference methods, as special cases. The parameter α, in a strong sense, measures the degree to which the mesh is well-shaped.

We now show that for each well-shaped mesh, there is an overlap graph with a pair α and k that contains the graph defined by (G, xyz) as a subgraph. We say a graph G_1 is a *spanning subgraph* of another graph G_2 if G_1 can be obtained from G_2 by deleting edges. A graph G is *(α, k)-embeddable* in $\mathbb{R}^d$ if it is a spanning subgraph of an (α, k)-overlap graph in $\mathbb{R}^d$. Notice that the small separator property is preserved under spanning subgraphs.

LEMMA 3.1. *If G is an α-density graph in $\mathbb{R}^d$, then G is $(2\alpha, 1)$-embeddable.*

Proof: Let π be an embedding of G with density α in $\mathbb{R}^d$. Without loss of generality, assume that G has vertex set $V = \{1, 2, ..., n\}$. Let $P = \{\pi(1), \pi(2), ..., \pi(n)\}$. For each $p \in P$, let $c(p)$ denote the point of $P - \{p\}$ closest to p. Let $\Gamma = \{B_1, \ldots, B_n\}$,

where for each $i : 1 \leq i \leq n$, B_i is a ball centered at $\pi(i)$, whose radius r_i is $0.5||c(p) - p||$. Clearly, balls from Γ do not intersect each other. We claim that G is a subgraph of the 2α-overlap graph G' of Γ.

For each edge (u, v) of P, we need to show that $(\pi(u), \pi(v))$ is an edge of G'. Without loss of generality, assume $r_u \leq r_v$. Because π is an α-embedding of G, we have

$$\frac{||\pi(u) - \pi(v)||}{2r_u} \leq \alpha.$$

So $\pi(v) \in (2\alpha) \cdot B_u$, and therefore $(\pi(u), \pi(v))$ is an edge of G', completing the proof. □

LEMMA 3.2. *Suppose G is the 1-skeleton of a simplicial complex K in $\mathbb{R}^d$. Let $\tilde{G}$ be the subgraph of G obtained by removing all vertices that appear on the external boundary of K. Then if the aspect ratio of K is bounded by a constant α, then there is a constant c depending only on d and α such that $\tilde{G}$ is $(c, 1)$-embeddable.*

Proof: Because the aspect ratio of the complex K is bounded by α, there is a Θ depending only on α and d such that the smallest angle of each simplex of K is at least Θ. Therefore, there is a constant Δ depending only on d and α, such that the degree of G is bounded by Δ. Furthermore, the bounded aspect ratio implies that the ratio of the longest edge over the shortest edge of any simplex in K is also bounded. Thus, there is a constant c_1 depending only on d and α, such that for each vertex v of K, the ratio of the longest edge to the shortest edge connecting v is bounded by c_1. Moreover, for each vertex v of K that is not on the external boundary, the closest vertex of v in K is also connected with v by an edge, because no vertex can appear in the interior of an simplex. Therefore, if we remove all vertices of K on the external boundary, we obtain a c_1-density graph $\tilde{G}$. Let $c = 2c_1$. It follows from Lemma 3.1 that $\tilde{G}$ is $(c, 1)$-embeddable. □

Therefore, the following theorem follows from Theorem 2.4.

THEOREM 3.3.

- *If G is the 1-skeleton of a simplicial complex K with bounded aspect ratio, letting $\tilde{n}$ be the number of exterior vertices of K, then G has an*

$$O\left(n^{\frac{(d-1)}{d}} + \tilde{n}\right) \text{-separator.}$$

- *If G is a graph with bounded density α, then G has an*

$$O\left(\alpha \cdot n^{\frac{(d-1)}{d}}\right) \text{-separator.}$$

Moreover, such separators that $(d + 1 + \epsilon)/(d + 2)$-split can be computed in random $O(n)$ time and in random $O(1)$ time using n processors, for any $1/n^{1/2d} < \epsilon < 1$.

An algorithm for partitioning finite element and difference meshes is given in Section 5.4.

3.4. Overlap graphs include all planar graphs. The proof that planar graphs are a special case of overlap graphs relies on the following theorem of Andreev and Thurston [3, 4, 73] characterizing all planar graphs in a novel geometric fashion.

THEOREM 3.4 (ANDREEV AND THURSTON). *Each triangulated planar graph G is isomorphic to a 2-dimensional sphere packing graph.*

Simply from the definition, each sphere packing graph is a $(1,1)$-overlap graph. Therefore, planar graphs are a special case of overlap graphs.

4. A randomized partitioning algorithm. In this section, we describe a randomized algorithm for computing a small separator of a given overlap graph. In the next section, we shall show how to make it efficient by using geometric sampling. In Section 6, we will outline a correctness proof of the algorithm to derive a constructive proof of Theorem 2.4.

4.1. The algorithm. Given a k-ply neighborhood system $\Gamma = \{B_1, \ldots, B_n\}$ and the α-overlap graph G of Γ, let $P = \{p_1, \ldots, p_n\}$ be the centers of Γ. The following algorithm first finds a $(d-1)$-sphere S with some desired properties to be stated later and then computes a vertex separator of G from S. In the following algorithm, let U_d be the unit d-sphere in $\mathbb{R}^{d+1}$. We define $ST : \mathbb{R}^d \to U_d$ to be the standard stereographic projection mapping. This mapping can be described as follows. Assume $\mathbb{R}^d$ is embedded in $\mathbb{R}^{d+1}$ as the $x_{d+1} = 0$ coordinate plane, and assume U_d is also embedded in $\mathbb{R}^{d+1}$ centered at the origin. Given a point p in $\mathbb{R}^d$, construct the line L in $\mathbb{R}^{d+1}$ passing through p and through the north pole of U_d (that is, the point $(0,..,0,1)$). Line L must pass through one other point q of U_d; we define $ST(p)$ to be q. For a set $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$, we denote $\{ST(p_1), ST(p_2), ..., ST(p_n)\}$ by $ST(P)$. Recall that the center point of a point set is the one such that every hyperplane passing through it about evenly divides the point set. We will define center point formally in Section 4.2.

Algorithm 1 (Generic Geometric Partitioning)
Input: (a neighborhood system Γ and the geometric coordinates of its centers P).

1. Compute $Q = ST(P)$;
2. Find a δ-center point c of Q;
3. Compute the rotation $\pi_1 : U_d \to U_d$ that maps c to c_1, a point on the diameter between the south and the north poles, say $c_1 = (0, ..., 0, r)$.
4. Let π_2 be the dilation of $\mathbb{R}^d$ by a factor of $\sqrt{(1-r)/(1+r)}$.
5. Choose a random great circle GC of U_d;
6. Transform GC back $\mathbb{R}^d$ using the inverse of the above transformations to obtain a separating sphere S, i.e., $S = [ST \circ \pi_2 \circ ST^{-1} \circ \pi_1 \circ ST]^{-1}(GC)$;
7. Compute a vertex separator of G from S.

Algorithm 1 defines some point sets that are not explicitly computed. We introduce them below only for the purpose of explaining the algorithm.

- Let $Q_1 = \pi_1(Q)$ in Step 3 above;
- Let $P_1 = ST^{-1}(Q_1)$, the pre-image of Q_1 in $\mathbb{R}^d \cup \{\infty\}$. The pre-image of the north pole is defined to be a point at infinity;
- Let $P_2 = \pi_2(P_1)$ in Step 4;

- Let $Q_2 = ST(P_2)$. Note that the origin $(0, 0, ..., 0)$ is a center point of Q_2. See further comments below.

4.2. Center points. Suppose P is a finite set of points in $\mathbb{R}^d$. A hyperplane H in $\mathbb{R}^d$ divides P into three subsets: $P^+ = H^+ \cap P$, $P^- = H^- \cap P$, and $P \cap H$. The *splitting ratio* of H over P, denoted by $\phi_H(P)$, is defined as

$$\phi_H(P) = \max\left(\frac{|P^+|}{|P|}, \frac{|P^-|}{|P|}\right)$$

For each $0 < \delta < 1$, a point $c \in \mathbb{R}^d$ is a δ-center point of P if every hyperplane containing c δ-splits P. Each $d/(d+1)$-center point is called a *center point* of P, and the set of all center points is denoted by Center(P). The balanced separation property of a center point makes it very useful for designing efficient divide and conquer algorithms [16, 30, 55, 83].

Given a set of points $P \subset \mathbb{R}^d$, the question of whether P has a center point is always affirmative. This follows from Helly's Theorem [18].

THEOREM 4.1 (HELLY). *Suppose $\mathcal{K}$ is a family of at least $d+1$ convex sets in $\mathbb{R}^d$, and $\mathcal{K}$ is finite or each member of $\mathcal{K}$ is compact. Then if each $d+1$ members of $\mathcal{K}$ have a common point, there is a point common to all members of $\mathcal{K}$.*

LEMMA 4.2 (CENTER POINTS). *For each set $P \subseteq \mathbb{R}^d$, Center(P) $\neq \emptyset$.*

Proof: [1] We prove the lemma by induction on d. When $d = 1$, the lemma is clearly true. We now assume that the lemma holds for all $d' < d$. If all points of P lie in a $(d-1)$-dimensional affine space, then we can reduce the dimension by one and apply the induction hypotheses to prove that a better center point exists.

So without loss of generality, assume that P does not lie in a $(d-1)$-dimensional affine space. Notice that P induces an equivalence relation on the set of closed halfspaces in $\mathbb{R}^d$: those halfspaces which contains the same subset of points from P are equivalent. Each equivalence class can be identified with a halfspace whose supporting hyperplane passes through d affinely independent points from P.

Let $\mathcal{H}$ be the set of all closed half-spaces with supporting hyperplane passing through d affinely independent points of P that contain more than $\lfloor d|P|/(d+1) \rfloor$ points of P. We want to show that

$$\text{Center}(P) = \bigcap_{H \in \mathcal{H}} H \neq \emptyset.$$

We first show that $\bigcap_{H \in \mathcal{H}} H \neq \emptyset$. Clearly, each element from $\mathcal{H}$ is convex and $\mathcal{H}$ is finite. By Helly's theorem, it is sufficient to show that for each $H_1, \ldots, H_{d+1} \in \mathcal{H}$, $\bigcap_{i=1}^{d+1} H_i \neq \emptyset$.

Note that

$$\bigcap_{i=1}^{d+1} H_i = \mathbb{R}^d - \bigcup_{i=1}^{d+1} (\mathbb{R}^d - H_i) \supseteq P - \bigcup_{i=1}^{d+1} (\mathbb{R}^d - H_i).$$

[1] We present this proof to indicate that there is an $O(n^d)$ time algorithm for computing a center point. Similar proofs can be found in many previous works, e.g., [18].

Note also

$$|\bigcup_{i=1}^{d+1}((\mathbb{R}^d - H_i) \cap P)| \leq \sum_{i=1}^{d+1}|((\mathbb{R}^d - H_i) \cap P)| < (d+1)\lfloor\frac{1}{d+1}|P|\rfloor < |P|.$$

Hence, $P - \bigcup_{i=1}^{d+1}(\mathbb{R}^d - H_i) \neq \emptyset$.

We now show that each point c in $\bigcap_{H\in\mathcal{H}} H$ is a center point of P. Suppose c is not a center point of P. Then there is a hyperplane h passing through c defining a halfspace H such that the interior of H contains at least $\lceil d|P|/(d+1)\rceil$ points of P. Thus, there is a closed halfspace H' contained in the interior of H that has at least $\lceil d|P|/(d+1)\rceil$ points of P, contradicting the assumption that $c \in H'$. Therefore, every point in $\bigcap_{H\in\mathcal{H}} H$ is a center point of P. Similarly, we can show that every center point of P is in $\bigcap_{H\in\mathcal{H}} H$. □

Immediately following from the above proof is an $O(n^d)$ time algorithm for computing a center point of a set P. This algorithm uses linear programming. It forms a collection of $O(n^d)$ linear inequalities by considering the set of hyperplanes passing through d affinely independent points of P, and finding the common intersection of the halfspaces that contain at least $dn/(d+1)$ points from P. The intersection of the $O(n^d)$ halfspaces can be found in $O(n^d)$ time using Megiddo's linear programming algorithm [22, 53]. We will refer this algorithm as the *LP algorithm*. Of course, this algorithm is too slow for applications in practice. An efficient algorithm will be presented in Section 6.

If c in Algorithm 1 is a δ-center point of Q, then the origin o is also a δ-center point of Q_2 [58]. First of all, the point c_1 is a δ-center point of Q_1. Now intuitively, a dilation of $\mathbb{R}^d$ moves a center point on the diameter between the south and the north poles along this diameter either up or down depending on the dilation factor. We will prove in our companion paper [58] that the dilation of by factor $\sqrt{(1-r)/(1+r)}$ indeed makes o a δ-center point of Q_2. So, any hyperplane passing through o δ-splits Q_2, and hence GC δ-splits Q_2. Because all transformations used in the above partitioning algorithm preserve the splitting ratio of spheres, S also δ-splits P.

4.3. Separating spheres. We now explain how to choose a random great circle in Algorithm 1.

A *great circle* of U_d is the intersection of U_d with a hyperplane passing through the center of U_d.

Let $randn(m)$ be a function that generates m normally distributed random numbers with mean 0.0 and variance 1.0. A random point p from U_d can be chosen as $p = q/||q||_2$, where $q = randn(d+1)$. A random great circle of U_d is then the great circle normal to the vector p.

Each $(d-1)$-sphere S separates $\text{int}(S)$ from $\text{ext}(S)$: any segment connecting a point in $\text{int}(S)$ and one in $\text{ext}(S)$ must intersect S. In analogy to vertex separators in graph theory, we say that S is called a *separating sphere* in d-space.

More specifically, for a set of points $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$ and a constant $0 < \delta < 1$, we say that S *δ-splits* P if both $|\text{int}(S) \cap P| \leq \delta n$ and $|\text{ext}(S) \cap P| \leq \delta n$.

4.4. Computing a vertex separator from a separating sphere. We now show how to compute a vertex separator of an overlap graph G from a separating sphere S.

One approach is to remove one of the two endpoints of each edge cut by S. We say S *cuts* an edge (B_i, B_j) of the overlap graph if the line segment between the centers of B_i and B_j has a common point with S. Let E_S be the set of edges cut by S. A ball B_i is a *boundary* ball with respect to S if there is an edge cut by S incident to it. Let U be the set of all boundary balls and let $G_S = (U, E_S)$ be the subgraph of G *induced* by S. Clearly, G_S is a bipartite graph, with boundary balls from the interior of S on one side and boundary balls from the exterior of S on the other side.

The discussion in this section assumes that no point of P lies exactly on S. Because we choose S at random in Algorithm 1, the occurrence of a point of P exactly on S is a zero-probability event. Even if this event were to occur, a slight generalization of the results in this section would cover that case. In fact, we can first put all points of P that appear exactly on S in the vertex separator.

Recall that a *vertex cover* of a graph G is a subset C of vertices such that each edge G has an endpoint in C. In other words, deleting C from G removes all edges of G. Simply from the definition of vertex cover, we have

LEMMA 4.3. *Suppose* $\Gamma = \{B_1, \ldots, B_n\}$ *is a* k*-ply neighborhood system in* $\mathbb{R}^d$ *and* G *is the* α*-overlap graph of* Γ. *If* S *is a* $(d-1)$*-sphere that* δ*-splits* Γ, *then each vertex cover* C *of* G_S δ*-splits* G.

Therefore, the best way to compute a vertex separator from S is perhaps to take a minimum vertex cover of G_S. In fact, a minimum vertex cover of a bipartite graph can be computed in polynomial time, using Dulmage-Mendelsohn decomposition [20] (see [64] for related applications of Dulmage-Mendelsohn decomposition in sparse matrix computations).

On the hand, a faster way to compute a vertex separator from S is to put all the boundary balls from either the interior of S or the exterior of S, whichever has smaller cardinality, into the separator (also see [84]).

However, in the above construction, one has to check the structure of an overlap graph to find a vertex separator from a sphere separator. In some applications, only the neighborhood system is given and it is relatively expensive to compute the overlap graph. We now show how to find a small vertex separator directly from the neighborhood system and the separating sphere.

For each edge (B_i, B_j) cut by S, let $q_{i,j}$ be the common point of S and line segment between the centers of B_i and B_j. Let r_i be the radius of B_i. Without loss of generality, assume $r_i \leq r_j$. Notice that $q_{i,j}$ is either in B_j or in $\alpha \cdot B_i$. If $q_{i,j}$ is in B_j, then we put B_j in D. If $q_{i,j}$ is not in B_j (in which case it must be in $\alpha \cdot B_i$), we put B_i in D. Clearly, since at least one endpoint of every cut edge is in D, D is a vertex cover of G_S.

We now introduce the notion of overlap neighbor. Let S be a $(d-1)$-sphere in $\mathbb{R}^d$, whose radius is r. A ball B_i is an *overlap neighbor* of S if one of the following conditions is true.

1. $B_i \cap S \neq \emptyset$;
2. $\alpha \cdot B_i \cap S \neq \emptyset$ and $r_i \leq r$.

The number of overlap neighbors of S is called the *overlap number* of S. The set of overlap neighbors of a sphere can be computed in $O(n)$ time directly from the neighborhood system.

LEMMA 4.4. *The set of all overlap neighbors of S is a vertex cover of G_S.*

Proof: From the discussion above, D is a vertex cover of G_S. We want to establish that each ball from D is an overlap neighbor of S to prove the lemma.

We partition the set D into two subsets D_1 and D_2, with

$$\begin{aligned} D_1 &= \{B_i \in D : B_i \cap S \neq \emptyset\} \\ D_2 &= D - D_1. \end{aligned}$$

Clearly, each ball from D_1 is an overlap neighbor. Now we need to show that for all $B_i \in D_2$, $r_i \leq r$.

If $B_i \in D_2$, then $B_i \cap S = \emptyset$. There are two possible cases:

- **Case 1:** If $p_i \in \text{int}(S)$, then it simply follows from $B_i \cap S = \emptyset$, that $r_i \leq r$;
- **Case 2**: If $p_i \in \text{ext}(S)$, then from $B_i \in D_2$, it follows that $\alpha \cdot B_i \cap S \neq \emptyset$, and there is a ball B_j in the neighborhood system such that (1) $p_j \in \text{int}(S)$; (2) $B_j \cap S = \emptyset$; (3) $r_i \leq r_j$; and (4) $\alpha \cdot B_i \cap B_j \neq \emptyset$. Because $q_{i,j}$ is not in B_j (otherwise B_i would not be in D) there is no intersection between B_j and S, and hence condition (2) holds. Hence $r_j \leq r$, and $r_i \leq r_j \leq r$.

Thus, in each case, we have $r_i \leq r$, i.e., B_i is an overlap neighbor, completing the proof of the lemma. □

So, our second method is to remove the set of all overlap neighbors. The method is efficient when the overlap graph G is not given. Section 6 shows that the expected number of overlap neighbors generated by the above algorithm is small.

Notice that the definition of overlap neighbors implicitly removes the assumption that no point of P lies exactly on S. If the center of B_i appears exactly on S, then $B_i \cap S \neq \emptyset$. Hence B_i is an overlap neighbor and is placed in the vertex separator.

5. Making the method practical. The run time of the algorithm above crucially depends on the time needed to compute a center point in $(d+1)$-space. All other steps of the algorithm can be performed in $O(n)$ time, and in constant parallel time using $O(n)$ processors.

Unfortunately, no linear-time algorithm is known for computing center points. As shown in Section 4.2, there is a method that requires solving a set of $\Theta(n^d)$ linear inequalities. The only improved result, due to Cole, Sharir, and Yap [16], is that a center point in two dimensions can be computed in $O(n \log^5 n)$ time, and in three dimensions in $O(n^2 \log^7 n)$ time. No subquadratic algorithm is known that always returns even an approximate center point.

In this section, we show that an approximate center point can be found efficiently

using geometric sampling [15, 42, 66], which is an important algorithmic technique for designing efficient geometric algorithms.

5.1. Geometric sampling for efficiency. To illustrate the idea, we first show how to use random sampling to compute an approximate center point in one dimension. In this case, the input is a set of $2n$ integers $P = \{p_1, \ldots, p_{2n}\}$. If $p_i < p_j$ for all $i < j$ then center$(P) = [p_n, p_{n+1}]$.

Now suppose we randomly select an element from P, say p. The probability that $p \in \{p_n, p_{n+1}\}$ is $1/n$, while the probability that $p_{\lceil n/2 \rceil} \leq p \leq p_{\lceil 3n/2 \rceil}$ is 0.5. So, with probability 0.5, a randomly selected element from P is an $\epsilon = 3/4$ center point.

We can improve ϵ using larger samples! Suppose l random elements $S = \{r_1, \ldots, r_l\}$ are selected and their median r is the output. Letting $I(r)$ be the rank of r in P, it follows from a simple analysis that $E[I(r)] = n$, and $V[I(r)] = (2n+1)(2n-1-2l)/(8k+6)$. By Chebyshev's inequality,

$$P(|I(r) - n|) \geq t) \leq \frac{n^2}{2lt^2}$$

Thus, with probability at least 0.5, $|I(r) - n| \leq n/\sqrt{l}$, i.e., r is a $1/2\sqrt{l}$ center point of $|P|$. A $1/2 + 1/2\sqrt{l}$ center point of $|P|$ can be computed in $O(l)$ time. A similar sampling idea was used by Floyd and Rivest [26] in their fast selection algorithm.

The algorithm can be generalized to higher dimensions. In d dimensions, the randomized δ-center point algorithm has the following form.

Algorithm 2: (The Sampling Algorithm for Center Points)
Input:(a point set $P \subset \mathbb{R}^d$)

1. Select a subset S of P with size l uniformly at random;
2. Compute a center point c_S of S, using the LP algorithm for center points, given in Section 4.2;
3. Output c_S.

The feasibility of Algorithm 2 above is specified in the following question: What is the probability that c_S computed above is an ϵ-center point?

We now introduce a notation which will be very useful in quantifying the quality of the c_S computed by Algorithm 2. Recall that $\phi_h(P)$ is the ratio in which hyperplane h splits a point set P.

DEFINITION 5.1 (ϵ-GOOD SAMPLE). *Suppose P is a set of points in $\mathbb{R}^d$. $S \subseteq P$ is an ϵ-good sample if for all hyperplanes h, $|\phi_h(S) - \phi_h(P)| \leq \epsilon$.*

The following lemma shows the importance of the ϵ-good sample in approximating center points. Its proof is straightforward.

LEMMA 5.2. *For each $P \subset \mathbb{R}^d$, if $S \subseteq P$ is an ϵ-good sample, then each δ-center point of S is a $(\delta + \epsilon)$-center point of P.*

Now the question becomes: how often does a set of l randomly chosen points form

an ϵ-good sample? This is not a trivial question, but was in fact answered by Vapnik and Chervonenkis [80] (see [70] for a detailed proof).

THEOREM 5.3 (VAPNIK AND CHERVONENKIS). *There is a constant c_d depending only on d such that for each $0 < \epsilon \leq 1$ and $l \geq 2/\epsilon^2$, if S is a set of l randomly chosen points from P, then*

$$Pr[S \text{ is an } \epsilon\text{-good sample}] \geq 1 - c_d l^{d+1} e^{\frac{-\epsilon^2 l}{8}}$$

The Vapnik-Chervonenkis bound implies that we only need to sample about $O(d \log d)$ points to compute an approximate center point with high probability.

THEOREM 5.4. *For all $P \in \mathbb{R}^d$, Algorithm 2 computes a $(\lambda_d + \epsilon)$-center point of P in*

$$O\left(\left[\frac{d}{\epsilon^2}\log\frac{d}{\epsilon} + \log\frac{1}{\eta}\right]^d\right)$$

time, with probability at least $1 - \eta$, where $\lambda_d = d/(d+1)$.

Notice that the computation above can be efficiently implemented in parallel.

5.2. Separators using sampling. We now incorporate random sampling into the partitioning algorithm for overlap graphs.

Algorithm 3 (Fast Geometric Partitioning)
Input: (a neighborhood system Γ and the geometric coordinates of its centers P).

1. Choose a random sample P' of size given by Theorem 5.4;
2. Let $Q = ST(P')$;
3. Compute a δ-center point c of Q using the LP algorithm;
4. Compute the rotation π_1 and the dilation π_2 that conformally map c to the origin;
5. Choose a random great circle GC of U_d;
6. Let $S = [ST \circ \pi_2 \circ ST^{-1} \circ \pi_1 \circ ST]^{-1}(GC)$;
7. Induce a vertex separator of G from S.

According to our experiments, about 800 points work very well for meshes in two dimensions, and 1100 points work very well for meshes in three dimensions.

THEOREM 5.5. *Algorithm 3 computes S in random constant time, and a vertex separator of an overlap graph in random $O(n)$ time. Using p processors, the time can be reduced to n/p.*

Algorithm 3 demonstrates the usefulness of geometry, sampling, and randomization in mesh partitioning. The random sampling in the above algorithm reduces the problem size and simultaneously guarantees a provably good approximation of the larger problem. It is the underlying geometric structure that ensures the quality of the partition.

5.3. A fast heuristic for center points. Although the sampling algorithm (Algorithm 2) for center point (in fixed dimensions) is efficient from theoretical viewpoint, it uses linear programming to solve the center point problem on a smaller sample point set. The use of linear programming becomes a serious concern in practical implementation. For example, the experimental results show that the sampling algorithm need to choose a sample of about five hundred to eight hundred points in two dimensions. The sampling algorithm thus needs to solve $\binom{500}{3} \approx 20$ million linear inequalities! Worse, the state-of-art linear programming algorithms (for fixed dimensions) have a large constant. The sample size would be larger for higher dimensions. The seemingly efficient sampling algorithm is too expensive for practical applications.

To overcome this difficulty, we have developed a heuristic for approximating center points [55]. The heuristic uses randomization and runs in linear time in the number of sample points. Most importantly, it does not use linear programming. Our algorithm is based on the notion of a radon point. Let P be a set of points in $\mathbb{R}^d$. A point $q \in \mathbb{R}^d$ is a *radon point* [18] if P can be partitioned into 2 disjoint subsets P_1 and P_2 such that q is a common point of the convex hull of P_1 and the convex hull of P_2. Such a partition is called a *radon partition.*

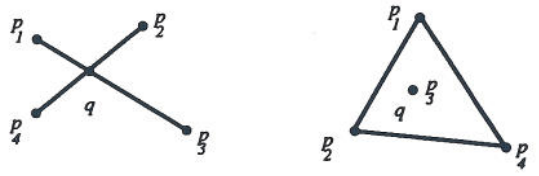

FIG. 6. *The radon point of four points in $\mathbb{R}^2$. When no point is in the convex hull of the other three (the left figure), then the radon point is the unique cross of two linear segments. Otherwise (the right figure), the point that is in the convex hull of the other three is a radon point.*

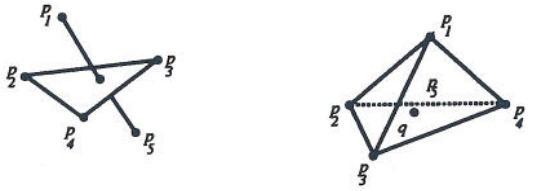

FIG. 7. *The radon point of five points in $\mathbb{R}^3$. Two cases are similar to those in two dimensions.*

The following theorem shows that if $|P| \geq d+2$, then a radon point always exists. Moreover, it can be computed efficiently.

THEOREM 5.6 (RADON [18]). *Let P be a set of points in $\mathbb{R}^d$. If $|P| \geq d+2$, then there is a partition (P_1, P_2) of P such that the convex hull of P_1 has a point in common with the convex hull of P_2.*

Proof: Suppose $P = \{p_1, \ldots, p_n\}$ with $n \geq d+2$. Consider the system of $d+1$ homogeneous linear equations

$$\sum_{i=1}^{n} \alpha_i = 0 = \sum_{i=1}^{n} \alpha_i p_i^j \qquad (1 \leq j \leq d),$$

where $p_i = (p_i^1, ..., p_i^d)$ are the usual coordinates of in $\mathbb{R}^d$. Since $n \geq d+2$, the system has a nontrivial solution $(\alpha_1, \ldots, \alpha_n)$. Let U be the set of all i for which $\alpha_i \geq 0$, and V the set for which $\alpha_i \leq 0$, and $c = \sum_{i \in U} \alpha_i > 0$. Then (U, V) is a partition of P, and $\sum_{i \in V} \alpha_i = -c$ and $\sum_{i \in U}(\alpha_i/c)p_i = \sum_{i \in V}(\alpha_i/c)p_i$. Let $q = \sum_{i \in U}(\alpha_i/c)p_i = \sum_{i \in V}(\alpha_i/c)p_i$. The point q is simultaneously written as a convex combination of points in U and a convex combination of points in V. Hence, q is in the convex hull of U and the convex hull of V, completing the proof. □

To compute a radon point of P, we need only to compute a radon point for the first $d+2$ points. It follows from the proof above that a radon point can be computed in $O(d^3)$ time.

We now describe our heuristic for approximating center points.

Algorithm 4: (Fast Center Points)
Input:(a point set $P \subset \mathbb{R}^d$)

1. Construct a complete balanced $(d+2)$-way tree T of L leaves (for an integer L);
2. For each leaf of T, choose a point from P uniformly at random, independent of other leaves;
3. Evaluate tree T in a bottom-up fashion to assign a point in $\mathbb{R}^d$ to each internal node of T such that the point of each internal node is a radon point of the points with its $(d+2)$ children;
4. Output the point associated with the root of T.

A complete $(d+2)$-way tree of L leaves has at most $L/(d+2)$ internal nodes. Algorithm 4 takes $O(d^2 L)$ time, with a small constant. Our experimental results suggest that, independent of the size of the mesh, $L = 900$ is sufficient for meshes from two dimensions and $L = 1200$ for meshes from three dimensions. Moreover, about 10 to 30 tries give a small cost separator that approximately 0.52-splits a mesh.

On the theoretical side, recently, Eppstein, Miller, Sturtivant, and Teng [23] gave a proof that Algorithm 4 finds a $(1 - 1/d^2)$-center point with high probability.

5.4. A practical algorithm for partitioning unstructured meshes. A finite element mesh is not given by a neighborhood system or an overlap graph. Fortunately, our partitioning algorithms do not require such a neighborhood system representation; only the proof does. To cope with the new setting, we show how to adapt our partitioning algorithm.

Algorithm 5 (Partitioning Finite Element Meshes)
Input: (the combinatorial structure of a mesh G, and the geometric coordinates of the mesh xyz).

1. Choose a random sample P of size given by Theorem 5.4 from xyz;
2. Apply Algorithm 3 (Fast Geometric Partitioning) to compute a separating sphere S for P; In practice, we replace the LP algorithm of Step 3 in Algorithm 3 by Algorithm 4, the Radon partition based fast center point algorithm.
3. Partition G into two subgraphs G_1 and G_2 by removing the set of edges $\tilde{E}$ that cut this sphere. This set of edges can be found using the structure of G; each vertex is placed into G_1 or G_2, depending on whether it is mapped to the interior or the exterior of S;
4. Compute the density α of the embedding given by (G, xyz). Then, for each v that is an endpoint of an edge in $\tilde{E}$, find the ball centered at v whose radius is given in the proof of Lemma 3.1. Use the rule of overlap neighbors of Section 4.4 to compute a vertex separator of (G, xyz).

However, in practice, we do not calculate the density of the embedding to compute a vertex separator. We will use more direct and practical subroutines for finding a vertex separator of (G, xyz) in the last step of the algorithm above. For example, a faster way to compute a vertex separator is to put every vertex v in the interior or exterior of S that is an endpoint of an edge in $\tilde{E}$, whichever has smaller cardinality, into the separator. Notice that the degree of a density graph or a finite element graph is bounded. So, this heuristic also finds a vertex separator that satisfies the separator bound in Theorem 3.3. To compute the smallest vertex separator induced by S, we can use the minimum bipartite vertex cover procedure (see Section 4.4). Although the worst case time complexity of such a procedure is quite expensive, its expected (average) time complexity is much lower, especially since in our case the bipartite graph induced by $\tilde{E}$ is much smaller than the original graph G.

Figure 8 shows the mesh of Figures 2 and 3 partitioned using our experimental implementation [39] of Algorithm 5 recursively.

Our mesh partitioning algorithms first use the geometric information to compute a continuous separator, then use the combinatorial structure to find a vertex separator. One major reason making the above partitioning algorithm suitable for efficient practical implementation is the use of *geometric sampling*, a technique that reduces the problem size and simultaneously guarantees a provably good approximation of the larger problem.

6. Proof outline for the main separator theorem. We now outline the proof of the Main Theorem 2.4, to show how to use geometric arguments to prove separator properties for graphs embedded in fixed dimension. The detailed proofs are presented in the companion paper [58]. In Section 6.1, we present a continuous separator theorem, based on which we give a geometric method for proving a small separator theorem in Section 6.2. We then apply this method to prove Theorem 2.4 in the remainder of this section.

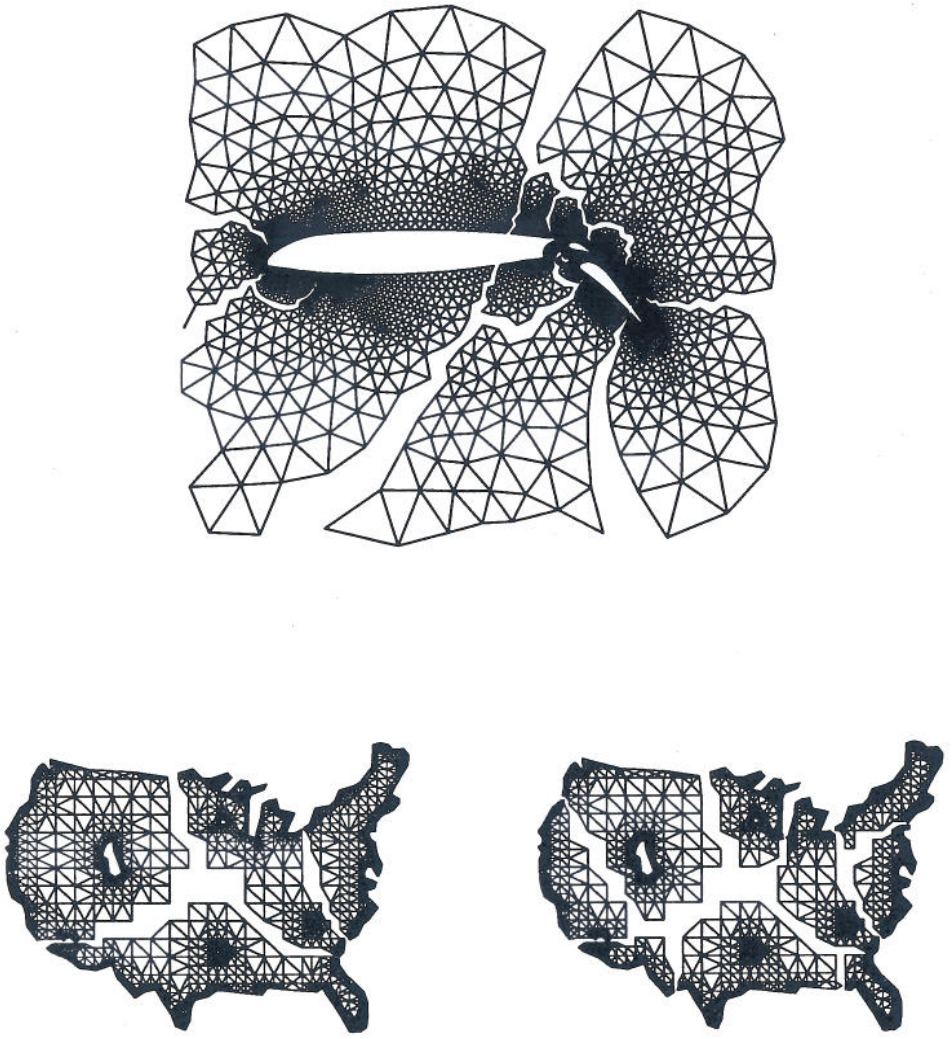

FIG. 8. *Recursive partitioning using Algorithm 5.*

6.1. A continuous separator theorem. In the partitioning algorithms presented above, a "continuous" separator, in the form of a sphere, is computed first, and then a "discrete" separator is deduced from its continuous counterpart. The quality of the continuous separating sphere, in a strong sense, determines the quality of the vertex separator.

Suppose $f(x)$ is a real-valued nonnegative function defined on $\mathbb{R}^d$ such that f^k is integrable for all $k = 1, 2, 3, \ldots$. Such an f is called a *cost function*. The *total volume* of the function f is defined as

$$\text{Total-Volume}(f) = \int_{v \in \mathbf{R}^d} (f(v))^d (dv)^d$$

Suppose S is a $(d-1)$-sphere in $\mathbb{R}^d$. The *surface area* of S is then

$$\text{Area}(f, S) = \int_{v \in S} (f(v))^{d-1} (dv)^{d-1}$$

Let π denote a map from $\mathbb{R}^d$ to U_d, which is formed by a stereographic projection, followed by a rotation, followed by an inverse stereographic projection, followed by a dilation, followed by a stereographic projection. Such a map is called an *H-map*. Our partitioning algorithms compute an H-map, choose a random great circle, and use the inverse of the H-map to transform the great circle to a sphere in $\mathbb{R}^d$.

For each great circle GC of U_d, let S_{GC} be the sphere defined by $\pi^{-1}(GC)$. Let $\text{Cost}(GC) = \text{Area}(f, S_{GC})$. Let $\text{Avg}(f)$ be the average cost of all great circles of U_d. We will use the following lemma, whose proof can be found in the companion paper [58].

LEMMA 6.1. *Suppose f is a cost function in $\mathbb{R}^d$. Then*

$$\text{Avg}(f) = O\left((\text{Total-Volume}(f))^{\frac{d-1}{d}}\right).$$

Consequently, we have the following continuous separator theorem.

THEOREM 6.2 (CONTINUOUS SEPARATOR). *Suppose f is a cost function on $\mathbb{R}^d$ and P is a set of n distinct points in $\mathbb{R}^d$. Let S be a sphere chosen by the random process described in Algorithm 1. Then S $(d+1)/(d+2)$-splits P, and with high probability,*

$$\text{Area}(f, S) = O\left((\text{Total-Volume}(f))^{\frac{d-1}{d}}\right).$$

The splitting ratio of the separator in the theorem above is $(d+1)/(d+2)$ instead of $d/(d+1)$. This is because points are mapped from $\mathbb{R}^d$ to the unit sphere in $\mathbb{R}^{d+1}$, and the center of the unit sphere is a $(d+1)$-dimensional center point of the image rather than a d-dimensional center point.

6.2. A new approach to proving small separator theorems. The continuous separator theorem of the last section provides the following generic approach to prove that a graph G embedded in $\mathbb{R}^d$ has an $O(c^{(d-1)/d})$-separator that $(d+1)/(d+2)$-splits.

A Geometric Approach to Proving Small Separator Theorems

1. Define a real-valued function f based on the structure of G so that Total-Volume(f) is bounded by a function c;
2. Find a $(d-1)$-dimensional separating sphere S that $(d+1)/(d+2)$-splits the vertices of G and has Area$(f, S) = O\left(c^{(d-1)/d}\right)$, by Theorem 6.2.
3. Deduce a vertex separator of G from the separating sphere S.

In order to deduce a vertex separator from its continuous counterpart, the function f must be *faithful* in the sense that the cost of a continuous separator models the size of a vertex separator of the underlying graph. In other words, the continuous function f faithfully encodes some combinatorial properties related to separators of the underlying graph.

We will follow the basic steps above to prove Theorem 2.4. To this end, for each α-overlap graph of a k-ply neighborhood system $\Gamma = \{B_1, \ldots, B_n\}$ in $\mathbb{R}^d$, we construct a real-valued function f based on Γ and prove that

$$\text{Total-Volume}(f) \leq O\left(\alpha^{\frac{d}{d-1}} k^{\frac{1}{d-1}} n\right);$$

then we show that from each separating sphere S we can deduce a vertex separator of size linearly bounded by Area(f, S).

6.3. Local cost functions. Just as each overlap graph is defined from its neighborhood system, $B_1, \ldots, B_n$, the cost function f itself is defined from the *local cost functions*, $f_1, \ldots, f_n$, with f_i based on B_i.

Let $P = \{p_1, \ldots, p_n\}$ be the set of centers of $\{B_1, \ldots, B_n\}$, and suppose that the radius of B_i is r_i. We define f_i as

$$f_i(x) = \begin{cases} 1/(2\alpha r_i) & \text{if } x \in (2\alpha) \cdot B_i, \text{ i.e., } ||x - p_i|| \leq 2\alpha r_i \\ 0 & \text{otherwise} \end{cases}$$

Intuitively, f_i sets up a cost on each $(d-1)$-sphere S such that the closer S is to B_i, the more B_i contributes to the surface area of S. The function f_i measures the cost of a sphere passing through B_i and its vicinity.

Notice that the function f_i defined above has Total-Volume(f_i) equal to V_d, the volume of a unit ball in d dimensions.

6.4. Putting local cost functions together. Now we need to put the local cost functions together into a global cost function. Perhaps the simplest way is to take the sum, $f = \sum_i f_i$. But this is not the best choice. To see this, just check the extreme case where the neighborhood is a collection of n identical balls and $\alpha = 1$. In this case $k = n$. The total volume of the sum is $n^d V_d$, while we need a cost function of total volume $O\left(n^{d/(d-1)}\right)$ to establish Theorem 2.4.

To achieve a tight bound, we make use of the "slight" difference between the various p-norms when applied to high-dimensional vectors. This is a technique that appears to be new and is interesting in its own right. Recently, Mitchell and Vavasis

[61] used a cost function similar to ours to analyze their three dimensional mesh generation algorithm.

Suppose $a_1, \ldots, a_n$ are reals. For each positive integer p, the L_p *norm* of $a_1, \ldots, a_n$, denoted $L_p(a_1, \ldots, a_n)$, is

$$L_p(a_1, \ldots, a_n) = \left(\sum_{i=1}^{n} |a|^p \right)^{\frac{1}{p}}.$$

The following lemma states the relationship between different norms.

LEMMA 6.3. *Let $a_1, \ldots, a_n$ be real. If $p \leq q$, then $L_p(a_1, \ldots, a_n) \geq L_q(a_1, \ldots, a_n)$.*

Proof: See Hardy, Littlewood and Pólya [41] (pages 26 and 144). □

We define the *global cost function* of the overlap graph to be the L_{d-1} norm of $f_1, \ldots, f_n$, i.e.,

$$f(x) = L_{d-1}(f_1, \ldots, f_n) = \left(\sum_{i=1}^{n} (f_i(x))^{d-1} \right)^{\frac{1}{d-1}}.$$

Notice that the L_d norm of f_i is not a good choice, because its total volume is nV_d for all neighborhood systems. The following lemma, proved in [58], bounds the total volume of the function f.

LEMMA 6.4. *Let $\Gamma = \{B_1, \ldots, B_n\}$ be a k-ply neighborhood system in $\mathbb{R}^d$. If $f_1, \ldots, f_n$ are the local cost functions of Γ and f is its global cost function, then*

$$\text{Total-Volume}(f) = O(\alpha^{\frac{d}{d-1}} k^{\frac{1}{d-1}} n).$$

Consequently, by Theorem 6.2, we have the following lemma.

LEMMA 6.5. *Suppose $\{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. Then there exists a $((d+1)/(d+2))$-splitting sphere S of $\{B_1, \ldots, B_n\}$ with $\text{Area}(f, S) = O\left(\xi \alpha k^{1/d} n^{(d-1)/d}\right)$, where $\xi = 2^{d-1} V_d$.*

6.5. A vertex separator from a continuous one. Recall that a ball B_i is an *overlap neighbor* of a sphere S if one of the following conditions is true.

1. $B_i \cap S \neq \emptyset$;
2. $\alpha \cdot B_i \cap S \neq \emptyset$ and $r_i \leq r$.

The number of overlap neighbors of S is called the *overlap number* of S, denoted $\vartheta_\Gamma(S)$. The following lemma bounds the overlap number $\vartheta_\Gamma(S)$ of a k-ply neighborhood system in $\mathbb{R}^d$.

LEMMA 6.6. *Suppose $\Gamma = \{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. Let $f_1, \ldots, f_n$ be local cost functions defined for the α-overlap graph of Γ and let $f = L_{d-1}(f_1, \ldots, f_n)$. For each $(d-1)$-sphere S,*

$$\vartheta_\Gamma(S) = O(\alpha^d k) + O(\text{Area}(f, S)).$$

The constant in the big-O notation depends only on d.

Theorem 2.4 follows from Lemma 6.4, Theorem 6.2, and Lemma 6.6.

7. Final remarks. We have demonstrated that geometric structure is useful for mesh partitioning. We have shown that geometric sampling can be used to reduce the problem size, making our approach feasible for practical applications in large-scale computation. We have implemented the random linear-time separator algorithm of Section 6 [39] and have experimented with various examples. The numerical results are encouraging. With the help of some heuristics to speed up the geometric transformation and the local optimization, the program is very fast. In practice, our program generates partitions much better than what the theoretical results predict; the partitions are competitive with such previous methods as those based on an expensive eigenvector computation [65].

Recently, Eppstein, Miller and Teng [24] have showed that a small separator for the intersection graph of a k-ply neighborhood system can be in fact found in deterministic linear time.

We conclude the paper with the following two open questions.

1. What is the computational complexity of deciding whether a graph G is k-embeddable or (α, k)-embeddable?
2. Is there a polynomial time algorithm for computing the disk packing of a planar graph?

Acknowledgments. We would like to thank David Applegate, Marshall Bern, David Eppstein, John Gilbert, Bruce Hendrickson, Ravi Kannan, Tom Leighton, Mike Luby, Oded Schramm, Doug Tygar, and Kim Wagner for invaluable help and discussions. We would like to thank John Gilbert especially for his editorial contribution which greatly improved this paper.

REFERENCES

[1] A. Agrawal and P. Klein. Cutting down on fill using nested dissection: Provably good elimination orderings. In *Sparse Matrix Computations: Graph Theory Issues and Algorithms, IMA Volumes in Mathematics and its Applications*, (this book), A. George, J. Gilbert and J. Liu, Springer-Verlag, New York. 1992.

[2] N. Alon, P. Seymour, and R. Thomas. A separator theorem for non-planar graphs. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing*, Maryland, May 1990. ACM.

[3] E. M. Andreev. On convex polyhedra in Lobacevskii space. *Math. USR Sbornik*, 10(3):413–440, 1970.

[4] E. M. Andreev. On convex polyhedra of finite volume in Lobacevskii space. *Math. USR Sbornik*, 12(2):270–259, 1970.

[5] M. J. Berger and S. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. Comp.*, C-36:570-580, 1987.

[6] M. Bern, D. Eppstein and J. R. Gilbert. Provably good mesh generation. In *31st Annual Symposium on Foundations of Computer Science, IEEE*, 231–241, 1990, (to appear JCSS).

[7] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*, F. K. Hwang and D.-Z. Du editors, World Scientific, 1992.

[8] G. Birkhoff and A. George. Elimination by nested dissection. *Complexity of Sequential and Parallel Numerical Algorithms*, J. F. Traub, Academic Press, 1973.

[9] P. E. Bjørstad and O. B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.*, 23:1097-1120, 1986.

[10] G. E. Blelloch. *Vector Models for Data-Parallel Computing.* MIT-Press, Cambridge MA, 1990.

[11] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. An iterative method for elliptic problems on regions partitioned into substructures, *Math. Comp.* 46:361-9, 1986.

[12] D. Calahan. Parallel solution of sparse simultaneous linear equations. in *Proceedings of the 11th Annual Allerton Conference on Circuits and Systems Theory*, 729-735, 1973.

[13] T. F. Chan and D. C. Resasco. A framework for the analysis and construction of domain decomposition preconditioners. UCLA-CAM-87-09, 1987.

[14] L. P. Chew. Guaranteed quality triangular meshes, Department of Computer Science, Cornell University TR 89-893, 1989.

[15] K. Clarkson. Fast algorithm for the all-nearest-neighbors problem. In *the 24th Annual Symposium on Foundations of Computer Science*, 226–232, 1983.

[16] R. Cole, M. Sharir and C. K. Yap. On k-hulls and related problems. *SIAM J. Computing*, 61, 1987.

[17] J. H. Conway, and N. J. A. Sloane. *Sphere Packings, Lattices and Groups.* Springer-Verlag, 1988.

[18] L. Danzer, J. Fonlupt and V. Klee. Helly's theorem and its relatives. *Proceedings of Symposia in Pure Mathematics*, American Mathematical Society, 7: 101-180, 1963.

[19] H. N. Djidjev. On the problem of partitioning planar graphs. *SIAM J. Alg. Disc. Math.*, 3(2):229–240, June 1982.

[20] A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Canadian J. Math.* 10, pp 517-534, 1958.

[21] I. S. Duff. Parallel implementation of multifrontal schemes. *Parallel Computing*, 3, 193–204, 1986.

[22] M. E. Dyer. On a multidimensional search procedure and its application to the Euclidean one-centre problem. *SIAM Journal on Computing* 13, pp 31–45, 1984.

[23] D. Eppstein, G. L. Miller, C. Sturtivant and S.-H. Teng. Approximating center points with and without linear programming. Manuscript, Massachusetts Institute of Technology, 1992.

[24] D. Eppstein, G. L. Miller and S.-H. Teng. A deterministic linear time algorithm for geometric separators and its applications. Manuscript, Xerox Palo Alto Research Center, 1991.

[25] I. Fáry. On straight line representing of planar graphs. *Acta. Sci. Math.* 24: 229–233, 1948.

[26] R. W. Floyd and R. L. Rivest. Expected time bounds for selection. *CACM* 18(3): 165–173, March, 1975.

[27] H. de Fraysseix, J. Pach, and R. Pollack. Small sets supporting Fáry embeddings of planar graphs. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 426–433, 1988.

[28] G. N. Fredrickson and R. Janardan. Separator-based strategies for efficient message routing. In *27st Annual Symposium on Foundation of Computation Science, IEEE*, 428–237, 1986.

[29] I. Fried. Condition of finite element matrices generated from nonuniform meshes. *AIAA J.* 10, pp 219–221, 1972.

[30] A. M. Frieze, G. L. Miller and S.-H. Teng. Separator based divide and conquer in computational geometry. *Proceedings of the 1992 ACM Symposium on Parallel Algorithms and Architectures*, 1992.

[31] H. Gazit. An improved algorithm for separating a planar graph. Manuscript, Department of Computer Science, University of Southern California, 1986.

[32] H. Gazit and G. L. Miller. A parallel algorithm for finding a separator in planar graphs. In *28st Annual Symposium on Foundation of Computation Science, IEEE*, 238–248, Los Angeles, October 1987.

[33] H. Gazit. A deterministic parallel algorithm for planar graph isomorphism. In *32nd Annual Symposium on Foundations of Computer Science, IEEE*, to appear, 1991.

[34] J. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numerical Analysis*, 10: 345–363, 1973.

[35] A. George, M. T. Heath, J. Liu, E. Ng. Sparse Cholesky factorization on a local-memory multiprocessor. *SIAM J. on Scientific and Statistical Computing*, 9, 327–340, 1988.

[36] J. A. George and J. W. H. Liu. An automatic nested dissection algorithm for irregular finite element problems. *SIAM J. on Numerical Analysis*, 15, 1053–1069, 1978.

[37] J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, 1981.

[38] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5 pp391–407, 1984.

[39] J.R. Gilbert, G.L. Miller, and S.-H. Teng. Geometric mesh partitioning: Implementation and experiments. Technical Report, Xerox Palo Alto Research Center, to appear, 1992.

[40] J. R. Gilbert and R. E. Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50(4):377–404, 1987.

[41] G. Hardy, J. E. Littlewood and G. Pólya. *Inequalities.* Second edition, Cambridge University Press, 1952.

[42] D. Haussler and E. Welzl. ϵ-net and simplex range queries. *Discrete & Computational Geometry*, 2: 127–151, 1987.

[43] J. P. Hutchinson and G. L. Miller. On deleting vertices to make a graph of positive genus planar. In *Discrete Algorithms and Complexity Theory - Proceedings of the Japan-US Joint Seminar, Kyoto, Japan*, pages 81–98, Boston, 1986. Academic Press.

[44] C. Jordan. Sur les assemblages de lignes. *Journal Reine Angew. Math*, 70:185–190, 1869.

[45] F. T. Leighton. *Complexity Issues in VLSI.* Foundations of Computing. MIT Press, Cambridge, MA, 1983.

[46] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multi-commodity flow problems with applications to approximation algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pp 422-431, 1988.

[47] C. E. Leiserson. *Area Efficient VLSI Computation.* Foundations of Computing. MIT Press, Cambridge, MA, 1983.

[48] C. E. Leiserson and J. G. Lewis. Orderings for parallel sparse symmetric factorization. in *3rd SIAM Conference on Parallel Processing for Scientific Computing*, 1987.

[49] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16:346–358, 1979.

[50] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. of Appl. Math.*, 36:177–189, April 1979.

[51] R. J. Lipton and R. E. Tarjan. Applications of planar separator theorem. *SIAM J. Comput*, 9(3): 615–627, August 1981.

[52] J. W. H. Liu. The solution of mesh equations on a parallel computer. in 2nd Langley Conference on Scientific Computing, 1974.

[53] N. Megiddo. Linear programming in linear time when the dimension is fixed. *SIAM Journal on Computing* 12, pp 759–776, 1983.

[54] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, June 1986.

[55] G. L. Miller and S.-H. Teng. Centerpoints and point divisions. Manuscript, School of Computer Science, Carnegie Mellon University, 1990.

[56] G. L. Miller, S.-H. Teng, and S. A. Vavasis. A unified geometric approach to graph separators. In *32nd Annual Symposium on Foundations of Computer Science, IEEE*, pp538–547, 1991.

[57] G. L. Miller, S.-H. Teng, W. Thurston and S. A. Vavasis. Separators for sphere-packings and nearest neighborhood graphs. in progress 1992.

[58] G. L. Miller, S.-H. Teng, W. Thurston and S. A. Vavasis. Finite element meshes and geometric separators. in progress 1992.

[59] G. L Miller and W. Thurston. Separators in two and three dimensions. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing*, pages 300–309, Maryland, May 1990. ACM.

[60] G. L. Miller and S. A. Vavasis. Density graphs and separators. In *Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 331–336, San Francisco, January 1991. ACM-SIAM.

[61] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. Proc. ACM Symposium on Computational Geometry, pp 212-221, 1992.

[62] M. S. Paterson. Tape bounds for time-bounded Turing machines. *J. Comp. Syst. Sci.*, 6:116–124, 1972.

[63] V. Pan and J. Reif. Efficient parallel solution of linear systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 143–152, Providence, RI, May 1985. ACM.

[64] A. Pothen and C.-J. Fan. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software* 16 (4), pp 303-324, 1990.

[65] A. Pothen, H. D. Simon, K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* 11 (3), pp 430–452, July, 1990.

[66] J. H. Reif and S. Sen. Polling: A new randomized sampling technique for computational geometry. In *Proceedings of the 21st annual ACM Symposium on Theory of Computing.* 394–404, 1989.

[67] E. Schwabe, G. Blelloch, A. Feldmann, O. Ghattas, J. Gilbert, G. Miller, D. O'Hallaron, J. Schewchuk and S.-H. Teng. A separator-based framework for automated partitioning and mapping of parallel algorithms in scientific computing. In *First Annual Dartmouth Summer Institute on Issues and Obstacles in the Practical Implementation of Parallel Algorithms and the use of Parallel Machines*, 1992.

[68] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 2:(2/3), pp135-148.

[69] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*, Prentice-Hall, 1973.

[70] S.-H. Teng. *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991. CMU-CS-91-184.

[71] C. Thomassen. Planarity and duality of finite and infinite graphs. *Journal of Combinatorial Theory*, Series B, 29: 244–271, 1980.

[72] J. F. Thompson, Z. U. A. Warsi and C. W. Mastin. *Numerical Grid Generation: Foundations and Applications.* New York, North Holland, 1985.

[73] W. P. Thurston. *The geometry and topology of 3-manifolds.* Princeton University Notes, 1988.

[74] W. T. Tutte. Convex representations of graphs. *Proc. London Math. Soc.* 10(3): 304–320, 1960.

[75] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc.* 13(3): 743–768, 1963.

[76] J. D. Ullman. *Computational Aspects of VLSI.* Computer Science Press, Rockville MD, 1984.

[77] P. Ungar. A theorem on planar graphs. *Journal London Math Soc.* 26: 256–262, 1951.

[78] P. M. Vaidya. Constructing provably good cheap preconditioners for certain symmetric positive definite matrices. *IMA Workshop on Sparse Matrix Computation: Graph Theory Issues and Algorithms*, Minneapolis, Minnesota, October 1991.

[79] L. G. Valiant. Universality consideration in VLSI circuits. *IEEE Transaction on Computers*, 30(2): 135–140, February, 1981.

[80] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16: 264-280, 1971.

[81] S. A. Vavasis. Automatic domain partitioning in three dimensions. *SIAM J. Sci. Stat. Comp.*, 12 (1991) 950-970.

[82] R. D. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. Technical Report, California Institute of Technology, 1990.

[83] F.-F. Yao. A 3-space partition and its application. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, ACM*, 258–263, 1983.

[84] E. E. Zmijewski. *Sparse Cholesky Factorization on a Multiprocessor.* PhD thesis, Department of Computer Science, Cornell University, 1987.