Towards a Theory of Automated Elliptic Mesh Generation

J.Q. Cordova
Visual Computing Incorporated
Mountain View, CA 94043

Abstract

The theory of elliptic mesh generation is reviewed and the fundamental problem of constructing computational space is discussed. It is argued that the construction of computational space is an NP-Complete problem and therefore requires a non-standard approach for its solution. This leads to the development of graph-theoretic, combinatorial optimization and integer programming algorithms. Methods for the construction of 2-D computational space are presented.

Introduction

Since the introduction of elliptic mesh generation as an all-purpose technology, there have been various attempts to codify the blocking process that must preceed the solving of elliptic equations. From these attempts, three distinct approaches have emerged: interactive, algorithmic, and knowledge based [5], [1], [20]. It is probably fair to say that this ordering also reflects the relative success of each approach. Why should this be? Could it be that human intervention is necessary for solving the blocking problem? Note that human interaction is minimal for unstructured mesh generation. Therefore, why should it be needed for structured mesh generation? This line of inquiry leads to the consideration of combining structured and unstructured mesh generation within a generalized theory.

Structured mesh generation (SMG) and unstructured mesh generation (UMG) belong to the general discipline of computational geometry (CG). Computational geometry is concerned with the systematic study of geometric algorithms [8], [16], [17]. A typical sample of CG problems might include geometric searching, computation of convex hulls, surface-surface intersection, visibility detection, and motion planning. Mesh generation is concerned with the problem of decomposing complex shapes into simpler pieces. The simplest definitions of UMG and SMG as CG problems are as follows:

(UMG) Decompose a region bounded by algebraic surfaces of degree 1 into tetrahedra.
(SMG) Decompose a region bounded by algebraic surfaces of degree ≥ 1 into deformed hexahedra.

While these definitions are not complete, they serve to emphasize the main relationship between unstructured and structured mesh generation. This is that UMG is a linear problem while SMG is a nonlinear problem. In particular, UMG is a special case of SMG since it is easy to construct a tetrahedral decomposition for each hexahedron.

The view that SMG should be developed within the framework of non-linear computational geometry can be motivated in another way. Namely, that it represents a natural generalization of previous theories. This is presented next as an evolution of algorithms A -> B -> C.

The standard formula for elliptic mesh generation is based on a four step procedure:

Algorithm A
I. Generate curves
II. Assemble curves into block faces
III. Set boundary point numbering and point distribution
IV. Solve elliptic equations in each block

The subtle difficulty with this approach is that topological and geometric constructions are intermixed. For example, Step I serves to simultaneously identify an edge in computational space and the mapping of that edge into a curve in physical space. Thus, "block" is used to reference a region in computational space as well as its image in physical space. This shortcoming was realized by Allwright [1], who proposed a different procedure:

Algorithm B
I. Approximate configuration using hypercubes
II. Add surrounding hypercubes to form a cube
III. Set boundary conditions on hypercube faces
IV. Solve elliptic equations globally over computational space

The fundamental idea behind Allwright's method is that it is possible to construct a suitable computational space for a configuration using only "simple" geometric information. Then, given a computational space, the positions of zonal boundaries are to be determined by solving the elliptic equations globally. Thus, with topological and geometric operations separated, a large part of the elliptic grid generation process can be automated.

The theory and algorithms for automated elliptic mesh generation presented in this paper are based on a five step procedure:

Algorithm C
I. Approximate configuration with polyhedra
II. Decompose surrounding space into convex polyhedra
III. Construct computational space
IV. Set boundary conditions on block boundaries
V. Solve elliptic equations globally over computational space

Thus, while B uses a piecewise constant approximation to the configuration, C uses a piecewise linear approximation. And, while B adds only exterior blocks, C considers both the addition of interior and exterior blocks as defined by a general decomposition problem. In addition, these blocks may be convex polyhedra instead of cubes. Finally, the relation between a blocking and computational space is addressed in C.

In summary, SMG is a non-linear problem and therefore one approach for constructing solutions is via appropriate linearization schemes. This leads directly into computational geometry and to the relationship between structured and unstructured mesh generation.

The remainder of this paper is organized around providing a clearer description of steps II-III in C and discussing algorithms for their solution in two dimensions. A central theme of this paper is that some problems are simply "hard" to solve. This idea is developed in the first section on complexity and algorithms. Section two is concerned with the principal "hard" problem of zone generation and its reformulation as a polygon decomposition problem. Algorithms for polygon decomposition are then outlined. The encoding of a blocking topology from a set of intersecting curves is presented in the third section. This establishes an identification between physical space and an incomplete computational space. To complete the construction of computational space, block coordinates must be resolved. This is discussed in the section on boundary point numbering.

Complexity and Algorithms

The complexity of an algorithm can be measured by counting the number of arithmetic operations it takes to process its input. Of particular interest is the asymptotic value of the operations count as the input size becomes large. The complexity of several important algorithms is listed in the table below.

Algorithm	Input Size	Operations Count
Polygon Triangulation	N vertices	O(N)
Sorting	N keys	O(N logN)
Hidden Surface Removal	N polygons	$O(N^2)$
Gaussian elimination	N unknowns	$O(N^3)$

These algorithms have the property that they solve a certain problem in polynomial time. That is, the operations count is a polynomial function of the input size. Such problems are said to be in the class P. There are many problems for which polynomial time algorithms are not known but for which a solution can be verified in polynomial time. A classic example is the traveling salesman problem: given a set of cities, and distances between all pairs, find a tour of all the cities of distance less than some value. Problems of this kind are said to be in the class NP. In summary then:

P: The class of problems which can be solved in polynomial time.
NP: The class of problems for which a solution can be verified in polynomial time.

Clearly all problems in P are also in NP but it is not known if P = NP. This is one of the outstanding problems in computer science. Among the class of NP problems, many of practical importance have a special property: if they could be solved in polynomial time, then *all* problems in NP could be solved in polynomial time. These problems are said to be NP-complete. To date, no one has been able to find efficient algorithms for any NP-complete problem and therefore it is generally believed that $P \neq NP$. This is usually interpreted as saying that NP-complete problems are harder than those in P. What about problems that are not NP-complete or are not in NP? Any problem which can be transformed to an NP-complete problem will have the property that it cannot be solved in polynomial time unless P = NP. Such problems are said to be NP-hard because they are at least as hard as the NP-complete problems. A more thorough discussion of these issues can be found in [9].

Many important problems in combinatorial optimization and computational geometry are either NP-complete or NP-hard. The only known algorithm guaranteed to produce a solution is exhaustive search, which has exponential complexity. As will be shown, several problems associated with blocking and the construction of computational space are NP-complete. This observation will be important in guiding a search for algorithms which are suitable for such intrinsically difficult problems.

Polygon Decomposition

The problem of constructing a zonal decomposition of a 2-D region bounded by smooth curves has been investigated previously [1], [7], [19], [20]. From these investigations there has emerged a weak understanding that the zonal decomposition problem consists of several sub-problems:

- When should two curves be connected by a zonal curve?
- How many connections should there be between connected curves?
- Where should connections intersect the curves they connect?
- What shapes should the connections have?

As noted in the introduction, this is a non-linear problem and local solutions may be obtained via a linearization process.

> Linearized Zonal Decomposition Problem: Given a polygonal region (a polygon with polygonal holes), decompose it into an optimal number of "quad-like" polygons.

Approximation by Polygons

The problem of polygonal decomposition of polygonal regions is central to both structured and unstructured mesh generation. When the polygon is a triangle, a comprehensive theory can be developed and this leads to the well known triangulation algorithms used in unstructured mesh generation [8], [17]. The general problem of n-gon decomposition is much more difficult and in several cases is known to be NP-complete or NP-hard [3], [10]. However, it is precisely the problem of 4-gon decomposition that is important to elliptic grid generation. It is interesting and significant that it is possible to isolate the essential difficulties of 4-gon decomposition. As discussed in [16], the quadrilateralization problem becomes difficult when the region is multiply connected. The following table summarizes these observations on approximation by polygons:

Problem	Classification	Algorithm
3-gon decomposition of polygonal regions with holes.	P	Constrained Delaunay triangulation [6].
Convex 4-gon decomposition of polygonal regions without holes.	P	Lubiw's dynamic programming algorithm [11].
Convex 4-gon decomposition of polygonal regions with holes.	NP-complete	?

The intrinsic difficulty of 4-gon decomposition is associated with deciding how to reduce the connectivity of a region. We interpret this to mean that more than geometric information is needed to decide whether a blocking of a multiply connected region is acceptable. This point of view is supported by examining the role of physics in blocking. As an example, consider the multiply connected region defined by the exterior of a 4-element airfoil geometry. For fluid dynamicists, it is natural to add wake-cuts to each airfoil and thereby reduce the connectivity of the region. However, this region might also represent a plate with 4 holes and the problem under investigation could be to simulate crack propogation. A structural dynamicist might therefore choose to add cuts between the holes. In either case, block boundaries are positioned so as to control grid density and therefore the quality of the physics calculation. More generally, block boundaries represent "initial guesses" for an adaptive meshing procedure. From this viewpoint, connectivity reduction becomes an intrinsically difficult problem for both structured and unstructured mesh generation.

The decomposition of polygonal regions into convex quadrilaterals is not always possible. For example, a pentagon cannot be quadrilateralized unless interior vertices are allowed. These interior vertices are known as Steiner points and are also necessary to achieve optimal decompositions [3].In practice, it may be convenient to add not only interior vertices, but interior line segments as well. The general decomposition problem relevant to elliptic mesh generation is therefore as follows:

> Convex Decomposition Problem: Construct a convex decomposition of the interior of a simple polygon containing obstacles of the form: simple polygon, line segment, point.

Of course, the main interest is in having a convex 4-gon decomposition. However, for the purpose of generating a blocking, n-gons with 4 distinguished corners are acceptable. A novel algorithm for constructing such a decomposition will be briefly discussed. More complete details are given in [4].

4-gon Decomposition Algorithm

The algorithm used to solve the convex decomposition problem has the following simple structure:

> (A) Construct a constrained Delaunay triangulation of the region.
> (B) Remove edges in the triangulation to generate convex 4-gons.

Efficient algorithms for (A) are well known [6] and will not be discussed here. Step B is a problem in combinatorial optimization. Moreover, it must be NP-complete or NP-hard because step A is in the class P but (A)-(B) solves the intrinsically difficult problem of 4-gon decomposition. Recently, a new class of algorithms (genetic algorithms) have been successfully applied to similar difficult combinatorial optimization problems [14]. These genetic algorithms are based on the principles of reproduction, crossover, and mutation as applied to a pool of near-optimal solutions. While the theory of genetic algorithms is in its infancy, they are proving to be of practical value on a growing number of problems. The development of such an algorithm for the solution of B is described in a separate paper [4].

In practice, the application of this algorithm results in a decomposition consisting of n-gons where n=4,5 and further optimization may produce polygons with even more edges. A data-structure for encoding the topology of such a mixed decomposition is discussed next.

Graphs and 2-D Blocking

Graphs and graph theory can be used to understand the topology of a blocking. In the simplest description, a blocking consists of a set of curves C (block boundaries) which intersect only at their endpoints P. The topology of a blocking can be described by a directed graph G = (V,E) through the correspondence {vertices V <-> points P}, {edges E <-> curves C}, and {edge directions <-> curve orientations}. This correspondence is realized by means of an embedding of the digraph into the surface.

$$(V,E) \rightarrow \mathbf{R}^2$$
$$V \mapsto P$$
$$E \mapsto C$$

Although a blocking determines a unique digraph and digraph embedding, recovering a blocking from a digraph requires additional structure.This structure is known as a rotation system [12]. A rotation system for a graph is defined via a rotation system for each vertex. A rotation system at a vertex is an ordering of edges incident to that vertex. This definition must be generalized for digraphs because edges may be incident to or incident from a vertex.

Definition: A rotation system for a digraph (V,E) is a subset of signed edges E_v and a rotation operator R_v which maps an edge in E_v to its successor. The set E_v contains e or -e depending on whether e is incident to or incident from v respectively.

With this definition, each vertex is assigned a unique subset of signed edges E_v , each edge in E_v is incident *to* v, and the union of E_v as v ranges over V is $\pm E$. Now, starting with an edge e_i, the rotation system can be used to construct a successor edge $e_{i+1} = - R\, e_i$. Continuing in this manner, a directed path through the digraph vertex set can be built. This leads to the following definition:

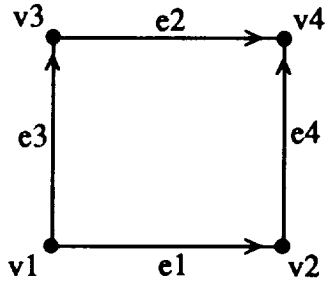
Definition: An R-path is a sequence of signed edges $(e_1, e_2, \ldots e_n)$ such that $e_{i+1} = -R\, e_i$ for some rotation operator R. Edges e_1 and e_2 are said to be R-path connected if they are members of the same R-path.

When an R-path is closed, it is called an R-cycle. More generally, the following definition holds:

Definition: An R-cycle is an R-path in which every edge has a successor.

An example helps to clarify the definitions given above. Consider a blocking which consists of a single square with curves oriented as shown below. A diagraph and rotation system for this blocking are

e1 = (v1,v2) E_{v1} = (-e1,-e3)
e2 = (v3,v4) E_{v2} = (e1,-e4)
e3 = (v1,v3) E_{v3} = (e3,-e2)
e4 = (v2,v4) E_{v4} = (e2,e4)



Note that because there are only two signed edges in E_v, the rotation operator R_v is trivially defined.

Starting at edge e3, the R-cycle (e3, e2, -e4, -e1) is computed as

$$
\begin{aligned}
& e3 \\
& e2 = -R_{v3}\, e3 \\
& -e4 = -R_{v4}\, e2 \\
& -e1 = -R_{v2}\, -e4
\end{aligned}
$$

and starting at e4, the R-cycle (e4, -e2, -e3, e1) is computed as

$$
\begin{aligned}
& e4 \\
& -e2 = -R_{v4}\, e4 \\
& -e3 = -R_{v3}\, -e2 \\
& e1 = -R_{v1}\, -e3
\end{aligned}
$$

It is easy to verify that there are no other R-cycles induced by this rotation system. In this case, all of the R-cycles have been found by computing the maximal R-paths. In fact, this can be proved in general.

Theorem: There is a 1-1 correspondence between maximal R-path components and R-cycles.

Proof: It is first shown that every maximal R-path component E is an R-cycle. If not, then there is some edge e in E that does not have a successor so that e' = -R e is not in E. But (e,e') is an R-path containing e which implies that E is not maximal. Similarly, if C is an R-cycle that is not a maximal R-path, then some edge e in C must have a successor e' = -R e that is not in C. But, C is R-path connected so that e' must be in C.

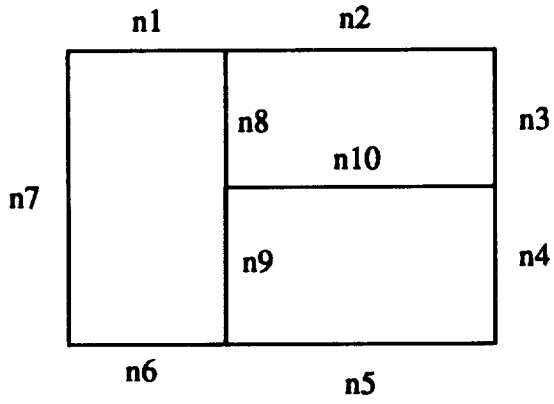
The maximal R-path components provide a partitioning of ±E into disjoint R-cycles. Intuitively, these R-cycles correspond to block boundaries. In the example above, the two R-cycles correspond to the two blocks representing the inside of the square and the outside of the square. In general, the R-cycles associated with an arbitrary rotation system do not correspond to block boundaries.

In summary, a digraph with certain vertex-edge operators can be used to encode the topology of a blocking. In practice, this may be used to reconstruct a blocking from a set of intersecting curves. This concept can be extended to higher dimensions where it can be used to encode the topology of a 3-D blocking [15].

Boundary Point Numbering

The density of grid points in a computational mesh is strongly influenced by the density of points on the corresponding boundary curves. In practice, the density of points is controlled by specifying the curve parametrization and the total number of points on the curve. While the selection of a parametrization can be quite arbitrary, the numbering problem is generally constrained. For example, multi-block point continuous meshes must have a consistent numbering across block boundaries. A blocking together with a such a numbering constitutes a description of computational space. In this section it is shown that the boundary point numbering problem is equivalent to computing integer solutions to a linear programming problem.

Theboundary point numbering problem is easier to discuss with a simple example in mind. Thus, consider the blocking shown in the figure below.



The problem under consideration is to number each curve so that opposing block faces have an equal number of points. Let (n1, n2, ..., n10) be the unknown values for the curve numberings. Then, the following system of 6 linear equations in 10 unknowns must hold:

$$\begin{aligned} n1 - n6 &= 0 \\ n2 - n10 &= 0 \\ n10 - n5 &= 0 \\ n7 - (n8+n9) &= 0 \\ n8 - n3 &= 0 \\ n9 - n4 &= 0 \end{aligned}$$

In general, for a blocking with E edges and B blocks, this will lead to a 2B x E linear system:

$$\mathbf{Ln = 0}. \qquad (1)$$

The linear system (1) is underdetermined and in this case it is easy to discover the extra degrees of freedom (n1, n2, n3, n4 for example). In practice, it may be desirable to fix certain values and have inequality constraints on others. For example:

$$\begin{aligned} n1 &= 10 \\ n2 &= 10 \\ n3 &= 10 \\ n7 &\geq 20. \end{aligned}$$

As a final constraint, the solutions to (1) should all be positive integers and it is desired to find the "smallest" such solution. After a simple transformation of variables, the edge numbering problem can be posed as follows:

Given integer valued **L** and **m**, find integer solutions to the problem:

$$\text{Minimize } \| \mathbf{n} \|_1$$

subject to:

$$\mathbf{L}\,\mathbf{n} = \mathbf{m} \qquad (2)$$

$$\mathbf{n} \geq \mathbf{0}$$

This problem is well known as an integer linear programming (ILP) problem [13]. Integer programming is another NP-Complete problem [9] and so it appears that the edge numbering problem is intrinsically difficult.

The simplex algorithm [2] is a well known iterative technique for solving linear programming (LP) problems. There is no guarantee that application of the simplex algorithm to (2) will produce an optimal integer solution. In other words, optimal solutions to LP problems are not necessarily integer. As discussed in [13], a necessary and sufficient condition for an LP with integer data to have an optimal solution that is integer is that the matrix **L** be totally unimodular. This means that every square, nonsingular submatrix of **L** has an integer inverse. It is interesting that the matrix **L** in (2) has this property and therefore that the simplex algorithm can be used to solve (2). The proof makes use of the following theorem.

Theorem: An integer matrix **L** with $L_{ij} = 0, 1, -1$ for all i and j, is totally unimodular if

1. No more than two nonzero entries appear in each column.
2. The rows can be partitioned into two subsets Q_1 and Q_2 such that
   (a) If a column contains two nonzero entries with the same sign, one entry is in each of the subsets.
   (b) If a column contains two nonzero elements of opposite sign, both elements are in the same subset.

Condition (1) is met because each curve is counted at most two times in the blocking equations. For curves i counted exactly twice (zonal curves), it is usually possible to arrange that n_i appear in the blocking equations with both plus and minus signs. In that case, set $Q_1 = \{\text{index set of the rows}\}$ and $Q_2 = \{\text{empty set}\}$.

Summary

A theory of structured mesh generation has been developed within the framework of computational geometry. In this theory, the blocking step of structured mesh generation is identified with the problem of hexahedral decomposition. In the plane, this reduces to quadrilateralizing a polygonal region. The fact that this problem is NP-complete accounts for the empirically observed result that blocking is hard. Knowing that a problem is NP-complete is useful in choosing appropriate algorithms for its solution. Novel iterative algorithms based on genetic search have been developed to solve the blocking problem. The relationship between a block decomposition and computational space has been investigated using methods of topological graph theory. This has resulted in the definition of a blocking as a digraph with rotation system. Computational space is then a blocking together with block coordinates. The problem of computing block coordinates is shown to be equivalent to an integer linear programming problem. The special structure of the boundary point numbering equations allows a standard solution to this problem using the simplex method.

References

[1] Allwright, S.E., *Techniques in Multiblock Domain Decomposition and Surface Grid Generation,* in Numerical Grid Generation in Computational Fluid Dynamics, 1988.

[2] Brickman, L., *Mathematical Introduction to Linear Programming and Game Theory,* Springer Verlag, 1988.

[3] Chazelle, B. and Dobkin, D.P., *Optimal Convex Decompositions,* in Computational Geometry, G. Toussaint Ed., Elsevier Science, 1985.

[4] Cordova, J.Q., *Computational Geometric Aspects of Automated Mesh Generation,* Int. J. Comp. Geometry & Applications (to appear), 1992.

[5] Cordova, J.Q., *VisualGrid: A Software Package for Interactive Grid Generation,* AIAA 90-1607, 21st Fluid Dynamics Conference, Seattle, 1990.

[6] Chew, P.L., *Constrained Delaunay Triangulations*, Proc. 3rd Symposium on Computational Geometry, ACM, 1987.

[7] Dannenhoffer, J.F., *A Block-Structuring Technique for General Geometries*, AIAA 91-0145, 29th Aerospace Sciences Meeting, Reno, 1991.

[8] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer Verlag, 1987.

[9] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., 1979.

[10] Keil, M.J. and Sack, J.R., *Minimum Decomposition of Polygonal Objects*, in Computational Geometry, G. Toussaint Ed., Elsevier Science, 1985.

[11] Lubiw, A. , *Decomposing Polygonal Regions Into Convex Quadrilaterals*, Proc. 1st Symposium on Computational Geometry, ACM, 1985.

[12] Gross, J.L. and Tucker, J.L., *Topological Graph Theory,* Wiley Interscience, 1987.

[13] Garfinkel, R.S. and Nemhauser, G.L., *Integer Programming*, Wiley Interscience, 1972.

[14] Goldberg, D.E., *Genetic Algorithms*, Addison Wesley, 1989.

[15] Noble, S.S. and Cordova, J.Q., *Blocking Algorithms for Structured Mesh Generation,* AIAA 92-0659, 30th Aerospace Sciences Meeting, Reno, 1992.

[16] O'Rourke, J., *Art Gallery Theorems and Algorithms*, Oxford Univ. Press, 1987.

[17] Preparata, F.P. and Shamos, M.I., *Computational Geometry*, Springer Verlag, 1985.

[18] Sedgewick, R., *Algorithms*, Addison Wesley, 1983.

[19] Steinbrenner, J.P., Chawner, J.R., and Fouts, C.L., *The GRIDGEN 3-D Multiple Block Grid Generation System*, WRDC-TR-90-3022, July, 1990.

[20] Vogel, A.A., *A Knowledge-Based Approach to Automated Flow-Field Zoning for Computional Fluid Dynamics*, NASA TM 101072, April, 1989.