FAST MONTE CARLO DOMAIN SAMPLING FOR DISCRETE FIELD VALUE ESTIMATION

K. Suresh
The Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY 14853

J. Zagajac*
Process Leadership
Ford Motor Company
Dearborn, MI 48121

Abstract

The predominant engineering analysis techniques are finite element, finite difference, and boundary element methods. These techniques are global in that they attempt to solve field problems over an entire domain, and involve tedious pre-processing (meshing). In contrast, Monte Carlo (MC) methods are meshless techniques for estimating field values at discrete points. In these methods, one constructs random walks by repeated execution of few geometric tasks. These tasks, though robust, are perceived to be prohibitively expensive. Hence, in practice, MC methods are employed sparingly.

In this paper, we briefly describe a well-established MC method based on interior random walks. We identify the underlying geometric task: computing the minimum distance from interior points to the boundary of a domain. We then show, through theoretical and numerical results, that this task can be performed swiftly using certain graph-structures that are derived from the Voronoi diagram of a domain. When these graph-structures are employed, the MC method is relatively insensitive to domain complexity, making it an attractive alternative to global techniques.

1. Introduction

The objective of mechanical design is to identify a form and arrangement of mechanical parts that satisfy some prescribed design criteria. Compliance with design goals is usually established through analysis which often requires the solution of 'field problems' (partial differential equations augmented with boundary constraints). The predominant techniques for solving such problems are finite element, boundary element and finite difference methods (FEM, BEM and FDM). When the problem size is not very large, these techniques are efficient and powerful. However, their appeal quickly diminishes as the shape subject to field analysis becomes complicated. At least two reasons, originating from the global character of these methods, are responsible:

- **Growing preprocessing costs.** Designers often define the shape and configuration of mechanical parts by using geometric modeling (CAD) systems. Most of these systems ordinarily represent geometry via Boundary (b-rep) or Constructive Solid Geometry representations. The popular field analysis methods on the other hand require that the geometry be represented via cell decompositions. Thus, to do field analysis we must carry out a representational conversion. Such conversion - often called meshing or grid-generation - is expensive because it is entire (i.e. involves all of shape) and uses algorithms that typically grow as a polynomial function of geometrical complexity.

- **Growing analysis costs.** The central ingredients of field analysis are typically: (1) setting up, and (2) solving (often large) systems of linear equations that are prescribed for each component of the cell decomposition representing a shape. For complex shapes, setting up costs are not always negligible (particularly in BEM). More gravely, the polynomial complexity of algorithms commonly used to solve such systems quickly renders this task a difficult one.

Global solutions to field problems are wasteful when design decisions can be made based on the field solution at a few isolated parts of a domain. Under such circumstances, what is needed is an easy-to-use method that (1) produces estimates of field values only in a few critical regions, and (2) does not require expensive geometric preprocessing.

Monte Carlo methods for solving field problems are well known techniques based on statistical sampling that have the desired ingredients. These methods can estimate the value of the field at selected points without having to find the field over the entire region. Moreover, (at least in principle) they do not require extensive pre-processing. The main computational task that Monte Carlo methods pose - performing random walks - is geometrical in character and can be supported on any complete representation of solids. However, the overall efficiency of the method is greatly affected by the choice of the representation - the better the representation of geometry is suited for the task at hand, the more efficient the Monte Carlo method will be. It may be argued that Monte Carlo methods in engineering are not very popular in part because this reality is often overlooked.

* Work done while a Graduate Research Assistant in The Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY.
Statistical sampling in Monte Carlo methods for field problems comes in two flavors: random boundary and random domain sampling. These reflect the character of the underlying random walks that they entail: either on the boundary [Hoffman 74] or in the interior [Brown 56] of a domain respectively. In [Zagajac 96] it was argued that Ray-representations provide an efficient environment for performing random boundary sampling. In this paper we show how Voronoi graphs and their derivatives can be used to speed up random domain sampling. The processing enhancement that can be achieved by using these techniques provide sufficient power to tackle field problems in very complicated domains.

This paper is organized as follows. In Section 2 we briefly describe the domain based Monte Carlo method. We use Laplace's equation with Dirichlet boundary conditions to describe the main computational aspects of the technique and identify computational bottlenecks. In Sections 3 and 4 we show how random domain sampling can be performed effectively by using certain graph-structures. In Section 5 we illustrate the merits of our approach through numerical examples; in Section 6 we cite references where the domain based MC method is applied to other field problems, and employed to compute the field over a larger point set (e.g. a curve or a surface); in Section 7 we summarize the significance of this work.

2. Random Walk on Inscribed Balls

Let us consider a simple Dirichlet boundary value problem. Some physical property - say, temperature - is modeled by Laplace's equation in a connected subset \( \Omega \) of Euclidean space parametrized by position coordinates \( x \); see Figure 2-1. The field \( T(x) \) is prescribed on the piece-wise smooth boundary of the domain \( \partial \Omega \) by: \( T(x)=f(x), \forall x \in \partial \Omega \). We wish to determine \( T(x) \) at a point \( P \) in \( \Omega \).

![Figure 2-1: A Dirichlet boundary value problem.](image)

This problem can be solved by computing the field everywhere in \( \Omega \) using a method such as FEM, and then examining the result at \( P \). However, that seems wasteful if \( T(x) \) is not desired elsewhere in the domain. Alternatively, we can determine \( T(P) \) statistically by launching imaginary 'heat particles' from \( P \), and constructing the following sequence of random walks through the interior of the domain [Brown 56], [Muller 56], [Haji-Sheikh 66], [Haji-Sheikh 88].

1. At \( P \) we inscribe a ball \( B \) (disk in 2-D) centered at \( P \) so that \( B \) is completely contained in \( \Omega \). (Figure 2-2a).

2. On \( \partial B \) - the boundary of the ball \( B \) - we choose a point \( \pi_1 \) at random (see Figure 2-2b). We assume that all points on \( \partial B \) have an equal probability of being chosen. (The routine of inscribing balls in the domain and selecting a random point on their surface is called random domain sampling.)

3. If the shortest distance from \( \pi_1 \) to \( \partial \Omega \) is less than some small pre-determined value \( \varepsilon \), we terminate the walk at \( \pi_1 \) - the point on \( \partial \Omega \) closest to \( \pi_1 \), and record the value of the prescribed field \( f_1 \) at \( \pi_1 \). Otherwise, we inscribe a new ball \( B' \) centered at \( \pi_1 \) and repeat steps 2 and 3 replacing \( \pi_1 \) with new domain samples \( \pi_1', \pi_1'' \ldots \) (see Figure 2-2c).

4. We repeat steps 1, 2 and 3 for a large number of walks \( N \) and record the sampled field as \( f_2, f_3, \ldots \) (Figure 2-2d).

![Figure 2-2: Monte Carlo domain sampling algorithm.](image)

The radius of the inscribed ball constructed in steps 1 and 3 of the algorithm can be anywhere in the range \((0, R_{\text{max}}]\) where \( R_{\text{max}} \) is the shortest distance from the center of the ball to \( \partial \Omega \). A ball whose radius is equal to \( R_{\text{max}} \) is referred to as the maximal inscribed ball centered at a given point. Since the expected number of steps in a random walk decreases with the increasing size of the inscribed balls, using maximal balls at each step often results in computational savings.

1 These maximal inscribed balls are with respect to chosen centers; in the Voronoi diagram literature (to be discussed), maximal balls are defined differently.

2 The expected number of steps in a random walk is also a function of the domain geometry, dimension of the underlying space and the magnitude of \( \varepsilon \) [Muller 56].
The field at $P$ can be estimated from the collected samples by computing their mean:

$$T(P) = \frac{1}{N} \sum_{i=1}^{N} f_i \quad (2-1)$$

How accurate is this result? The stochastic nature of Equation (2-1) suggests that standard statistical tools can be used to answer this question. Suppose the prescribed field $f(x)$ on $\partial \Omega$ ranges between 0 and 1. Then assuming that $\varepsilon$ is sufficiently small, we can compute a coarse estimate of the magnitude of $N$ needed to determine $T(P)$ to within $\pm 0.01$ as follows. The variance of samples in Equation (2-1) will be at most $0.01$, and by the central limit theorem the error in $T(P)$ will be less than $0.01$ (with a $99.9\%$ confidence) if $N \geq (3\sqrt{1/4}/0.01)^2 = 10,000$ random walks. This value is independent of the dimension of the problem or the complexity of the domain (a remarkable feature of Monte Carlo methods).

In steps 1 and 3 of the algorithm, we compute the shortest distance from an interior point to the boundary of the domain. Since the boundary of the domain is usually a collection of piece-wise smooth faces, such as, edges in 2-D, the shortest distance computation amounts to finding the nearest boundary face. The nearest boundary face determination is the main computational load in random domain sampling. It is repeated many ($N \times$ average number of steps per walk) times. For example in 2-D, the above analysis suggests that the total number of random walks is of the $O(10^4)$, and the expected number of steps per random walk is of the $O(10)$ [Muller 56]. Therefore, the nearest boundary face computation will be repeated at least $O(10^5)$ times. The effectiveness of the Monte Carlo procedure hangs on our ability to minimize the cost of this calculation. In the next two Sections, we describe an efficient method for doing so.

3. Geometric Methods

In geometric modeling systems based on traditional representation schemes, such as boundary representation (b-rep), the nearest boundary face (NBF) task involves a linear search among all the faces of a domain, i.e., it involves computing and comparing the distances from an interior point to all the faces of $\partial \Omega$. This is computationally expensive since the NBF computation may be repeated many times, and is impractical when $\partial \Omega$ contains a large number of faces. Is it possible to do this computation more efficiently?

Consider the two domain walks illustrated in Figure 3-1a; observe that both walks are clustered around the starting point $P$. This statistical clustering is an inherent characteristic of the domain sampling technique, and implies that in many trials only a small subset of $\partial \Omega$ will play a role in the NBF computation; one such subset is highlighted in Fig. 3-1a. Thus a NBF search of the entire set $\partial \Omega$ is rather naive. We argue in this Section that the key to an improved NBF search lies in the clustering phenomena. By definition (of a Voronoi cell), it follows that the NBF of point $P$ is face $A$. Now consider an inscribed ball centered about $P$. One can argue that the ball intersects only a small portion of boundary faces $\partial \Omega$ contained in $\partial \Omega$ in $\partial \Omega$, and that a large portion of the circumference of the ball must typically lie in the Voronoi cell of $A$, as illustrated in Fig. 3-1c. Thus the NBF of the points on the circumference of the ball at $P$, is either face $A$ (with high probability) or one of the faces whose Voronoi cells are adjacent to the Voronoi cell of $A$. The notions of boundary faces and their Voronoi cells are made formal below.

3.1 Voronoi diagrams

For the purpose of this paper, we define a face of a domain $\Omega$ to be any open cell of dimension $n-1$ or less in a cell decomposition of $\partial \Omega$, where $n$ is the dimension of the space in which $\Omega$ resides. (An appropriate cell-decomposition of $\partial \Omega$ is one that facilitates boundary condition specification.) For example, in 2-D, all edges (devoid of vertices) and all vertices are considered to be faces, of dimension 1 and 0 respectively. We denote a face in $\partial \Omega$ by $\sigma$, a collection (set) of faces by $\Sigma$, and the number of faces in $\Sigma$ by $\# \Sigma$. Given a face $\sigma$ in $\partial \Omega$, and a point $p$ in $\partial \Omega \setminus \partial \Omega$, we define $D(p, \sigma)$ as the shortest (infimum) Euclidean distance from $p$ to $\sigma$.

---

3 The error introduced by the $\varepsilon$-criterion is proportional to the magnitude of field variation in the terminal inscribed ball.

4 A precise definition of a boundary face is provided in the next Section.
Now consider a non-empty collection $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ of faces. The Voronoi cell of $\Sigma$ - denoted by $V(\Sigma)$ - is defined to be the set of points $p$ in $\Omega \setminus \partial\Omega$ which satisfy the following two conditions.

1. $D(p, \sigma_1) = D(p, \sigma_2) = \ldots = D(p, \sigma_n),$
2. $D(p, \sigma) < D(p, \sigma_1), \forall \sigma \in \partial\Omega \setminus \Sigma.$

This definition is an extension of the definition of Voronoi cells of point sites in $E^d$ [Fortune 92]. As a special case, if $\Sigma$ consists of a single face $\sigma$, the first condition (above) is satisfied trivially, and the Voronoi cell $V((\sigma))$ - or, abusing the notation, $V(\sigma)$ - is the set of points in $\Omega \setminus \partial\Omega$ that are closer to $\sigma$ than to any other face in $\partial\Omega$. In Fig. 3-1b we illustrated the Voronoi cell of face $A$. Note that, by definition, (1) $V(A)$ does not contain face $A$ since face $A$ belongs to $\partial\Omega$, and (2) $V(A)$ does not contain the various bisectors since the bisectors are not strictly closer to face $A$.

A Voronoi cell of a face may be empty. For example, the Voronoi cell of any convex vertex in a 2-D polygon is always empty since there exists no point in $\Omega \setminus \partial\Omega$ whose nearest boundary face is a convex vertex. The Voronoi cell of $\Sigma$, where $\Sigma$ consists of two faces is either empty or is a subset of the bisector of the two faces, and so on. The Voronoi diagram of $\partial\Omega$ is the collection of all non-empty Voronoi cells $V(\Sigma)$, for all non-empty subsets $\Sigma$ of $\partial\Omega$. Figure 3-2 illustrates the Voronoi diagram of the $L$-bracket; note that only seven faces, out of the total twelve (six edges + six vertices) in $\partial\Omega$ contribute to the Voronoi diagram; these seven faces are labeled A through G in Fig. 3-2, the remaining five faces are the non-contributing convex vertices. Observe in Fig. 3.2 that the Voronoi cell $V((A, B))$ is a line segment, and $V((A, F, G))$ is a point. In general, the dimension of a non-empty cell $V(\Sigma)$ is at most $n+1 - |\Sigma|$, where $n$ is the dimension of the space in which $\Omega$ resides.

Voronoi diagrams exhibit a number of interesting and useful properties. We study a few relevant ones in this paper. For example, it can be easily shown that every point in $\Omega \setminus \partial\Omega$ belongs to an unique Voronoi cell. Moreover, by definition, if a point belongs to $V(\Sigma)$, the set of faces nearest to that point in $\partial\Omega$ is $\Sigma$. In Figure 3-3, point $p$ belongs to $V(A)$, thus the NBF(p) is $\{A\}$, and point $q$ belongs to $V((A, B, E))$, thus the NBF(q) is $\{A, B, E\}$.\(^5\)

\(^5\) Henceforth, NBF(x) will denote the set of faces in $\partial\Omega$ nearest to x.

3.2 Maximal graphs

Let us now consider a domain walk originating from point $p$ in Fig. 3-3. In step 1 of the algorithm described in Section 2, we inscribe a ball (disc in 2-D) about $p$, then in step 2 we select a domain sample $\pi_1$ on the circumference of the ball, and in step 3 we find the NBF of $\pi_1$. Since every inscribed ball is contained in the maximal ball at $p$, $\pi_1$ must lie within the maximal ball centered about $p$. In Figure 3-4, we see that the maximal ball at $p$ intersects only the Voronoi cells involving faces $A$, $F$, and $G$. Thus the NBF($\pi_1$) must be a subset of $\{A, F, G\}$.

On the other hand, the maximal ball at $q$ intersects the Voronoi cells involving faces $A$, $B$, $D$, $E$, and $F$ (Fig. 3-4). Thus the NBF of a point successive\(^6\) to point $q$ in a random walk, must be a subset of $\{A, B, D, E, F\}$. A crucial observation to make here is that there exists no point in $V(A)$ such that the maximal disc centered about that point intersects a Voronoi cell involving face $C$. One can verify this from brute force construction of the union of all maximal balls centered in $V(A)$.

\(^6\) By a successive point to $q$, we mean a point on the circumference of an inscribed ball at $q$. 
and checking if this set - referred to as the maximal cell of A - intersects a Voronoi cell involving face C; the construction of this set is illustrated in Figure 3-5.

Thus if a domain sample \( \pi^m_n \), at any step, falls within \( V(A) \), then the NBF of the successive point to \( \pi^m_n \), i.e., the NBF of \( \pi^m_{n+1} \), cannot contain face C. This can be captured succintly by assigning nodes to the seven faces A through G, and constructing a set of directed arcs from node A to all the nodes, except node C, as illustrated in Figure 3-6. Fig. 3-6 states that if, at any step in the walk, one enters \( V(A) \), then, one can reach all Voronoi cells except cell \( V(C) \) in the next step. This abstraction is referred to as the maximal connectivity of A; nodes A, B, D, E, F and G are said to be the maximal neighbors of node A.

The superposition of the maximal connectivities of all the nodes results in a (directed) maximal graph; the maximal graph of the \( L \)-bracket is illustrated in Figure 3-7. The maximal graph in Fig. 3-7 is a simplified one: bidirected arcs have been substituted with undirected arcs. Such a simplification may not always be possible, and it is done here for convenience and clarity.

To generalize, given any face \( \sigma \) in \( \partial \Omega \), such that \( V(\sigma) \) is not empty, one can define a maximal cell of \( \sigma - M(\sigma) \) - as the union of all maximal balls centered in \( V(\sigma) \). Moreover, given two faces \( \sigma_1 \) and \( \sigma_2 \) with non-empty Voronoi cells, \( \sigma_2 \) is said to be a maximal neighbor of \( \sigma_1 \) if and only if \( M(\sigma_1) \cap V(\sigma_2) \neq \emptyset \). Finally, the (directed) maximal graph of \( \partial \Omega \) is defined as follows: the nodes of the graph correspond to the faces in \( \partial \Omega \) with non-empty Voronoi cells, and a directed arc is drawn from node \( \sigma_1 \) to node \( \sigma_2 \) if and only if \( \sigma_2 \) is a maximal neighbor of \( \sigma_1 \). Observe the asymmetry in the definition: a directed arc from node \( \sigma_1 \) to node \( \sigma_2 \) does not imply the existence of a directed arc from node \( \sigma_2 \) to node \( \sigma_1 \). maximal graphs are directed graphs. With this definition, we can now make an important claim whose proof follows from the definition of maximal neighbors.

Lemma 3.1: If \( (\sigma) \) is the NBF of a domain sample \( \pi^m_n \) in a random walk, then the NBF of \( \pi^m_{n+1} \) - \( \pi^m_{n+1} \) being the successive point to \( \pi^m_n \) - is a subset of the maximal neighbors of \( \sigma \).

The above claim allows swift computation of (successive) NBF, provided the maximal graph of \( \Omega \) is known. But, how does one construct maximal graphs, and what are the associated costs? Maximal graph construction is computationally expensive if one explicitly constructs and intersects the various Voronoi and maximal cells (as suggested by the definition). Is it possible to generate the graphs directly, without explicitly constructing the various cells? For example, is it possible to generate a maximal graph by exploiting the converse of Lemma 3.1?

4. Practical Surrogates to Maximal Graphs

In this paper, we do not pursue the problem of maximal graph generation; instead we search for surrogate graphs - graphs that are 'similar' to maximal graphs, but are easier to compute. Such surrogates are sometimes constructed in an ad hoc manner. For example, [Pickles 77] suggests using simple geometric rules to maintain a list of 'relevant' boundary faces

7 By convention, \( \pi^m_{n+1} \) is a successive point to \( \pi^m_n \).

8 Observe that the maximal graph boosted algorithm does not require the individual Voronoi and maximal cells.
during domain sampling. No formal claims can be made on the efficiency of the NBF computation using such ad hoc constructions. We introduce below a particular surrogate, namely, a Voronoi graph, and study (1) its formal relationship to a maximal graph, (2) its role in the domain sampling algorithm, and (3) its computability.

4.1 Voronoi graphs

Two faces $\sigma_1$ and $\sigma_2$ of $\partial \Omega$ with non-empty Voronoi cells, are defined to be Voronoi neighbors if and only if $V((\sigma_1, \sigma_2)) \neq \emptyset$. By convention, $\sigma$ is considered to be a Voronoi neighbor of itself. A Voronoi graph of $\partial \Omega$ is then defined as follows: the nodes of the graph correspond to the faces of $\partial \Omega$ with non-empty Voronoi cells, and an undirected arc is drawn between nodes $\sigma_1$ and $\sigma_2$ if and only if $\sigma_1$ and $\sigma_2$ are Voronoi neighbors. In 2-D, this means that every Voronoi 'edge' corresponds to an arc in the graph. Thus the Voronoi graph of the $L$-bracket, determined from Fig. 3-2, is illustrated in Figure 4-1.

![Voronoi graph of the $L$-bracket.](image)

Observe that the Voronoi graph in Fig. 4-1 is a subset of the maximal graph in 3-7. This follows from the fact that if $V((\sigma_1, \sigma_2)) \neq \emptyset$, then $M(\sigma_1) \cap V(\sigma_2) \neq \emptyset$, and $M(\sigma_2) \cap V(\sigma_1) \neq \emptyset$, i.e., every arc of a Voronoi graph of $\partial \Omega$ corresponds to a bidirected arc in a maximal graph of $\partial \Omega$, but not vice-versa. The differences between these two graphs are best understood by considering regular n-gons in 2-D. For example, consider the hexagon, and its Voronoi diagram illustrated in Figure 4-2a; its maximal graph - a 'complete' graph - is illustrated in Figure 4-2b, and its Voronoi graph in Figure 4-2c. (Incidentally this example also illustrates that an maximal graph based domain sampling algorithm for regular n-gons is computationally no better than a naive linear search algorithm.)

Since Voronoi graphs are subsets of maximal graphs, the claim - Lemma 3.1 - on maximal graphs is not applicable to Voronoi graphs. So, how does one exploit Voronoi graphs to speed up the domain sampling algorithm? Before we make a relevant claim, we state an important property of Voronoi graphs.

**Property 4.1:** Let $p$ be a point in $\Omega \setminus \partial \Omega$, let $\sigma$ be a face in $\partial \Omega$, and let the shortest line segment from $p$ to $\sigma$ be contained in $\Omega$. Then, $p \in V(\sigma)$ if $p$ is closer to $\sigma$ than to all the (other) Voronoi neighbors of $\sigma$.

**Proof (Sketch):** Let us assume that $p$ is closer to $\sigma$ than to all the (other) Voronoi neighbors of $\sigma$, but $p \notin V(\sigma)$. This, we show below, will lead to a contradiction.

Let $L$ be the shortest line from $p$ to $\sigma$ that is contained in $\Omega$. Since $p \notin V(\sigma), L$ must intersect the boundary of $V(\sigma)$; let the point of intersection be $q$. Now $q$ must belong to $V(S)$, for some $S = \{\sigma, \sigma_1, \sigma_2, ..., \sigma_n\}$. For all $\sigma_i \in S$, it follows from the triangle inequality, that

$$D(p, \sigma_i) \leq D(p, q) + D(q, \sigma_i).$$

But since $q \in V((\sigma, \sigma_1, \sigma_2, ..., \sigma_n))$,

$$D(q, \sigma_i) = D(q, \sigma),$$

$$D(p, \sigma_i) \leq D(p, q) + D(q, \sigma).$$

Since $q$ lies on $L$ (the shortest line from $p$ to $\sigma$),

$$D(p, q) + D(q, \sigma) = D(p, \sigma),$$

$$D(p, \sigma) \leq D(p, \sigma)$$

On the other hand, since $V(S)$ is not null there must exist a face $\sigma_j$ in $S$ which is a Voronoi neighbor of $\sigma$. And, we are given that point $p$ is closer to $\sigma_j$ than to all the Voronoi neighbors of $\sigma$, i.e.,

$$D(p, \sigma) < D(p, \sigma_j)$$

But, this contradicts the previous conclusion. Thus, we conclude that $p \in V(\sigma)$.

(End of proof.)

![Voronoi diagram of a hexagon.](image)

![Its maximal graph.](image)

![Its Voronoi graph.](image)

**Figure 4-2:** A hexagon, its maximal graph, and Voronoi graph.

The above property states that a linear search (among all the faces of $\partial \Omega$) is not required to confirm if a particular face $\sigma$ is the NBF of a given point $p$. In Section 4.3, we argue that the number of Voronoi neighbors is usually much smaller than the total number of faces in $\partial \Omega$, implying that the confirmation test is typically inexpensive. However, Property 4.1 is

---

9 We make a few 'reasonable' assumptions here; a formal proof of these assumptions is beyond the scope of this paper, and will appear elsewhere. Moreover, a formal proof of a closely related property of Voronoi graphs appears in [Zagajac 97].
Fortunately, in a random walk, we can exploit once again the clustering phenomena: there is a 'good chance' that two successive samples, \( \pi_n^m \) and \( \pi_n^{m+1} \) share the same NBF. Keeping this in mind, we make the following claim (a special case of Property 4.1).

**Lemma 4.1:** If \( \{\sigma\} \) is the NBF of a domain sample \( \pi_n^m \) in a random walk, then the NBF of \( \pi_n^{m+1} \), \( \pi_n^{m+1} \) being the successive point to \( \pi_n^m \), is also \( \{\sigma\} \) provided \( \pi_n^{m+1} \) is closer to \( \sigma \) than to the Voronoi neighbors of \( \sigma \).

Observe that this claim is weaker than the maximal graph based claim (Lemma 3.1) in that Lemma 4.1 is merely a confirmation, but not a determination, test. However, as with Lemma 3.1, Lemma 4.1 can be applied to each (except the starting) point of a domain walk. If the test is affirmative, then we have determined the NBF of a successive point. However if a check based on Lemma 4.1 is not affirmative, i.e., given a domain sample \( \pi_n^{m+1} \) and a test candidate \( \sigma \), if \( \{\sigma\} \) is not the NBF of \( \pi_n^{m+1} \), then Lemma 4.1 does not explicitly provide a solution. But if Lemma 4.1 is not affirmative, then we have found a Voronoi neighbor of \( \sigma \), say \( \sigma' \), such that \( \pi_n^{m+1} \) is closer to \( \sigma' \) than to \( \sigma \). This implies that \( \{\sigma'\} \) is a better candidate for the NBF of \( \pi_n^{m+1} \) than \( \{\sigma\} \); Lemma 4.1 can be employed once again to confirm if \( \{\sigma'\} \) is indeed the NBF of \( \pi_n^{m+1} \). Thus a Voronoi graph based search is recursive unlike a maximal graph based search. This is illustrated in Figure 4-3.

![Figure 4-3](image)

**Figure 4-3: Voronoi graph based successive NBF computation.**

One can argue that the Voronoi graph based search must always converge (in a finite number of steps) to the correct NBF, and is computationally no worse than a naive linear search. (The Voronoi graph based search typically converges in a few, say 1 or 2, iterations.)

## 4.2 Voronoi graph computation

The Voronoi graph based domain sampling algorithm entails the computation of the Voronoi graph as a preprocessing step. Presently, there are two broad strategies by which Voronoi graphs can be computed: (1) a 'diagram' approach, wherein the Voronoi diagram is first computed either exactly [Srinivasan 87], [Dutta 90] or approximately [Sudhalkar 93], [Reddy 95], followed by the graph construction, and (2) a 'Delaunay triangulation' approach wherein a domain Delaunay triangulation [Sapidis 91], [Yu 91] is first carried out, from which the Voronoi graph is deduced. In this paper, we employ the diagram approach for the numerical experiments (discussed in Section 5), and the domain Delaunay triangulation approach to generate Table 4-1 below.

A slight digression is in order at this point. Observe that neither the Voronoi diagram nor the Delaunay triangulation is required in the sampling algorithm. We are therefore exploring a third 'direct' approach for computing Voronoi graphs. In this approach, points are sampled either randomly or systematically from within the domain, and based on certain properties of these points, one can directly deduce the graph connectivity; this work is expected to appear in the literature at a later time. Moreover, due to the very nature of the MC domain sampling algorithm, a large number of nodes in a Voronoi graph are never visited, i.e., only subsets of a Voronoi graph are utilized. It appears wasteful to compute the entire graph when partial graphs are sufficient; partial Voronoi graph generation is another promising research avenue that we are pursuing.

## 4.3 Average connectivity of Voronoi graphs

The performance of the Voronoi graph based random sampling algorithm (Fig. 4-3) clearly depends on the average connectivity of the graph, where average connectivity is defined as the number of emanating arcs from a node, averaged over all nodes.

An important property of Voronoi graphs of 2-D objects is that the average connectivity does not exceed 10.\(^{10}\) It is important to note that this is independent of the number of nodes in the graph.

This property does not hold true in 3-D, i.e., theoretically, the average connectivity of a Voronoi graph in 3-D, can be linearly dependent on the number of nodes in the graph. However, a probabilistic analysis, based on point sites in 3-D, seems to suggest that the average connectivity of Voronoi graphs of 3-D objects does not exceed 20 [Dwyer 89], [Fortune 92]. We verified this claim by computing the Voronoi graphs of various 3-D components by: (1) first computing a domain Delaunay triangulation [Sapidis 91] of the objects (using TGrid\textsuperscript{TM}, a commercial mesh-generator available from Fluent Inc.), and (2) parsing the triangulation data to deduce the

---

\(^{10}\) The proof consists of two essential steps: (1) embed a 2-D Voronoi diagram on the surface of a 3-D sphere, resulting in a 3-D polyhedron, and (2) apply 3-D Euler's formula to deduce the maximum number of edges in the polyhedron.
Voronoi connectivity [Yu 91]. Table 4-1 summarizes the average connectivity of the associated Voronoi graphs.\footnote{This work was done at the Ford Motor Company, and due to proprietary reasons, we are unable to furnish further details on the various engine components.}

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Average connectivity of Voronoi graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3-D) L-bracket</td>
<td>8</td>
</tr>
<tr>
<td>Water pump pulley</td>
<td>27</td>
</tr>
<tr>
<td>Connecting rod</td>
<td>74</td>
</tr>
<tr>
<td>Throttle valve</td>
<td>277</td>
</tr>
<tr>
<td>Cylinder block</td>
<td>686</td>
</tr>
</tbody>
</table>

Table 4-1: The average connectivity of Voronoi graphs of a few 3-D objects.

5. Numerical Experiments

In this Section we substantiate the theoretical results of Section 4 with numerical experiments. These experiments are restricted, without loss of generality, to 2-D linear polygonal objects. First, we investigate the computational cost of Voronoi graph generation for a few 2-D benchmark objects. Then we investigate Voronoi graph based MC domain sampling; for comparison, we also investigate the traditional b-rep based MC domain sampling.

5.1 Voronoi graph computation

The various benchmark objects that were considered for the numerical experiments are illustrated in Figure 5.1. These objects are in the increasing order of the number of effective\footnote{A face is effective if its Voronoi cell is non-empty.} faces, and are labeled 'a' through 'f'.

Fig. 5.1 also illustrates the Voronoi diagrams of these objects which were computed using the algorithm described in [Srinivasan 87], and [Meshkat 87]; the complexity of this algorithm is of the $O(N^* \max\{\log_2 N, H\})$, where $N$ is the total number of effective faces in the object (or nodes in the graph), and $H$ is the number of holes in the object. When $N$ is not too large, say less than $O(10^5)$ as in our experiment, the algorithm is of linear complexity. In our experiment, the constant of proportionality (on a DEC 3000/600 APP workstation) was approximately 1 CPU second per 500 effective faces, and thus the Voronoi diagram computation required at most 2 CPU seconds for the various objects considered. Once the diagram is obtained, the computation of the Voronoi graph is trivial.

5.2 Domain sampling

We then compared the domain sampling rate - the number of domain samples computed per second in the MC algorithm using the two representation schemes: Voronoi graphs and b-reps. The 2-D linear polygons illustrated in Fig. 5-1 served as benchmark objects. Starting points were chosen at random from the interior of these objects, and the MC domain sampling algorithm was executed from each of these starting points. (The Voronoi graph based algorithm exploits Lemma 4-1.) Figure 5-2 summarizes the results; all computing was done on a DEC 3000/600 APP workstation, and the 'seconds' on the y-axis of Fig. 5-2 denote CPU time.

(a) Object with 7 faces.  
(b) Object with 12 faces.

(c) Object with 15 faces.  
(d) Object with 57 faces.

(e) Object with 170 faces.  
(f) Object with 804 faces.

Figure 5-1: The various benchmark objects, and their Voronoi diagrams.

Observe in Fig. 5-2 that the naive b-rep sampling shows a linear drop in performance as expected, and the sampling rate was approximately 350,000 samples per CPU second per face. On the other hand, the Voronoi graph based sampling is almost independent of the complexity of the object (the pre-processing cost is not included), and the sampling rate is approximately 40,000 domain samples per CPU second. The shape independent performance of the latter is a direct consequence of the fact that the average connectivity of the Voronoi graphs of 2-D objects is almost a constant (Section 4.3).
5.3 Discussion

To summarize, recall from Section 2, that reasonably accurate point solutions require approximately $10^5$ domain samples. Thus to obtain such a solution in 2-D polygonal objects:

- the b-rep based cost is approximately $10^5((350,000/N))$, i.e., 0.28*N CPU seconds, where N is the total number of effective faces;
- the pre-processing cost of Voronoi graph generation is approximately (N/500), i.e., 0.002*N CPU seconds, and the sampling cost is (10^5/40,000), i.e., 2.5 CPU seconds, independent of N; the total cost is $0.002*N + 2.5$ CPU seconds.

Through a quick comparison, one can check that a Voronoi graph based domain sampling is far superior for N > 10. It is interesting to note that the pre-processing cost of Voronoi graph generation far exceeds the sampling cost for large N. This merely strengthens the arguments of Section 4.2 made in favor of direct (and perhaps partial) Voronoi graph generation.

6. Generalizations of the MC Technique

The Monte Carlo point solution technique discussed here is not restricted to the solution of the Laplace equation with Dirichlet boundary conditions; it can be used to find point solutions to a wide variety of field problems. For example, the generalization of the technique to Neumann boundary conditions is described in [Zagajac 97]; mixed boundary conditions are addressed in [Burmeister 85]. The generalization of the method to problems in elasticity is discussed in [Vorshko 73], [Pobedrya 88]. The biharmonic equation is considered in [Sabelfeld 88]. A few non-linear field problems also have been addressed: the Burgers equation in [Roberts 89], Kolmogorov equation in [Puckett 89]. The various solution procedures differ mainly in the domain sampling strategy, and in the character of the estimators that are accumulated. However, all these procedures share a common geometric utility viz. computing the nearest boundary face, and can therefore exploit the graph-structures discussed in this paper to speed up the computation.

One significant drawback of the Monte Carlo point solution technique is its discrete character. Although discrete point-values are occasionally useful in engineering analysis, they fall short of helping us in the analysis of problems where a field is required over a larger point set (e.g. a curve or a surface). However, the described point technique can be used as a building block to devise more powerful Spectral Monte Carlo methods that permit us to obtain regional solutions to boundary value problems. These generalizations are described in [Zagajac 97).

7. Conclusions

This paper illustrates an approach for coupling CAD and engineering analysis that avoids the pitfalls of traditional techniques that use mesh generation. By relaxing the expectations of engineering analysis (global vs. point solutions), we were able to identify an old Monte Carlo (MC) technique that permits us to solve field problems at discrete points in domains, without meshing, and without inverting large matrices. The technique is characterized by a single geometrical procedure: random domain sampling.

In Sections 3, 4 and 5, we argued that random domain sampling can be performed swiftly using certain graph-structures derived from the Voronoi diagram of a domain. When these graph-structures are employed, the MC method is relatively insensitive to domain complexity, and can be employed to analyze field problems in complex domains. We also concluded, based on numerical experiments, that if a Voronoi diagram is explicitly constructed, the overhead cost of (the graph construction) may exceed the sampling cost; alternate means of generating these graphs are desirable, and are being investigated.

8. Acknowledgments

We would like to thank Herb Voelecker and Vijay Srinivasan for both technical and editorial suggestions. We would also like to thank the authors of [Srinivasan 87] and [Meshkat 87] for providing the software for generating Voronoi graphs of 2-D polygons. Finally, thanks to the reviewers for their constructive comments. The research reported here was supported by the National Science Foundation under Grants MIP-90-07501 and MIP-93-17620, and by IBM under a cooperative fellowship.

9. References


[Dutta 90] Dutta, D., and Hoffmann, C. M., "A geometric investigation of the skeleton of CSG objects", Proc. of


