Meshing $\log n$ Dimensions in Polynomial Time[*]

Gary L. Miller [†] Donald R. Sheehy [‡] Ameya A. Velingker [§]

Abstract

We present a new algorithm that produces an approximation of the 1-dimensional skeleton of a Delaunay mesh (or its dual Voronoi diagram) for point sets in any dimension with guaranteed optimal mesh size and quality. Our comparison based algorithm runs in time $O(2^{O(d)}(n \log n + m))$, where n is the input size, m is the output point set size, and d is the ambient dimension. The constants only depend on the desired element quality bounds.

1 Introduction

The meshing problem is closely linked with numerical solutions to PDEs arising in scientific computing, particularly from finite element analysis (FEA) [1]. However, it would be a mistake to only equate the tool, meshing, and the application, FEA. In particular, there is no reason to limit general meshing techniques to two- and three-dimensional problems. The same features that make meshing useful in FEA also make it useful for geometric and topological data analysis in higher dimensions.

A mesh provides a basis for piecewise linear functions on a geometric space. This is useful when approximating one or more functions on the space and is particularly useful when the functions are relatively smooth (Lipschitz functions, for example). The graded meshes we discuss here have the added benefit that they are adaptive to the local density. Thus, they can give higher fidelity approximations in some areas and lower fidelity in others, increasing space efficiency. In geometric and topological data analysis, a standard technique is to consider distance-like functions induced by a set of points in $\mathbf{R}^d$. These functions are generally Lipschitz, and there are often regions of varying density.

Another application of high dimensional meshing is robotics, specifically motion planning in a high-dimensional configuration space [4].

[*]Partially supported by the National Science Foundation under grant number CCF-1065106.

[†]Computer Science Department, Carnegie Mellon University glmiller@cs.cmu.edu

[‡]INRIA, don.r.sheehy@gmail.com

[§]Computer Science Department, Carnegie Mellon University avelingk@cs.cmu.edu

2 Past Work

Hudson, et. al. devised the Sparse Voronoi Refinement (SVR) algorithm [5] which produces a size-optimal quality Voronoi diagram in output-sensitive time $O_d(n \log \Delta + m)$, where n is the number of input points, m is the number of output points, and Δ is the **spread**, the ratio of the diameter of the input set to the smallest pairwise distance between two disjoint features of the input. The notation O_d signifies the fact that the constant is allowed to depend on the dimension d. However, Δ can be arbitrarily large for fixed n.

Subsequently, Miller, et. al. presented the NetMesh algorithm [7] which runs in $O_d(n \log n + m)$ time. It manages to remove the dependence on spread by optimizing point location and is the first algorithm that is work-optimal in a comparison-based model.

One of the shortcomings of the NetMesh algorithm lies in the constant that is hidden by the O_d term. In particular, the algorithm stores uninserted input points in Delaunay balls, whose centers are corners of Voronoi cells. Since a Voronoi cell can have up to $2^{O(d^2)}$ corners, this yields a $2^{O(d^2)}$ dependence on d, which is unsuitable for meshing in high dimensions. For this reason, any algorithm which stores the full Voronoi simplices can be expected to suffer from the $2^{O(d^2)}$ dependence.

Our newly proposed algorithm avoids storing the corners of Voronoi cells and therefore circumvents the $2^{O(d^2)}$ dependence. By using a preprocessing step to deal with point location, we expect to be able to create a quality mesh in $O(2^{O(d)}(n \log n + m))$. In many useful data-driven applications, one is concerned with a set of n points in high-dimensional space which, by the Johnson-Lindenstrauss Lemma [6], may be embedded with low distortion into a smaller space of dimension roughly $d \approx \log n$. Hence, in this case, our algorithm would produce a mesh in output-sensitive polynomial time. This would constitute a large improvement over the time complexity of existing algorithms.

Another related notion that has been explored in the past is the approximate Voronoi diagram [2], which is comparable to Steiner point methods. However, we have adopted a different approach here.

3 Proposed Algorithm

The basic structure of our proposed algorithm is as follows: First, one constructs a **greedy permutation** of the input points. The points are then inserted one at a time, yielding an **incremental construction** of the mesh. However, instead of maintaining the mesh directly, we approximately maintain its dual, the Voronoi diagram. After each input point insertion, Steiner points are added as part of a **refinement procedure** to retain mesh quality. The refinement procedure involves using a random sampling based LP method to find Voronoi cells with bad aspect ratio. As points are inserted into the mesh, a **point location** data structure containing uninserted points is updated. The use of the greedy permutation cuts down on the number of exposed input points we keep in the point location data structure while retaining enough points to avoid the need for backtracking within the algorithm.

3.1 Point Location

A major component of our algorithm is the new point location method. We keep uninserted points in Voronoi cells of the incremental mesh at each step. Suppose we are given a set of input points $P \subset \mathbf{R}^d$ for which we wish to compute the Delaunay mesh. We preprocess the given input set by permuting the points of P to produce the greedy permutation, which is the ordering $p_1, p_2, \ldots, p_n$ for which each p_i is the farthest point in P from its predecessors. We also require a **predecessor pairing** ϕ that assigns each p_i to its nearest neighbor in $P_{i-1} = \{p_1, p_2, \ldots, p_{i-1}\}$. A good approximation to the greedy permutation along with the pairing ϕ can be computed in $2^{O(d)} n \log n$ time using net trees [3].

We define the radius of a mesh point p_i as

$$r_i = \mathbf{d}(p_i, \phi(p_i)).$$

It can be shown that $r_i \leq r_j$ for $i > j$.

For any nonnegative real number s, let $Q(s)$ denote all points of radius at least s, i.e.,

$$Q(s) = \{p_i \in P : r_i \geq s\}.$$

By the aforementioned property, for each s there exists an $i \in \{1, 2, \ldots, n\}$ such that $Q(s) = P_i$. No two points of $Q(s)$ are closer than s from each other, and no point of $P \setminus Q(s)$ is farther than s from a point of $Q(s)$.

The data structure for storing uninserted points is very simple. At any time in the algorithm, if s denotes the smallest edge length in the mesh, then the points of $Q(\epsilon s)$ are stored in the point location data structure. The rest of the uninserted input points are designated as "hidden points."

Each point in the data structure is associated with the Voronoi cell of the mesh that it lies inside. Thus, every time we add an input point p to the point location data structure, we shoot a ray from $\phi(p)$ to p and carefully keep track of crossings across Voronoi cells to find the Voronoi cell of the mesh in which p is contained.

We can expect the following invariant: **At any time, no Voronoi cell stores more than a constant number of points.** This can be shown using a packing argument, since the stored points inside a Voronoi spaced are evenly-spaced. Thus, each insertion into the mesh should touch at most a constant number of points.

3.2 Maintenance of Mesh

While running our algorithm, we maintain a mesh of points. However, since we do not store the corners or topology of Voronoi cells, we actually maintain a superset of the true Voronoi neighbors of each mesh vertex. It is this design choice which allows us to get by with a $2^{O(d)}$ dependence on d. In particular, for any vertex p in the mesh, q is stored as a "neighbor" of p if two separate conditions are satisfied, namely **density** and **sparsity**. These conditions guarantee that any stage, we are storing $2^{O(d)}$ vertices in the neighbor superset.

The aforementioned structure can be pruned in $\left(\frac{1}{\epsilon}\right)^{O(d)}$ time to an ϵ-approximate Delaunay graph, as defined below:

Defnition 1 *An ϵ-approximate Delauany graph contains all edges that appear in the true Delaunay skeleton or could appear if the vertices are perturbed by some a factor of ϵ times the local feature size.*

4 Conclusion

We have proposed an outline for an algorithm to compute an approximate Delauany mesh for point set inputs. It would be interesting to consider how to adapt the algorithm to be able to mesh inputs that also contain higher dimensional features.

References

[1] P. L. George and H. Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermes, 1998.

[2] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. *Proc. 10th ACM-SIAM SODA*, 1999.

[3] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SICOMP*, 35(5):1148–1184, 2006.

[4] Y. Huang and K. Gupta. A Delauany triangulation based node connection strategy for probabilistic roadmap planners. *Proc. ICRA*, 2004.

[5] B. Hudson, G. Miller, and T. Phillips. Sparse Voronoi refinement. *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, 2006.

[6] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[7] G. Miller, T. Phillips, and D. Sheehy. Beating the spread: Time-optimal point meshing. *SoCG*, 2011.