NSF Graduate Research Fellowship Previous Research

Bill March

1 Research Overview and Interests

Current work. Since summer 2006, I have worked as a research assistant for Alex Gray's FASTlab. I have explored fast algorithms for problems in computational geometry, including convex hulls, path finding, and minimum spanning trees. I have spent much of my time investigating scientific applications for these algorithms, especially in biology and chemistry. As a graduate student, I am investigating ways to apply my earlier techniques to designing faster and more accurate computational chemistry algorithms.

Past research. Previously, during Summer 2005, I participated in the NSF Research Experience for Undergraduates program at Georgia Tech. I studied online graph coloring algorithms with Prof. Trotter in the School of Mathematics. This experience gave me valuable experience with reading papers, organizing a complex research topic, and exploring new ideas. However, I found mathematics too far removed from the scientific applications I was interested in, which led me to my current work.

2 Euclidean Minimum Spanning Trees

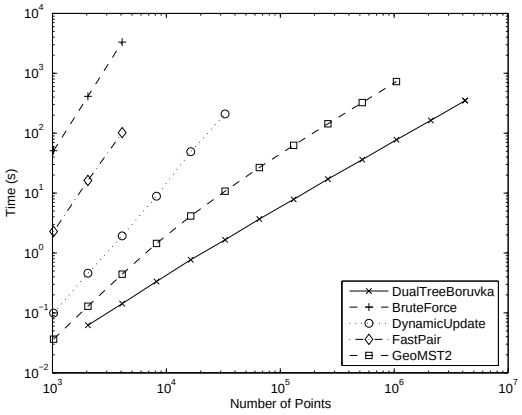
Context. Euclidean Minimum Spanning Trees are fundamental structures in computational geometry and are widely applied in network design, optimization, and computer vision. Additionally, the EMST is used to find a hierarchical clustering of the underlying points. This method of clustering is commonly applied throughout science and is particularly popular for clustering gene microarray data and cosmological surveys.

The EMST problem is well-studied, and there are effective algorithms for many cases, such as in two dimensions. However, none of the existing algorithms are truly scalable to massive data sets and arbitrary dimensions. New algorithms are necessary to handle large-scale data gathering efforts, such as the Sloan Digital Sky Survey and the Human Genome Project.

Teamwork. As an initial research problem with his lab, Dr. Gray suggested it might be possible to apply some of his earlier work on fast machine learning algorithms to the EMST problem. Since I was new, I relied on discussions with his graduate students to become familiar with our existing approaches and learn the lab's code base. Throughout the project, I often discussed ideas and difficulties with Alex and the other students in the lab. These interactions often allowed me to find ways around difficulties.

Individual work. Although others helped as needed, I did most of the work individually. I searched the existing literature to determine the best existing algorithms. I determined the link between the previous work and the EMST problem and worked out the details of my own EMST algorithm. I wrote the code for my algorithm and the competitors, then organized and carried out the comparisons. I have submitted a conference paper on my algorithm which is currently under review.

Algorithm details. I applied an existing framework, known as *generalized N-body problems* [1], to finding EMST's. This framework has been previously used to create fast algorithms for many problems in statistics and machine learning including the N-point correlation, all-nearest-neighbors, and kernel density estimation. Using these ideas, I was able to design and implement the fastest existing algorithm for EMST's in any dimension.



(a) Log-log scale runtimes on synthetic clustered data.

dim	N	GeoMST2	DTB	Speedup
3	389354	78.0	16.9	**4.6**
12	320000	702	62.3	**11.3**

(b) Runtimes for SDSS data and protein folding trajectories. GeoMST2 is the previous fastest EMST algorithm. DTB stands for my DualTreeBoruvka algorithm.

My algorithm, DualTreeBoruvka, applies multi-tree recursion to Borůvka's algorithm for finding MST's. Borůvka's algorithm is similar to Kruskal's, in that it maintains a spanning forest and iteratively connects components to build the tree. While Kruskal's algorithm connects the closest two components in each step, Borůvka's connects each component with its nearest neighbor. I used the space-partitioning trees and multi-tree recursion ideas to quickly compute the nearest neighbor of each component.

The algorithm uses a *kd-tree* to organize the points. Each node of the tree consists of a hyper-rectangle or bounding box containing a subset of the data. The root node contains the entire data set. Children are created as follows: choose the longest dimension of the current node's bounding box, partition the data along the midpoint in this dimension, and form two new, smaller bounding boxes to cover the subsets. The tree forms leaves when a node contains fewer than some specified number of points.

Dual-tree Recursion. The simplest way to compute the nearest neighbor of a component Q is to compute the distances from all points $q \in Q$ to all points $r \in \overline{Q}$ and keep the smallest. For each q, we store the distance to the nearest neighbor found so far, $d^u(q)$, as an upper bound on the true distance. If, for any point $r, d(q,r) > d^u(q)$, then we can be confident r is not the nearest neighbor of q.

Using the *kd*-tree, it is possible to improve on this method. Instead of iterating over the points $r \in \overline{Q}$, compare a point q with an entire node R. Using the bounding box of R, the algorithm can compute a lower bound $d^l(q,R) < d(q,r)$ for all $r \in R$. Then, if $d^l(q,R) > d^u(q)$ for some R, it can *prune* any further consideration of points in R. Otherwise, recursively consider the two children of R.

This idea can be extended even further. Since Borůvka's algorithm needs the nearest neighbor of each component Q, my algorithm exploits the fact that these points are also grouped in the *kd*-tree. We can maintain an upper bound $d^u(Q)$ on $d^u(q)$ for all points $q \in Q$. We compare nodes Q and R, compute the minimum distance between their bounding boxes $d^l(Q,R)$, and prune if $d^l(Q,R) > d^u(Q)$. Otherwise, we recursively compare the children of Q with the children of R. By accounting for points in the same component, this method can quickly and efficiently solve the problem of finding each component's nearest neighbor.

Results This method proved to be the fastest known algorithm for finding EMST's in general metric spaces. I compared it to several well known MST algorithms, including GeoMST2 [2], the previous fastest algorithm. Since this research was motivated by scientific applications, especially cosmology and computational biology, I also tested my algorithm on two large data sets from these areas. One is three-dimensional spectral data taken from the SDSS, and the other is a compressed (12-dimensional) representation of 16,000 steps of folding simulations for 20 proteins. These experiments clearly demonstrated the scalability of my algorithm in terms of running time and storage requirements.

What I learned. Through my research involvements, particularly designing my EMST algorithm, I gained experience with all aspects of algorithm design. I searched and compared the existing methods and explored the scientific literature for applications. I gained experience with powerful algorithmic techniques and the challenges associated with applying them to new problems. I also have worked on communicating my research through a conference submission.

3 Publications

EMST paper (submitted).

References

[1] A. Gray and A. W. Moore. N-body problems in statistical learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[2] G. Narasimhan, J. Zhu, and M. Zachariasen. Experiments with Computing Geometric Minimum Spanning Trees. In *Proceedings of ALENEX'00*, Lecture Notes in Computer Science, pages 183–196. Springer-Verlag, January 2000.