Kinetic Tree and Forest Problems

Nitish Korula - nkorula2
CS 573 - Spring 2006

March 28, 2006

List of Open Problems

1. Finding an efficient (approximate, if necessary) algorithm for maintaining minimum spanning trees of moving points in the plane.
2. Exactly computing the Euclidean Minimum Spanning Tree for moving points.
3. A simple algorithm that maintains exact or approximate MSTs for graphs where edge weights change with time.
4. A small constant-factor approximation for any of the Steiner Tree or Steiner Forest variants (see below).

Consider a set X of n moving points in the plane (or, more generally $\mathbb{R}^d$). If the cost of connecting 2 points is proportional to the distance between them, a natural question might be how to ensure the connectivity of the set X with low cost. For instance, we may wish to maintain the Minimum Spanning Tree (MST) of X. This, and similar problems, have been studied; if the points move linearly (that is, the position of each point is a linear function of time), kinetic data structures due to Basch, et al. [1] can be used to find the MSTs when distances are measured using the L_1 or L_∞ norms. They also show that if the Euclidean distance is used, a $(1+\varepsilon)$-approximate Euclidean MST can be maintained. The authors conclude that their algorithm is "compact, local, responsive, but not efficient."

It would be interesting to find an efficient way to maintain the MSTs, perhaps even at the price of optimality. That is, a simple and efficient algorithm that only computes $(1+\varepsilon)$-approximate MSTs (even with the L_1 or L_∞ metrics) would be significant. Alternately, an algorithm that computed exact Euclidean MSTs would also be of interest.

Agarwal, et al. [2] gave deterministic and randomized algorithms for a slightly different version of the MST problem, where edge weights in a given graph change linearly. However, their algorithms are very complex, and use a large array of tools. They say that they "plan to work on both simplifying [their] methods and improving [the] bounds."

A related problem is that of computing Minimum Steiner Trees of moving points. In the (static) Steiner Tree problem, we have a set X of n points, and a set $T \subseteq X$ of *terminals*. The goal is to find a tree of minimum cost that spans all the terminals, and possibly some of the points in $X \setminus T$. (In general, the added flexibility of using non-terminals (called Steiner vertices) allows us to find a tree that costs less than the MST of T. However, the MST of T gives a 2-approximation to the minimum Steiner Tree, and in the plane, this is improved to $2/\sqrt{3}$.) The Steiner Tree problem is NP-Hard, even in the plane, though Arora [3] gave a randomized polynomial-time approximation scheme (PTAS). (That is, for any $\varepsilon > 0$, the algorithm produces a $(1+\varepsilon)$-approximation in time polynomial in n, though exponential in $\frac{1}{\varepsilon}$.)

The *Steiner Forest* problem is, in a sense, harder still. Here, instead of a single set of terminals, we are given m disjoint sets $T_1, T_2, \ldots T_m$ such that $\bigcup_{i=1}^{m} T_i \subseteq X$. The goal is to compute a forest such that each T_i is contained in a single tree (different sets of terminals can share a tree). A 2-approximation algorithm exists for general graphs ([4] contains a simple description). No PTAS is known for points in the plane if $m \in \omega(\log n)$; if the number of sets is logarithmic or better, Arora's technique for Steiner Trees can be used.

The Steiner Tree problem has a large number of applications and has been well-studied, but there appears to have been no work on maintaining trees when the points move. (This is not very surprising, as work on the easier MST problem has not been particularly successful.) There are several natural kinetic variants; we list a few:

- A (linear) motion pattern is specified for both terminals and non-terminals. The points may be in the plane, or vertices of a graph (in which case, the edge weights change linearly.)
- The terminals move linearly in the plane, but all the non-terminals (potential Steiner vertices) are fixed in specified positions. (In a slightly different version, the non-terminals may be regularly spaced, perhaps on a grid. This could model mobile users and fixed transmission towers, for instance.)
- Moving terminals are given, but any points in the plane can be chosen as Steiner vertices. The set of Steiner vertices can change from one instant to the next, so they do not need to move.
- The motion pattern is specified for each terminal, and the same set of (moving) Steiner vertices must be used throughout the algorithm. However, the algorithm can choose initial positions for the Steiner points anywhere in the plane, and any linear motion plan independently for each such point. (This model is a little unusual, and perhaps is reasonable only if no updates to the motion plan of terminals are allowed.)

Similar kinetic questions can be asked about Steiner Forests. Since these problems cannot be optimally solved even in the static case, we can only expect to find approximate solutions. While a $(1+\varepsilon)$-approximation scheme would be ideal, any (non-trivial) constant would be interesting, particularly in the Steiner Forest case.

Summary of Previous Work

1. To obtain a $(1+\varepsilon)$-approximation to the Euclidean MST of points moving in $\mathbb{R}^d$, the data structure of [1] uses $O(\varepsilon^{-(d-1)/2} n \log^{d-1} n)$ space. The structure is also local and responsive; each point is in few edges and it takes at most $O(\log^d n)$ time to handle a single event. The total number of events the data structure processes is $O(\varepsilon^{-(d-1)} n^3)$; for linear motions, the number of changes to the MST under a polyhedral metric is proportional to $n^{7/3}$ and so this is not efficient. (For any fixed dimension, the number of changes to the Euclidean MST is $O(n^3 2^{\alpha(n)})$, but it is not known if this is tight.)

2. If the edge weights change linearly in a graph, [2] shows how to maintain the MST in time $O(n^{2/3} \log^{4/3} n)$ per event. They also give a randomized algorithm that runs in $O(n^{2/3} \log n)$ time per event.

3. Both [5] and [6] provide a detailed overview of kinetic data structures in general.

References

[1] J. Basch, L. J. Guibas, L. Zhang. Proximity Problems on Moving Points. *Proc. 13th Annual ACM Symposium on Compuational Geometry*, 344-151, 1997.

[2] Pankaj K. Agarwal, D. Eppstein, L. J. Guibas, M. R. Henzinger. Parametric and Kinetic Minimum Spanning Trees. *Proc. 39th Annual ACM IEEE Symposium on Foundations of Computer Science*, 596-605, 1998.

[3] Sanjeev Arora. Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems. *Journal of the ACM* 45(5), 753-782, 1998.

[4] Reuven Bar-Yehuda. One for the Price of Two: a Unified Approach for Approximating Covering Problems. *Algorithmica* 27(2), 131-144, 2000.

[5] Leonidas Guibas. Kinetic data structures. *Chapter 23, Handbook of Data Structures and Applications* (Dinesh P. Mehta and Sartaj Sahni, eds.), Chapman and Hall/CRC, 2005.

[6] Julien Basch. Kinetic Data Structures. Ph.D. Thesis, Stanford University, June 1999.