Lower-Stretch Spanning Trees

Michael Elkin*
Department of Computer Science
Ben-Gurion University of the Negev

Yuval Emek
Department of Computer Science
and Applied Mathematics
Weizmann Institute of Science

Daniel A. Spielman†
Department of Mathematics
Massachusetts Institute of Technology

Shang-Hua Teng‡
Department of Computer Science
Boston University and
Akamai Technologies Inc.

March 4, 2005

Abstract

We show that every weighted connected graph G contains as a subgraph a spanning tree into which the edges of G can be embedded with average stretch $O(\log^2 n \log\log n)$. Moreover, we show that this tree can be constructed in time $O(m \log^2 n)$ in general, and in time $O(m \log n)$ if the input graph is unweighted. The main ingredient in our construction is a novel graph decomposition technique.

Our new algorithm can be immediately used to improve the running time of the recent solver for symmetric diagonally dominant linear systems of Spielman and Teng from

$$m 2^{\left(O(\sqrt{\log n \log\log n})\right)} \quad \text{to} \quad m \log^{O(1)} n,$$

and to $O(n \log^2 n \log\log n)$ when the system is planar.

1 Introduction

Alon, Karp, Peleg, and West (SIAM J. Comput. 1995) proved that every weighted connected graph G contains as a subgraph a spanning tree into which the edges of G can be embedded with average stretch $\exp\left(O(\sqrt{\log n \log\log n})\right)$, and that there exists an n-vertex graph G such that all its spanning trees have average stretch $\Omega(\log n)$.

Boman and Hendrickson (Manuscript, 2001) were the first to realize that low-stretch spanning trees could be used precondition symmetric diagonally dominant matrices. They applied the spanning trees of Alon, *et. al.* to design solvers that run in time $m^{3/2} 2^{O(\sqrt{\log n \log\log n})} \log(1/\epsilon)$. Spielman

*Part of this work was done in Yale University, and was was supported by the DoD University Research Initiative (URI) administered by the Office of Naval Research under Grant N00014-01-1-0795. The work was also partially supported by the Lynn and William Frankel Center for Computer Sciences.

†Partially supported by NSF grant CCR-0324914. Part of this work was done at Yale University.

‡Partially supported by NSF grants CCR-0311430 and ITR CCR-0325630.

and Teng (STOC 2004) improved their results by showing how to augment the low-stretch spanning trees to solve diagonally-dominant linear systems in time

$$m2^{O(\sqrt{\log n \log\log n})} \log(1/\epsilon).$$

By applying the low-stretch spanning trees developed in this paper, we can reduce the time for solving these linear systems to

$$m \log^{O(1)} n \log(1/\epsilon),$$

and to $O(n \log^2 n \log\log n \log(1/\epsilon))$ when the systems are planar. Applying a recent reduction of Boman, Hendrickson and Vavasis, one obtains a $O(n \log^2 n \log\log n \log(1/\epsilon))$ time algorithm for solving the linear systems that arise when applying the finite element method to solve two-dimensional elliptic partial differential equations.

2 Definition

Let $G = (V, E, w)$ be a weighted connected graph, where w is a function from E into the positive reals. We define the length of each edge $e \in E$ to be the reciprocal of its weight:

$$d(e) = 1/w(e).$$

Given a spanning tree T of V, we define the distance in T between a pair of vertices $u, v \in V$, $\text{dist}_T(u, v)$, to be the sum of the lengths of the edges on the unique path in T between u and v. We can then define the stretch of an edge $(u, v) \in E$ to be

$$\text{stretch}_T(u, v) = \frac{\text{dist}_T(u, v)}{d(u, v)},$$

and the average stretch over all edges of E to be

$$\text{ave-stretch}_T(E) = \frac{1}{|E|} \sum_{(u,v)\in E} \text{stretch}_T(u, v).$$

3 Techniques

We build our low-stretch spanning trees by recursively applying a new graph decomposition that we call a *star-decomposition*. A star-decomposition of a graph is a partition of the vertices into sets that are connected into a star: some central set is connected to each of the others. We then add one edge from each of the external sets to the central set. We show how to find star-decompositions that do not cut too many edges and such that the radius of the graph induced by the star decomposition is not much larger than the radius of the original graph.

The central set of the star-decomposition is found using the classic technique of choosing all vertices within some distance of a given central vertex. Each of the external sets is a cone, where the cone induced by a set of vertices S contains all vertices whose shortest path to the central vertex goes through S. We choose the partitions between the cones by generalizing the technique used to choose the central set.