Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving SDD Linear Systems

short abstract

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

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

October 28, 2003

1 Abstract

By improving upon the combinatorial preconditioners of Vaidya, we develop nearly-linear time algorithms for approximately solving sparse symmetric diagonally-dominant linear systems. In particular, for every $\beta > 0$ we present a linear-system solver that, given an n-by-n symmetric diagonally-dominant matrix A with m non-zero entries and an n-vector $\boldsymbol{b}$, produces a vector $\tilde{\boldsymbol{x}}$ within relative distance ϵ of the solution to $A\boldsymbol{x} = \boldsymbol{b}$ in time $O(m^{1+\beta} \log(1/\epsilon) \log(n\kappa_f(A))^{O(1/\beta)})$.

Our algorithm exploits two novel tools. The first is a fast algorithm for approximately computing crude graph partitions. For any graph G having a cut of sparsity ϕ and balance b, this algorithm outputs a cut of sparsity at most $O(\phi^{1/3} \log^{O(1)} n)$ and balance $b(1-\epsilon)$ in time $n((\log n)/\phi)^{O(1)}$.

Using this graph partitioning algorithm, we design fast graph sparsifiers and graph ultra-sparsifiers. On input a weighted graph G with Laplacian matrix L and an $\epsilon > 0$, the graph sparsifier produces a weighted graph $\tilde{G}$ with Laplacian matrix $\tilde{L}$ such that $\tilde{G}$ has $n(\log^{O(1)} n)/\epsilon^2$ edges and such that for all $\boldsymbol{x} \in \mathbb{R}^n$,

$$\boldsymbol{x}^T \tilde{L} \boldsymbol{x} \leq \boldsymbol{x}^T L \boldsymbol{x} \leq (1+\epsilon) \boldsymbol{x}^T \tilde{L} \boldsymbol{x}.$$

The ultra-sparsifier takes as input a parameter t and outputs a graph $\tilde{G}$ with $(n-1) + tn^{o(1)}$ edges such that for all $\boldsymbol{x} \in \mathbb{R}^n$

$$\boldsymbol{x}^T \tilde{L} \boldsymbol{x} \leq \boldsymbol{x}^T L \boldsymbol{x} \leq (n/t)^2 \boldsymbol{x}^T \tilde{L} \boldsymbol{x}.$$

Both algorithms run in time $m \log^{O(1)} m$.

The analysis of these sparsifiers reveals that most of the edges that Vaidya used to augment his maxmimum spanning trees are unnecessary. Our best preconditioners are build by sparsifying augmentations of the spanning trees of Alon, Karp, Peleg and West.

*spielman@math.mit.edu
†steng@cs.bu.edu