ARC Proposal: *Preconditioning in non-Laplacian case*

Ioannis Panageas Advisor: Prasad Tetali, Co-advisor: Frank Dellaert

October 31, 2012

Motivation

Challenges in Robotics motivate the problem of solving a system of the form $A^TAx = A^Tb$, where the matrix A has at most two nonzero arbitrary entries (reals or block matrices) in each row. There is a lot of interest in solving these kind of systems efficiently, which correspond to the errors in "measurements" of the position of the robot while it moves in the 3-D space, so this problem has lots of practical applications.

To do this, a standard approach is to find a preconditioner P, namely a non-singular matrix and solve the system $P^{-1}(A^TAx - A^Tb) = 0$. The iterative methods that are commonly used (such as preconditioned conjugate gradient) have cost that is equal to a single iteration (involving the operation of the matrix and of the preconditioner on a vector) multiplied by the number of iterations. The number of iterations in preconditioned conjugate method is bounded by $c\sqrt{k(A^TA, P)}$ where c is constant and $k(A^TA, P)$ is the condition number of the system and is equal to $\frac{\lambda_{max}(P^{-1/2}A^TAP^{-1/2})}{\lambda_{min}(P^{-1/2}A^TAP^{-1/2})}$. So what we are interested in is to find a preconditioner P such that the condition number $k(A^TA, P)$ is as small as possible.

Prior Work

A lot of work has been done when matrix A is indexed by edges and vertices of a simple weighted graph G and the two nonzero entries in each row and corresponding to an edge e are $+\sqrt{w_e}$ and $-\sqrt{w_e}$, namely A^TA corresponds to a Laplacian matrix L_G. A subgraph preconditioner for L_G is a Laplacian matrix L_H where H is a subgraph of G. Vaidya (1990) was the first (while a lot of work on general preconditioners had been done in 80's) that introduced the subgraph preconditioners but he didn't publish his work. In Boman et al. [5] the authors present a set of techniques for bounding extreme eigenvalues and condition numbers for pencil matrices (A, B) (namely of matrix $B^{-1/2}AB^{-1/2}$) where A, B are symmetric, positive definite. This paper is an introductory paper to the so-called *support theory*. Another paper that goes one step further than the results of Vaidya is [4]. A lot of work towards finding a "good" subgraph preconditioner has been done in [8, 7, 2]. In these papers the notion of κ-approximation is used, which is denoted by

$$L_H \preccurlyeq \kappa \cdot L_G$$

where $A \preccurlyeq B$ means that $B - A$ is positive semidefinite. It is clear that the condition number $k(L_G, L_H)$ is upper bounded by κ. Observe that if H is a subgraph of G with the same weights on the edges then $L_H \leq L_G$. In [8] Spielman and Teng give a randomized algorithm that outputs a low-stretch spanning tree with some additional edges as a preconditioner. The best algorithm known so far that outputs a low-stretch spanning tree can be found in [1]. To define stretch, we consider an embedding matrix W such that $WA_H = A_G$ and A_H, A_G are the incidence matrices of H, G respectively. The stretch of an edge $e = (u, v)$ is defined (see [5, 8]) as follows:

$$st(e) = \frac{\sum_{e':e' \in path(u \to v)} w_{e',H}}{w_{e,G}}$$

(in the nominator we sum over all the weights of the edges in H of the path that (u, v) is mapped through the embedding, and the denominator is the weight of (u, v) in the original graph G). It is straightforward that $st(E(G)) = \sum_{e \in E(G)} st(e) = ||W||_F^2$ and it can be proven that if H is a subgraph of G with the same weights for the edges, it holds that (see [5])

$$k(L_G, L_H) \leq \min_{W: WA_H = A_G} ||W||_2^2 \leq \min_{W: WA_H = A_G} ||W||_F^2 = st(E(G))$$

where stretch in last inequality is defined with respect to the embedding matrix W that minimizes $||W||_F^2$.

In [7] a randomized algorithm is presented that achieves $(1+\epsilon)$ approximation and the number of the edges of subgraph H is $O(n \log n/\epsilon^2)$ and uses notion of resistance (from electric circuits) and improved previous results of [3, 8]. Batson et al [2] give the first deterministic algorithm that outputs a subgraph preconditioner that achieves $\frac{d+1+2\sqrt{d}}{d+1-2\sqrt{d}}$ where $E_H \leq \lceil d(n-1) \rceil$. Both [7, 2] consider the edges of H to have different weights than the edges of G.

If A is symmetric weekly diagonally dominant, then solving $Ax = b$ reduces to another system $\tilde{A}\tilde{x} = \tilde{b}$ which is twice as large and $\tilde{A}$ is symmetric, weekly dominant with non-positive off diagonal entries (Gremban's reduction [6]). Finally, $\tilde{A}$ can be decomposed into Laplacian matrix $\tilde{L}_G$ and diagonal matrix D and the corresponding preconditioner is $\tilde{L}_H + D$ where $\tilde{L}_H$ is a preconditioner for $\tilde{L}_G$ (such a reduction can be found in [8]).

Our goal

As we mentioned at the beginning, we are interested in finding a preconditioner for the system $A^T Ax = A^T b$ with condition number as small as possible and matrix A has at most two nonzero arbitrary reals in each row. We will focus our efforts on subgraph preconditioners. In our setting matrix A can be seen as the "incidence" matrix of a *factor* graph. Clearly a subgraph preconditioner is a matrix B which is the resulting matrix after deleting some rows from A.

There are two obstacles that concern us. First of all observe that a "spanning tree" in our cases cannot necessarily support every row(="edge") that we delete from the initial matrix. To support every deleting edge of G, it suffices to consider a spanning tree plus another row of exactly one nonzero entry (we call that vertex a "*prior*"). The second barrier is the definition of the stretch. Since we are not dealing with Laplacian matrices, it is not clear what will be the weight of every "edge" in our graph. Additionally, if we try to define stretch through the Frobenius norm of an embedding W, where the preconditioner is a spanning tree plus a prior, the position of the prior seems to matter in order to define the stretch for an edge.

In this project, first of all we will try to generalize the notion of stretch to our setting. After doing this, we will focus on finding "good" preconditioner with respect to stretch. From experimental analysis of our collaborators Doru Balcan and Yong-Dian Jian from Robotics Lab , it seems that a random spanning tree plus a prior consist a good preconditioner for our system. So this will be the first approach we will try to analyze in terms of the "generalized" stretch or condition number.

After dealing with the "scalar" case (i.e the entries of A are reals), we want to extend our results to the non-scalar case where the entries are invertible block matrices of constant size: for example, 2-D motion indicates entries which are 3×3 invertible matrices (position and rotation), 3-D motion indicates entries which are 6×6 invertible matrices.

Bibliography

[1] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. *Proc. 44th Ann. ACM Symp. Theory of Computing*, pages 395–406, 2012.

[2] Joshua Batson, Daniel Spielman, and Nikhil Srivastava. Twice-ramanujan sparifiers. *Proc. 41st Ann. ACM Symp. Theory of Computing*, pages 255–262, 2009.

[3] Andras A. Benczur and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. *Proc. 28th Ann. ACM Symp. Theory of Computing*, pages 47–55, 1996.

[4] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *Siam J. Matrix Anal. Appl*, 27(4):930–951, 2006.

[5] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *Siam J. Matrix Anal. Appl.*, 25(3):694–717, 2003.

[6] Keith Gremban. Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems. *PhD thesis*, 1996.

[7] Daniel Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *Proc. 40th Ann. ACM Symp. Theory of Computing*, pages 563–568, 2008.

[8] Daniel Spielman and Shang-Hua Teng. Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 81–90, 2004.