Semesterproject: Approximation algorithms for bandwidth
Carsten Moldenhauer
February 21, 2013

Abstract

Given a graph with \(n\) vertices a linear arrangement is a one-to-one mapping of the vertices of the graph to the set of integers \(\{1, \ldots, n\}\). The bandwidth of a linear arrangement is the maximum difference between the images of the endpoints of any edge of the graph. The bandwidth of the graph is the minimum bandwidth of all possible linear arrangements.

The problem of finding the minimum bandwidth is NP-complete. This project will consider approximation algorithms for this problem.

Previous results

As an example, consider the cycle on \(n\) vertices \(C_n\). It is obvious that the bandwidth of this graph is greater than 1. In fact, it is a fun exercise to find a linear arrangement with bandwidth 2.

There is a number of graph classes for which the optimal bandwidth is known (cf. introduction of [DKENS0]). However, the problem was proved to be NP-hard on general graphs. In particular, it is even hard to find the bandwidth of a caterpillar with hair length 3 (a caterpillar is a tree where all vertices of degree greater equal 3 lie on a path). Hence, there is no hope of finding a polynomial time algorithm that finds the optimal solution for all instances (unless P=NP). Therefore, we consider approximation algorithms.

An \(\alpha\)-approximation algorithm is an algorithm that runs in polynomial time (in the size of the input) and returns a solution of cost \(\text{ALG} \leq \alpha \text{OPT}\), where OPT denotes the cost of the optimal solution. For bandwidth minimization, this means we are looking for an algorithm that runs in polynomial time in the number of vertices and edges of the graph and outputs a solution of bandwidth at most \(\text{ALG} \leq \alpha b^*\), where \(b^*\) is the optimal bandwidth. The factor \(\alpha\) is also called the approximation factor.

There are several results known about approximation. [Fei00] gave an algorithm with approximation factor \(O\left((\log n)^3 \sqrt{\log \log \log n}\right)\). The algorithm is quite simple and the analysis is based on volume respecting embeddings. [BKRV00] gave an algorithm with approximation factor \(O\left(\sqrt{n} \log n\right)\) that is based on a semidefinite-programming relaxation with spreading constraints. The algorithm simply solves the SDP relaxation and projects the values onto a random unit vector which yields the linear arrangement. For general graphs, the best known result is a \(O\left((\log n)^3 \sqrt{\log \log n}\right)\) approximation in [DV01] which essentially combines the two previous approaches to give a slightly better approximation. Better results are known if the graph is restricted to be a tree. [Gup01] gives a beautiful randomized algorithm to compute a \(O\left((\log n)^{2.5}\right)\) approximation.

Concerning lower bounds it is known that it is NP-hard to approximate the bandwidth with a factor better than \(3/2\) on general graphs. This bounds was significantly improved in [DFU11] where it was proved that it is impossible to approximate bandwidth within a factor of \(O(\sqrt{\log n/\log \log n})\) unless there are quasi-polynomial time algorithms for \(NP\).

A lower bound on the bandwidth of a graph is the local density, defined as

\[
D = \max_{v \in V, r} \left\lfloor \frac{N(v, r) - 1}{2r} \right\rfloor,
\]

where \(N(v, r)\) is the set of all vertices that are at distance at most \(r\) from \(v\). There are graphs where this lower bound deviates from the true bandwidth by a factor of \(\Omega(\log n)\) [CS89]. In fact, the algorithm in [Fei00] computes a solution of bandwidth \(O(D(\log n)^3 \sqrt{\log n \log \log n})\). Hence, on the above graphs where local density and minimum bandwidth largely deviate, we can expect the algorithm to perform much better.

Project Goals/Learning Objectives

• Understand the algorithm on trees [Gup01]. This will get you started on randomized algorithms, their analysis and concentration bounds.
• Understand volume respecting embeddings and the algorithm in [Fei00]. A deep understanding is necessary to get an understanding of why/if the algorithm performs better on the particular instances with large discrepancy between local density and minimum bandwidth.

• Implement the algorithm from [Fei00]. Some programming skills are required.

• Understand instances with large difference between bandwidth and local density. This will broaden your knowledge in graph theory.

• Empirical evaluation of the algorithm on those instances.

• Improved analysis of the algorithm in [Fei00] for these instances. Lots of good theorem proving!

• Potentially improved algorithm for minimum bandwidth on trees. Again, lots of theory.

• You are expected to give a midterm and final presentation, as well as prepare a final report.

References


