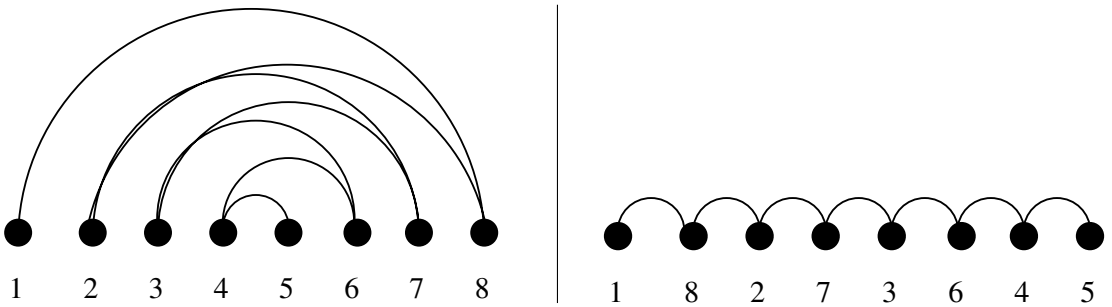

13.2 Bandwidth Reduction

Input description: A graph $G = (V, E)$, representing an $n \times n$ matrix M of zero and non-zero elements.

Problem description: Which permutation p of the vertices minimizes the length of the longest edge when the vertices are ordered on a line—i.e. , minimizes $\max_{(i,j)\in E} |p(i) - p(j)|$?

Discussion: Bandwidth reduction lurks as a hidden but important problem for both graphs and matrices. Applied to matrices, bandwidth reduction permutes the rows and columns of a sparse matrix to minimize the distance b of any non-zero entry from the center diagonal. This is important in solving linear systems, because Gaussian elimination (see Section 13.1 (page 395)) can be performed in $O(nb^2)$ on matrices of bandwidth b. This is a big win over the general $O(n^3)$ algorithm if $b << n$.

Bandwidth minimization on graphs arises in more subtle ways. Arranging n circuit components in a line to minimize the length of the longest wire (and hence time delay) is a bandwidth problem, where each vertex of our graph corresponds to a circuit component and there is an edge for every wire linking two components. Alternatively, consider a hypertext application where we must store large objects (say images) on a magnetic tape. Each image has a set of possible images to visit next (i.e. , the hyperlinks). We seek to place linked images near each other on the tape to minimize the search time. This is exactly the bandwidth problem. More general formulations, such as rectangular circuit layouts and magnetic disks, inherit the same hardness and classes of heuristics from the linear versions.

The *bandwidth* problem seeks a linear ordering of the vertices, which minimizes the length of the longest edge, but there are several variations of the problem. In *linear arrangement*, we seek to minimize the sum of the lengths of the edges. This

has application to circuit layout, where we seek to position the chips to minimize the total wire length. In *profile minimization*, we seek to minimize the sum of one-way distances (i.e., for each vertex v) the length of the longest edge whose other vertex is to the left of of v.

Unfortunately, bandwidth minimization and all these variants is NP-complete. It stays NP-complete even if the input graph is a tree whose maximum vertex degree is 3, which is about as strong a condition as I have seen on any problem. Thus our only options are a brute-force search and heuristics.

Fortunately, ad hoc heuristics have been well studied and production-quality implementations of the best heuristics are available. These are based on performing a breadth-first search from a given vertex v, where v is placed at the leftmost point of the ordering. All of the vertices that are distance 1 from v are placed to its immediate right, followed by all the vertices at distance 2, and so forth until all vertices in G are accounted for. The popular heuristics differ according to how many different start vertices are considered and how equidistant vertices are ordered among themselves. Breaking ties with low-degree vertices over to the left however seems to be a good idea.

Implementations of the most popular heuristics—the Cuthill-McKee and Gibbs-Poole-Stockmeyer algorithms—are discussed in the implementation section. The worst case of the Gibbs-Poole-Stockmeyer algorithm is $O(n^3)$, which would wash out any possible savings in solving linear systems, but its performance in practice is close to linear.

Brute-force search programs can find the exact minimum bandwidth by backtracking through the set of $n!$ possible permutations of vertices. Considerable pruning can be achieved to reduce the search space by starting with a good heuristic bandwidth solution and alternately adding vertices to the left- and rightmost open slots in the partial permutation.

Implementations: Del Corso and Manzini's [CM99] code for exact solutions to bandwidth problems is available at *http://www.mfn.unipmn.it/∼manzini/bandmin*. Caprara and Salazar-González [CSG05] developed improved methods based on integer programming. Their branch-and-bound implementation in C is available at *http://joc.pubs.informs.org/Supplements/Caprara-2/*.

Fortran language implementations of both the Cuthill-McKee algorithm [CGPS76, Gib76, CM69] and the Gibbs-Poole-Stockmeyer algorithm [Lew82, GPS76] are available from Netlib. See Section 19.1.5 (page 659). Empirical evaluations of these and other algorithms on a test suite of 30 matrices are discussed in [Eve79b], showing Gibbs-Poole-Stockmeyer to be the consistent winner.

Petit [Pet03] performed an extensive experimental study on heuristics for the minimum linear arrangement problem. His codes and data are available at *http://www.lsi.upc.edu/∼jpetit/MinLA/Experiments/*.

Notes: Diaz et al. [DPS02] provide an excellent up-to-date survey on algorithms for bandwidth and related graph layout problems. See [CCDG82] for graph-theoretic and algorithmic results on bandwidth up to 1981.

Ad hoc heuristics have been widely studied—a tribute to its importance in numerical computation. Everstine [Eve79b] cites no less than 49 different bandwidth reduction algorithms! Del Corso and Romani [CR01] investigate a new class of spectral heuristics for bandwidth minimization.

The hardness of the bandwidth problem was first established by Papadimitriou [Pap76b], and its hardness on trees of maximum degree 3 in [GGJK78]. There are algorithms that run in polynomial time for fixed bandwidth k [Sax80]. Approximation algorithms offering a polylogarithmic guarantee exist for the general problem [BKRV00].

Related Problems: Solving linear equations (see page 395), topological sorting (see page 481).