Memory-Efficient Fast Shortest Path Estimation in Large Social Networks

Volodymyr Floreskul, Konstantin Tretyakov (kt@ut.ee) and Marlon Dumas
Institute of Computer Science, University of Tartu, Estonia.

Shortest path estimation
Is the President a friend of a friend of yours? Or perhaps a friend of a friend of a friend? How many social links, do you think, separate you from the author of this poster? Have you ever played the Kevin Bacon game? Do you know the Erdős number of Albert Einstein? Games aside, computing distances between nodes of social networks is a rather common problem both for sociological research as well as for various practical applications.

The problem of finding distances in graphs is well-studied. Unfortunately, all exact algorithms for solving it are a bit too slow to be useful in certain. Indeed, the naive shortest path search algorithm will have to traverse nearly the whole graph in order to answer even a single distance query. For a modern billion-user social network, this means performing billions of operations — too much to spend on a single search request.

Landmark-based methods
Computation time may be significantly reduced if we resort to certain approximate algorithms. A popular family of such algorithms are landmark-based methods. The idea is simple: pick several (say, 100) random “landmark nodes” in the graph, then precompute and store shortest path distances from each landmark node to all other nodes.

Now suppose that we want to quickly find the approximate distance between “Joe” and “Mary”. For that we simply check the lengths of all possible paths through the landmark nodes, i.e. [Joe → Landmark1 → Mary], [Joe → Landmark2 → Mary], etc., and choose the smallest one of them. Recall that all the distances [Joe → Landmarkk] and [Landmarkk → Mary] have been precomputed and stored. We’ll only need to retrieve and add them up, so the whole computation will take around a couple of hundred operations — orders of magnitude less than a billion.

Shortest path trees
There are many variations of the landmark-based methods. Importantly, computing a complete shortest path tree (rather than just the distances) makes it possible to find much better distance approximations by locating lowest common ancestors of the query nodes or combining paths from several landmarks.

Pruned landmark trees
The problem with the landmark-based approach is that it may require quite a lot of memory to store the precomputed distances or shortest path trees. For a graph with a billion nodes, storing a shortest path tree (or a set of distances) for a single landmark requires no less than a gigabyte of disk space or RAM. With hundreds of landmarks the space consumption increases proportionally to hundreds of gigabytes.

To overcome this problem we propose to limit the set of nodes, spanned (“covered”) by each landmark so that every node is present in exactly k shortest path trees (where k is considerably smaller than the number of landmarks). As a result, we need to store much smaller trees for each landmark. It turns out such pruned landmark trees can be computed efficiently using a rather straightforward algorithm.

We compared the results to the baseline approach (with full landmark trees) and discovered the following:
• The approximation quality with pruned landmark trees is similar to the baseline approach.
• The query execution time is also similar.
• The amount of memory required to store the landmark data structures is, however, reduced up to 10 times.

Acknowledgments
The authors acknowledge the feedback from Andu Sadan from Skype/Microsoft Labs. This research is funded by DSHF via the Estonian Competence Centre Programme and Microsoft/Skype Labs.