CSE305 Project: Parallel shortest paths

Gleb Pogudin, gleb.pogudin@polytechnique.edu

Finding shortest distances in a graph is one the fundamental problems in computer science with numerous applications (route planning by CityMapper and Google-maps, for example). You have probably learned classical algorithms used for this task in your algorithms course (BFS, Dijkstra, etc). However, these algorithms are inherently sequential and hard to parallelize (although parallel versions exist). In this project, you will be asked to implement and benchmark one of the most standard shortest path algorithms, Δ-stepping algorithm. One can think of Δ-stepping algorithm as a "coarsened" version of the Dijkstra algorithm. While in the latter, at each step we relax edges from a vertex with the smallest tentative distance (which makes the algorithm quite sequential), the former takes at the same time a whole "bucket" of nodes with the tentative distances differing by at most Δ (hence the name). While allowing parallel updates, this approach does not guarantee that the distance is finalized, so the same node may be considered again. Well, there are no free lunches...

The goal of the project is to implement the algorithm and study its comparison on a set of benchmarks. In particular, you have to investigate how the number of threads and other parameters (primarily, Δ) affect the performance. You should also compare your implementation with a serial Dijkstra (it is not so easy to outperform it, in fact!).

Here are some sources on the algorithm:

- The original paper "Δ-stepping: a parallelizable shortest path algorithm", Journal of Algorithms 2003, https://doi.org/10.1016/S0196-6774(03)00076-2.
- Some further development and practical considerations as well as ideas of benchmarks can be found in the paper "Engineering a Parallel Δ-stepping Algorithm", IEEE ICBD 2019, https://ieeexplore.ieee.org/abstract/document/9006237.

For benchmarking, in addition to generating random graphs of different size/sparsity/etc, you may want to look at the SNAP database (https://snap.stanford.edu/data/index.html).