Report BTP

Topic: High probability algorithm for finding transitive closure of a graph under parallel setting

Aniket Jain (201301070)
Sriram Narayanan (201301113)

As a part of this project we implemented the paper by J Ullman and M. Yannakakis to solve the problem of finding transitive closure of a graph.

Transitive closure is defined as follows:

Let G := (V, E) be a directed graph, with n := |V| and e := |E|
Let B be its adjacency matrix ((i, j) ∈ E ⇔ B(i, j) = 1)
Let B* be the transitive closure (TC) of G, i.e., B* (i, j) = 1 if and only if there exists a (directed) path from i to j.
B* is the transitive closure of B.

The problem can be solved using Serial Matrix Multiplication or Floyd-Warshall algorithm. But the above approaches are not efficient if we want to solve single source transitive closure or if the graph is sparse.
Instead of an optimal solution, it is acceptable to get answer with high probability, if we can save up on computation.

We implemented the solutions for the following problems:

1. Single source transitive closure.
    - The algorithm implemented gave an accuracy of about 98%.
    - The results were compared with optimal answers.
    - Implementation was done in C++ and OpenMP for parallelization. On Intel Core i5 Processor.

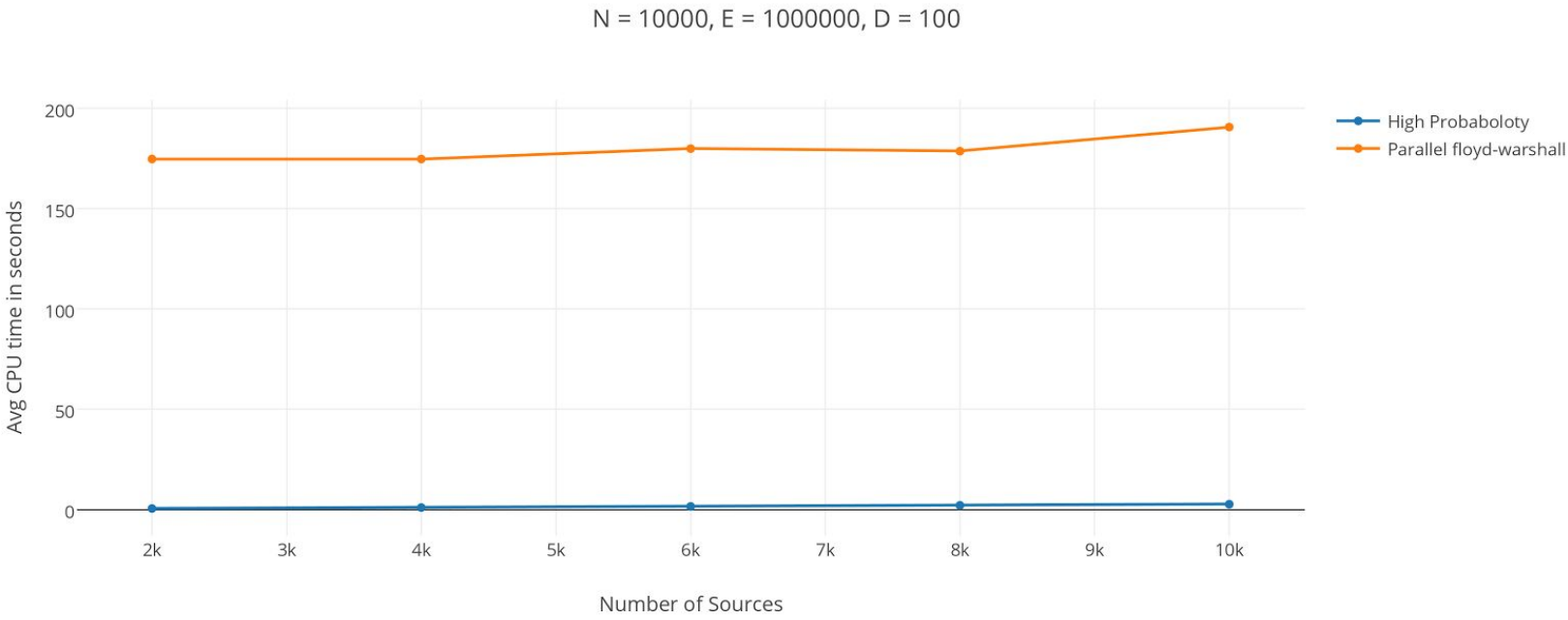
    Observations:
    The algorithm never gives a wrong answer.
    The accuracy of answer depends on the random nodes initially chosen as part of the algorithm.

2. The computation of multiple source transitive closure (with high probability) in sparse graphs.
    S(s, d, n, e) → Problem with n vertices, e edges, s sources and d the maximum path length of transitive closure.
    The problem was broken down into simpler subproblems using 4 recursive rules and 2 base rules. Of which our team implemented 2 recursive rules and the base rules. Using just those rules our algorithm performed as follows:
    - The algorithm implemented gave an accuracy of about 78%.

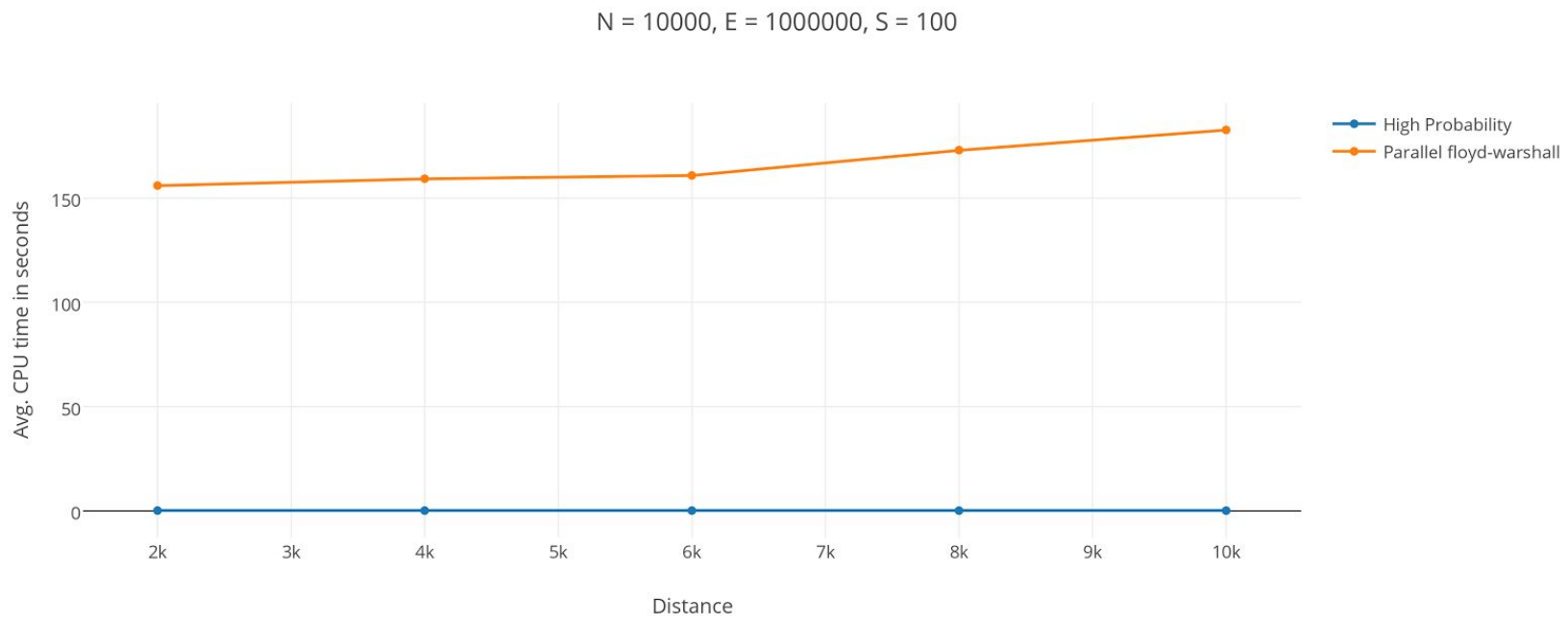
- The results were compared with optimal answers.
- Implementation was done in C++ and OpenMP for parallelization. On Intel Core i5 Processor.
  Observations:
  The algorithm never gives a wrong answer.
  The accuracy of answer depends on the random nodes initially chosen as part of the algorithm.

3. Combined all the rules together to complete the implementation of the paper. Tested the implementation on various graphs and analysed the performance.
   - To test that there are no False Positives, we ran our algorithm on numerous medium sized graphs (N ~ 500-1000) and compared results with parallel floyd-warshall. As stated in our approach there were no False Positives.
   - To measure the time difference between the optimal approach and our approach, we ran both of them on many graphs with random N, E, S and D then compared results.
   - Analysed the accuracy and time difference by varying each one of the parameters of the problem definition namely N, E, S and D.

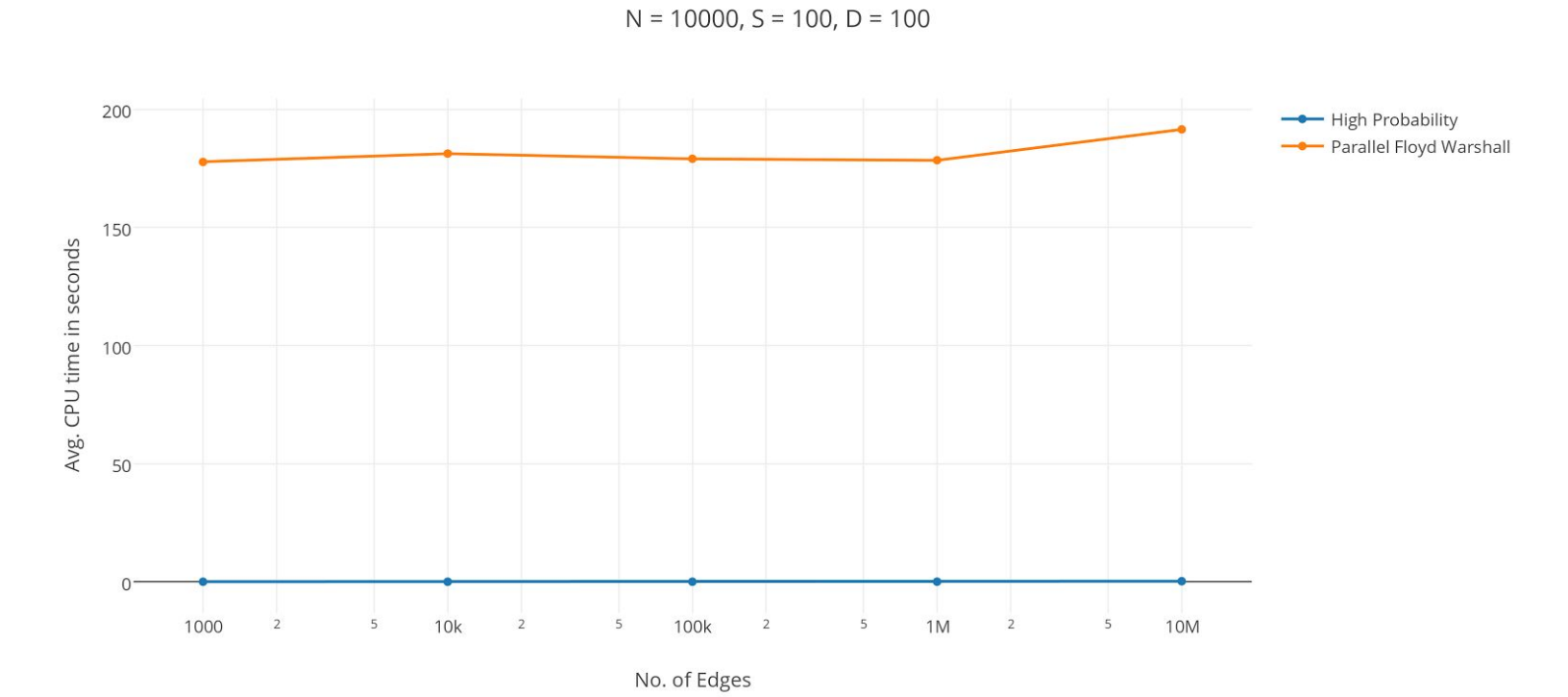
     <u>Keeping N, E and D constant and varying number of sources</u>



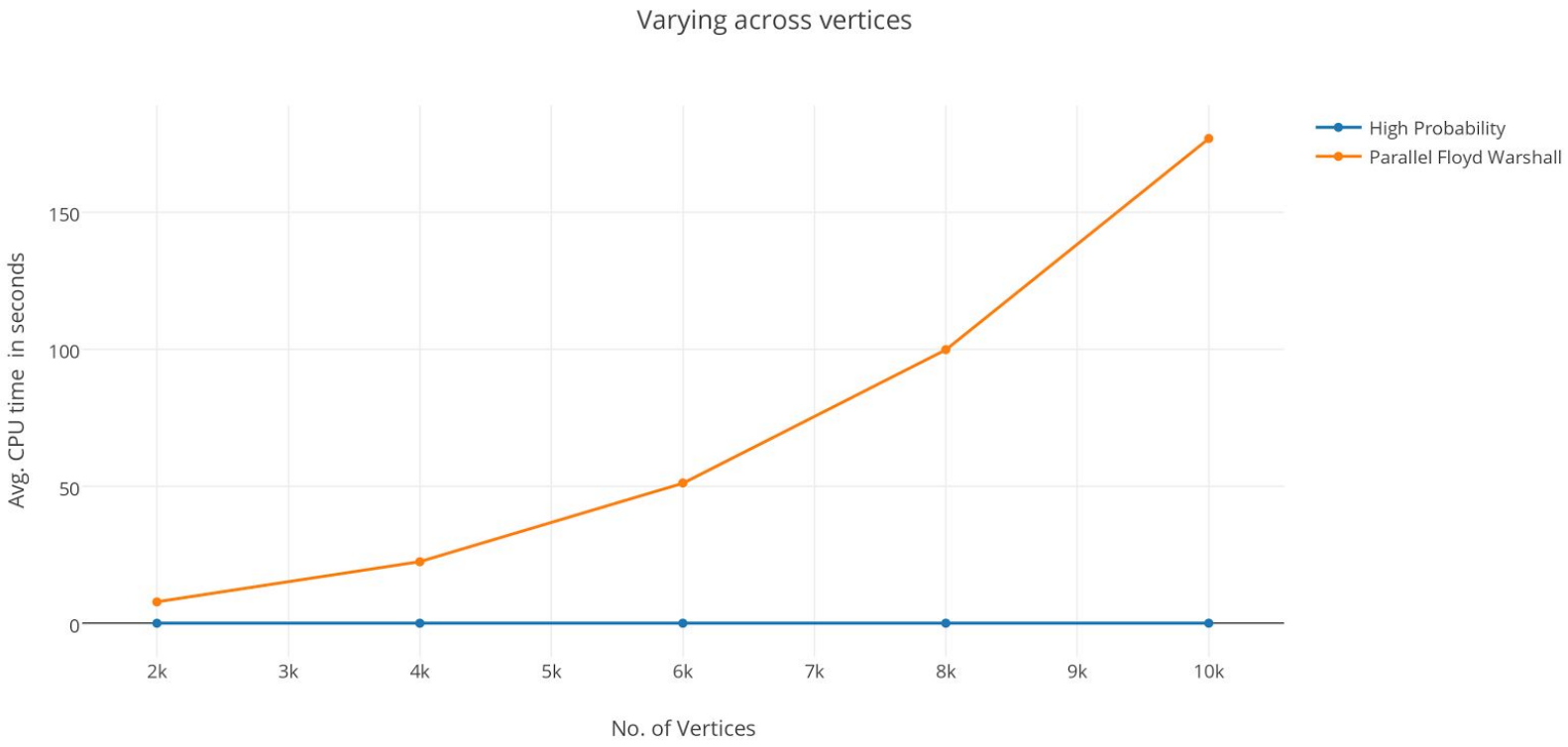
Keeping N, E and S constant and varying the distance limit



Keeping E, S and D constant and varying number of edges



Varying number of vertices; E, S and D vary proportionally to N



- The accuracy of the algorithm was between 95-100%.
- All these results were obtained on the C-STAR machine with 40 cores and 124GB RAM. (The make of the processor is unknown).
- Implemented in C++ using OpenMP.

The source code and the observation results can be found here:
https://github.com/aniketisin/Parallel-Transitive-Closure