CS 470 Project Proposal

Randy Shoemaker, Eliza Shoemaker

February 2019

1 Goal

Our goal is to create a parallel version of the greedy triangulation algorithm and compare it to our serial version. We shall conduct and present a performance analysis. It is our contention that the parallel implementation will provide significant speedup for large data sets of planar points. We do not expect the parallel version to provide any speedup for small data sets, in fact, we conjecture that it will be significantly slower for such data sets due to communication overhead.

The greedy algorithm takes a set of points in the plane and returns a triangulation of the point set. The triangulation is built by adding the smallest line segment between points that does not intersect any line previously in the triangulation. Our serial and parallel implementations of greedy triangulation shall use the "straight forward" approach. This involves "throwing out" all lines that intersect a newly added line of the triangulation and then adding the smallest line left; this process is repeated until the triangulation is complete. This has a worst case complexity of $O(n^3)$. Dickerson et. al. developed an $O(n \log n)$ algorithm, but it relies on the input point set being uniformly distributed in a convex region [1]. Levcopoulos and Krznaric developed an $o(n)$ algorithm to build the greedy triangulation given the Delauny triangulation [3]. The advantage of the "straight forward" algorithm for building the greedy triangulation is that it does not rely on already having the Delauny triangulation, nor does it rely on the features of the input set. This approach is also less complex than other approaches [1]. There has been previous work in parallelizing the greedy triangulation algorithm. Jansson developed a parallel algorithm that runs in linear time but requires $O(n^4)$ processors [2]. In this implementation each pair of line segments is assigned to a single processor as a result, this approach only works for small data sets. It is our contention that our approach will be practical for large data sets.

2 Relevance

Our project will use a distributed memory approach. We may augment this with each node running multiple threads if this aides in the performance.

3 Methods

We are currently developing a serial implementation that computes the greedy triangulation. We plan to analyze its performance for various input sizes. We shall develop the distributed memory approach of the parallel version using MPI. If we can get an improvement in performance we shall use OpenMP to carry out multi-threading on each process. Our serial version operates in three phases. In the first phase we construct line segments between every pair of points and compute their distances. In the second phase we sort the lines in non-decreasing order. In the third phase we build the triangulation according to the "straight forward" approach mentioned above.

4 Possible Roadblocks

We anticipate that implementing phase three of the algorithm will be the most difficult portion of the parallelization process. We believe this because each time a new line is selected to be in the triangulation we must eliminate all lines that intersect with it. To overcome this we are going to split up the lines between all processes and use MPI's *Allreduce* to select and broadcast the shortest line.

5 Mid-Project Deliverable

We shall produce a working parallel version using only MPI for our mid-project deliverable. We will also provide a means of producing an image from our program's output so that its correctness can be visually verified. At this point we do not expect to have any performance analysis completed.

6 Showcase

To showcase our work we shall use a poster that displays our performance analysis and some sample outputs of our software.

7 Final Deliverable

Our final deliverable shall include the serial and parallel software along with software to view the output. We shall also include software that produces random inputs for our program so that who ever is interested may use our software with minimal effort. The final product may have multi-threading within processes if it aides in performance.

References

[1] Matthew T. Dickerson, Robert L. Scot Drysdale, Scott A. McElfresh, Emo Welzlc. *Fast greedy triangulation algorithms.* Computational Geometry, 8:67-86, 1997.

[2] Jesper Jansson. *Planar Minimum-Weight Triangulations.* Lund University, Sweden. 1995 Retrieved from https://www.semanticscholar.org

[3] Christos Levcopoulos, Drago Krznaric. *The greedy triangulation can be computed from the Delaunay triangulation in linear time.* Computational Geometry, 14:197–220, 1999.