Parallel Functional Programming Proposal

George Morgulis, Henry Lin

November 2024

Convex Hull

Consider a set of points on a plane. The convex hull for this set is the smallest convex polygon that completely encloses all the points, ensuring that no corners curve inward.

Approaches

I will attempt to parallelize several known algorithms and compare the results. Below, I list a few algorithms that I would like to parallelize, in order of preference. However, due to time constraints, I may not finish them all.

Quickhull

Expected: $O(nlogn)$
Worst: $O(n^2)$

1 Find the points with the minimum and maximum x coordinates.

2 Divide the set of points into two subsets: those above and below the line. Process them recursively.

——————For each——————

3 Determine the point with the maximum distance from the line.

4 Form the respective triangle, ignore all points within the triangle.

5 Recursively repeat the previous two steps until no points are left.

The selected points consitute the convex hull.

Quickhull Parallel

1. Parallelizable for large sets through divide and conquer
2. Process both subsets in parallel
3. Again, divide and conquer for large sets

Jarvis March (Gift-Wrapping Algorithm)

1. Start with the point with the smallest Y coordinate
2. Select the point with the smallest counter-clockwise angle, compared to previous vertex. This is done looping through all points in hull and computing the cross product.
3. Return when starting vertex is reached

Jarvis March Parallel

1. For each search, divide the total points evenly among the threads. Let each thread perform a brute force search and then compare the results.

Chan's Algorithm

Finally, there is Chan's algorithm, which uses Jarvis' algorithm and Graham's algorithm as subroutines. Graham's algorithm was parallelized by a student in this class in 2022 (Andrei Coman). If time permits (and if I am permitted to do this), I would like to merge our implementations of Graham's and Jarvis' algorithms to see if I can speed up Chan's algorithm. However, I might not have enough time for this.

Testing

I will generate large random data sets to test my algorithm.

Additional Features

I hope to use the Haskell binding for Matplotlib to optionally produce a rendering of the convex hull.

Citations

https://en.wikipedia.org/wiki/Convex_hull_algorithms

https://en.wikipedia.org/wiki/Quickhull

https://www.cs.umd.edu/class/spring2020/cmsc754/Lects/lect03-hulls-bounds.pdf

https://www.cs.columbia.edu/~sedwards/classes/2022/4995-fall/proposals/ConvexHull.pdf

https://www.youtube.com/watch?v=B2AJoQSZf4M