

Final Project Guidelines

Parallel Algorithms (Fall'07)

Your final project is an opportunity to explore the use of parallel algorithms on real-world datasets. Below are some project ideas to help you start thinking. You are more than welcome to suggest your own project; there is really no limit in creativity on your part. Each project is to be completed by 1 - 2 students. The final project is worth 20% of your grade.

Graph Theme. Graphs are an important abstraction in computer science. Many important problems in optimization, machine learning, and various other fields can be formulated succinctly as graph problems. For the project, you can consider implementing one of the graph algorithms and running it on real-world graphs. You can select from a number of graph algorithms we discussed in class, or be more adventurous and read recent papers.

You will have access to (at least) the following datasets: partial Google Web Graph, a number of road networks, a power-grid graph, a number of Mesh graphs, and the Netflix graph. Here is a brief list of what fun applications you can try on these graphs:

- **Spanning Tree.** Often, we would like to know whether or not a graph is connected. For many real-world applications, this is the first step before running other sophisticated algorithms. Constructing a spanning tree automatically gives us connectivity and some more.
- **Shortest Paths.** Your favorite trip planning program, e.g. Google Map, MapQuest, needs this (and a few more tricks).
- **Maximal Independent Set.** A great number of parallel algorithms rely on being able to compute a maximal independent set efficiently. There are various tradeoffs you can consider here.
- **Graph Separator.** In the simplest form, the graph separator problem is seeking a partition of a graph into 2 (roughly) equal parts that minimize the number of edges going across the partition. This subject has fascinated people from medical imaging and computer vision to theoretical computer science. You can be thrilled, too.
- **Biconnected Components.** When the route from A and B fails, we seek an alternate route. A biconnected component is a graph where there are always 2 disjoint paths between every pair of nodes. Can you find all of them?
- **Graph Drawing.** I have 1,342,269 nodes and about ten million edges. How could I possibly visualize this graph? We want to render the graph nicely on a screen so that what we see makes sense.