Parallel Algorithms:
The Minimum Spanning Tree
And
Minimum Steiner Tree Problems

Katie Zrncic

COMP 512
Spring 2005

Introduction

"Parallel computing is one of the most exciting technologies to achieve prominence since the invention of electronic computers" [6]. The goal of this project is to explore the field of parallel computing by implementing parallel algorithms. The first algorithm will solve a common problem – finding a minimum spanning tree for a graph. The second will be an approximation algorithm for a harder problem – finding a minimum Steiner tree.

Problem

The Minimum Spanning Tree problem (MST) is defined as follows in [2].

Input: A connected, undirected graph $G = (V, E)$ with a weight function $w : E \to \mathbf{R}$.

Output: An acyclic subset $T \subseteq E$ that connects all of the vertices and whose total weight is minimized.

Minimum spanning trees have many common applications such as designing electronic circuits, computer and telephone networks, and airline routes. Any problem that involves connecting points with a minimum cost can be viewed as an instance of MST.

The most common methods for solving the MST problem are to use Kruskal's or Primm's algorithms, which are sequential. Although not mentioned as often, Boruvka's algorithm was actually developed earlier and has a structure that can be readily adapted for parallel processing [4]. For this project, I plan to implement a parallel algorithm based on Boruvka's work. As part of this implementation, I will need to focus on how to convert a sequential algorithm into a parallel one, what data structures to use for the most efficient communication among processes, and how to partition the data and work among the available processors.

I am not the first person to create a parallel algorithm for finding minimum spanning trees. However, I believe this is still a valuable piece of my project because it will allow me to explore the area of parallel algorithms and distributed computation.

The second piece of my project will be to implement a parallel algorithm for the Minimum Steiner Tree problem. This differs from MST in that additional points may be added to the solution in order to reduce the total edge weight. Therefore the result is a minimum spanning tree covering all of the vertices in the graph plus the added Steiner points. The Minimum Steiner Tree problem is NP-hard, and so approximation algorithms are often used [9]. One approximation method is to start with a minimum spanning tree and then to insert Steiner points as needed. This shows how the two problems are related and may be a good starting place for my Minimum Steiner Tree implementation.

Strategic Plan

This section of the proposal describes more details about my planned implementation.

I will use the C MPI library for my implementation. MPI is a standard, well-documented parallel message passing library and I am already familiar with programming in C. Therefore I feel that this will be a good environment for me to work in. MPI handles many distributed issues such as communication protocols (blocking/non-blocking, buffering, etc.), which will allow me to concentrate on the interaction between processes, with data and results being sent back and forth.

Tactical Plan

This project will be implemented using a three-stage plan of attack. This section outlines those stages and gives anticipated timeframes for each stage.

1. Design (2 week): My first step will be to examine the design areas for parallel algorithms, make design decisions, and document why I made those choices. This will result in a detailed plan for the actual implementation.

2. Program and Test (4 weeks): Once a detailed plan is in place, I will work iteratively rather than trying to build the end-product from the start. My milestones will include:

a. Create a simple parallel algorithm to demonstrate basic communication between the processes.
b. Add functionality for a user to input a graph and pass some information among processors.
c. Complete the parallel MST algorithm. (Worst Case Result)
d. Implement a Minimum Steiner Tree approximation algorithm with some parallel steps (e.g. finding a minimum spanning tree to start building a Steiner Tree from). (Expected Result)
e. Complete a parallel Minimum Steiner Tree algorithm. (Best Case Result)

3. Write Report (2 weeks): Completing the written report will include documenting my final design decisions, describing problems or surprises that were encountered during the project and how I handled them, and documenting any outstanding issues and how I would approach those issues given more time.

References

1. Chandy, K. Mani and Taylor, Stephen. “An Introduction to Parallel Programming”, Jones and Bartlett Publishers, 1992.
2. Cormen, Thomas H., Leiserson, Charles E., and Rivest, Ronald L. “Introduction to Algorithms”, The MIT Press, 1990.
3. Foster, Ian T. “Designing and Building Parallel Programs”, Addison-Wesley Publishing Company, 1995.
4. Graham, R. L. and Hell, Pavol. “On the History of the Minimum Spanning Tree Problem.” Annals of the History of Computing. Volume 7:1, pp. 43 – 57. January, 1985.
5. Gropp, William, Lusk, Ewing, and Skjellum, Anthony. “Using MPI: portable parallel programming with the message-passing interface”, The MIT Press, 1999.
6. Lester, Bruce P. “The Art of Parallel Programming”, Prentice Hall, 1993.
7. Lewis, Ted G. and El-Rewini, Hesham. “Introduction to Parallel Computing”, Prentice Hall, 1992.

8. Quinn, Michael J. “Parallel Programming in C with MPI and OpenMP”, McGraw-Hill, 2004.
9. Weisstein, Eric W. "Steiner Tree." From *MathWorld* -- A Wolfram Web Resource. http://mathworld.wolfram.com/SteinerTree.html.