Parallel Graph Isomorphisms  
Research Proposal  
Rohan Yadav

Introduction

Graph isomorphism is the problem of deciding whether two input graphs $G_1$ and $G_2$ have the same structure, but with just different labels. Intuitively, two graphs are isomorphic if there is a mapping between the vertex labels of the two graphs that preserves the structure of the graph. Subgraph isomorphism is a similar problem, but where $G_1$ is smaller than $G_2$, and we want to know if $G_1$ exists inside the structure of $G_2$. A concrete example of this is to let $G_1$ be a simple square graph. If we want to know if there is a subgraph isomorphism from $G_1$ to $G_2$, that means we want to know if this small square graph is a subgraph of $G_2$.

While this problem sounds simple on small instances, for large problem instances, it is computationally intractable. Subgraph isomorphism is provably NP-Complete, meaning a polynomial time algorithm for the problem unlikely, and graph isomorphism is unknown to be NP-Complete or not, but all of the practical algorithms for the problem are exponential time in the worst case as well. The implications of these classifications is that for most large graphs that are of interest, it is not possible to run isomorphism and subgraph isomorphism algorithms on these interesting graphs because the algorithm just can not complete.

However, there are many real world applications of graph and subgraph isomorphisms, and having fast practical implementations of tools for these problems is very useful to various groups in fields other that computer science as well. Graph isomorphisms are extremely useful in the field of biochemistry, where researchers need to identify occurrences of a particular protein structure, which can be modeled as either the subgraph or graph isomorphism problems. In fact, biochemistry uses graph and subgraph isomorphisms so heavily that much of the algorithm development for these problems is in response to problems in biochemistry. In addition to biochemistry, subgraph isomorphism appears in fields like computer vision and pattern recognition, as finding a target pattern or object in a picture is akin to finding a subgraph inside a larger graph. Another example of the use of isomorphism is in circuit design, to verify whether the connections encoded by the integrated circuit are the same connections in the circuit schematic.

There are a large number of existing tools and algorithms for graph and subgraph isomorphism, including VF2, Nauty, Traces, and Bliss to name a few. However these algorithms are all purely sequential in nature. As the sizes of the input graphs get large, these algorithms are unable to provide an answer within a reasonable time, which limits the graphs that these groups above can use for recognition or verification. As processor clock rates have already begun to stagnate, we need to scale solutions for isomorphism horizontally to multiple cores, and create a parallel algorithm for graph and subgraph isomorphism. Groups who need access to fast isomorphism should be able to utilize the high core count machines they have access to, and be able to process larger and larger graphs faster than with only a single processor. My research proposal is to develop a fast, performant, and practical parallel algorithm for graph and subgraph isomorphism.
Project Goals

75% Goals
As a 75% goal, I hope to accomplish at least a functioning parallel algorithm and implementation, with acceptable multi core performance, with perhaps poor single core performance. In essence, this goal is to accomplish at least a working parallel algorithm for isomorphism.

100% Goals
As a 100% goal, I hope to present a strong and performant algorithm for parallel isomorphism. My goal is to have equal single core performance as other existing algorithms and tools, and have strong speedups on different types of graphs, especially on isomorphism challenge graphs. Additionally, I hope to see strong results on graphs that are used in real applications like biochemistry and computer vision.

125% Goals
As a 125% goal, I hope to accomplish everything in the 100% goal category, as well as being able to extend my algorithm to scheduling other similar NP-Complete and computationally intractable problems. Since a majority of NP-Complete have a similar branching structure to their search space, perhaps an algorithm for parallel graph isomorphism can be extended to these computationally hard problems as well.

Evaluation Metrics
There are a large number of benchmark sets for graph isomorphism tools, as well as challenge graphs. Additionally, there are papers written comparing different tools on specific benchmark sets, so I plan on benchmarking my algorithm on these benchmark sets, and measuring the speedup on different core counts on these sets, as well as the times on one core. My goal is to have an algorithm that is only slightly slower than the serial algorithms on 1 core, and achieves good speedups on higher core counts.

Milestones

1st Technical Milestone
Before 15-400 starts, I hope to have a solid understanding of various graph and subgraph isomorphism algorithms. Additionally, I want to have a deeper understanding of different scheduling systems so that I can have a large toolkit of techniques to apply and modify to parallelize existing algorithms.

Biweekly Milestones
1. January 31 - Have ideas for parallel algorithm / scheduling technique
2. February 14 - Formalize parallel algorithm
3. February 28 - Begin implementation
4. March 21 - Finish implementation, begin test suite construction
5. April 4 - Run large tests on benchmarks, construct comparisons to existing algorithms
6. April 18 - Finish compiling all results together, begin write up and poster
7. May 2 - Finish write up and poster, and be presentation ready

Faculty
I will be advised by Professor Umut Acar on this project.

Literature
I have been collecting multiple papers on different graph isomorphism algorithms, including papers the papers on the VF2 Algorithm and Ullman Algorithm, as well as various papers on scheduling that Professor Umut Acar has given me to read.

Resources
The major resources I will need for this project are high core count machines to run tests on, which I have access to through Professors Umut Acar and Guy Blelloch.

Website Link
The website link for my project is here.