Nelson Batista, Max Inciong, and Francesca Truncale

Senior Project II — Fall 2017

Professor Jianting Zhang

Report for Oct. 23

The past four weeks, we have successfully completed and profiled a CPU implementation of our project using the Ullmann algorithm. We also completed a terminal-based interface which takes two filenames as arguments, the space-separated adjacency lists of the graph and subgraph. Alternatively, these filenames may be omitted if the `--interactive` flag is specified. In this case, the user is prompted to enter the adjacency lists manually from the terminal. This is useful for entering small graphs for testing purposes. There is also a `--debug` flag which generates additional output for debugging purposes. One of our members is currently in the process of testing the CPU implementation a bit more thoroughly than normal to make sure there are no bugs which may be affecting performance, but so far it seems to run correctly.

Speaking of performance, we ran the CProfile functions in Python to test the performance of our CPU implementation, and found that it took about 15 seconds to run on our Facebook graph with a small (about 5 edges) subgraph. We have since identified various bottlenecks in the code that would benefit the most from parallelization. This includes an O(n) function nested within several layers of for loops which runs many, many times every time we must identify an isomorphism. This is compounded by the fact that we must run the isomorphism function repeatedly to count the number of matches of the subgraph in the larger graph.

The main sticking point currently is a few errors in our `isomorphism.cpp` file. Namely, we're having a number of issues with matching types. We've done all of our business using the C++ standard template library, and have on several occasions changed our representations for various graph features, leading to a bit of a mess since we're passing things around by pointers and not all container classes in the STL have the same access methods. We've also not had very much time to work on fixing these errors due to deadlines in other classes along with midterms. Overall, it's been a very stressful few weeks. We've begun work on looking into various resources to help with identifying a good approach to take with parallelizing the above-mentioned bottlnecks. One of our team members seems to have a promising idea, and is currently in the process of fleshing it out into something more concrete. This may involve a slight change in the algorithm itself, rather than a simple parallelization, but we believe this may be the appropriate course of action regardless.