Parallel Functional Programming

Project Proposal

TEAM

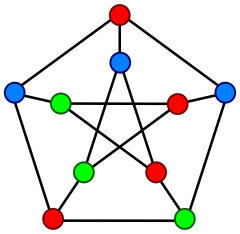
Ishan Guru	ig2333
Haneen Abdulrashid Mohammed	ham2156

OVERVIEW

The problem we are aiming to solve is the graph colouring problem:

Given a graph with n vertices, find a way to assign a colour, given x colours, to each of the vertices such that no two adjacent vertices are of the same colour.

For example, the graph below is a valid, 3-colour solution to our problem:



Our aim is to provide both a single threaded solution and multi-threaded solution to this problem.

In the single threaded case, the graph and number of colours would be input parameters, while our parallelized solution would also take the number of threads to use to solve the problem (with a predefined limit that we find through testing). The output of our both solutions would be one, if any, valid colouring of the graph. Our hypothesis is that as the number of threads increase, to a certain point, the time taken to solve the problem will decrease.

Although we are still determining the exact details of our parallelization, our current thoughts are to parallelize in multiple different ways:

- Distribute the graph into smaller subsets (based on number of threads) and solve each sub-graph in parallel to come up with a solution (will involve merging each subgraph in the final solution)
- Concurrently solve for different colourings and end execution when any thread finds a valid solution

- During exploration of the graph, hand off each possible sub-graph traversal to a new thread to assign colours until each node has been visited.

Through these algorithms, we'd be able to effectively test how performance increases or decreases based on the number of threads used to run the program.

Our goal is to also develop a testing suite for our solution, to easily and effectively be able to determine the impact of how tuning different parts of our algorithm (in the parallel case, thread count or algorithm choice) would increase or decrease performance. We're hoping to be able to find out what the optimal choices (both the algorithm and the number of threads) are to solve this problem.