PGraphColor: Parallel Genetic Algorithms for Graph Coloring

Milen Ferev, mf3231

November 2021

Introduction

The Graph Coloring Problem is an NP-Complete problem which requires the vertices of a graph to be colored such that no two adjacent vertices have the same color. More formally, given a graph $G = \{V, E\}$ and a function $\delta : V \to C$ that maps nodes to a set of colors C, it is required that $|C|$ is minimal and that

$$\forall (u, v) \in E : \delta(u) \neq \delta(v)$$

In the literature on the problem, different techniques have been applied for finding a valid graph coloring for a given graph. Such techniques include Ant Colony Algorithms, Simulated Annealing, Genetic Algorithms, Backtrack Search, and others. For this project my focus will be on developing a Parallel Genetic Algorithm for the Graph Coloring Problem.
Genetic Algorithms are inspired by evolutionary processes. Over its run-time the algorithm maintains a population of individuals, each of which is a potential solution. At each iteration a new population is generated using the current one. The individuals are altered via three operators: mutation, crossover, and selection. The mutation operator causes a random mutation in an individual. The crossover operator uses individuals from the current population to create a new individual. The selection operator is used to select individuals for crossover operator to be used. The quality of each solution is measured by a fitness function.

Problem Formulation

For this project I will be using a common formulation of the Graph Coloring problem used in many of the techniques mentioned above. The formulation consists of fixing the number of colors k and running the search algorithm in order to find a valid coloring. This then allows for a search to be done over the value of k.
In my project I will be using a Genetic Algorithm (GA) as a routine, where $GA(G, k)$ returns a coloring of the graph G with k colors, if any valid coloring

was found. Then I will be running a binary search over k, returning the coloring with the minimum number of colors found.
I plan on representing each potential coloring as a list of size $|V|$ where each color will be represented by an integer. The fitness function will be given by the number of adjacent vertices of the same color. The mutation operator will modify the color of a randomly selected node in the solution. The crossover operator will take two lists and return a new one where all elements up to some index i are taken from the first list and after i taken from the other.
For testing the correctness and performance of my implementation I will be using the tests given by The Center for Discrete Mathematics and Theoretical Computer Science, which can be found here.

Parallelism

The overall structure of the Genetic Algorithm and the operators allow for the routine to be redesigned for parallel execution in multiple ways. As of now I have three ideas for achieving parallelism of the algorithm. The first would be to run multiple instances of the genetic algorithm with different populations and possibly different hyper-parameters and then collect the results. Alternatively, one instance of the algorithm can be executed, but the operators can be applied in parallel to generate the next population. Lastly, the two ideas can be applied at the same time. As of now I am in favor of the third idea but will design my implementation using the first. In case I have time left I will move onto implementing the third idea.

Project Deliverables

- An implementation of a parallel genetic algorithm for the graph coloring problem
- Benchmark of performance and correctness recorded from running the algorithm with different sets of hyper-parameters and parallel architectures
- Benchmark alongside a single-threaded solution