15618 Project Proposal

Haoxuan Zhu (haoxuanz), Ziyi Liu (ziyiliu)

TITLE

Parallelizing Genetic Algorithm for Traveling Salesman Problem

URL

https://hxzhu9819.github.io/pGA/

SUMMARY

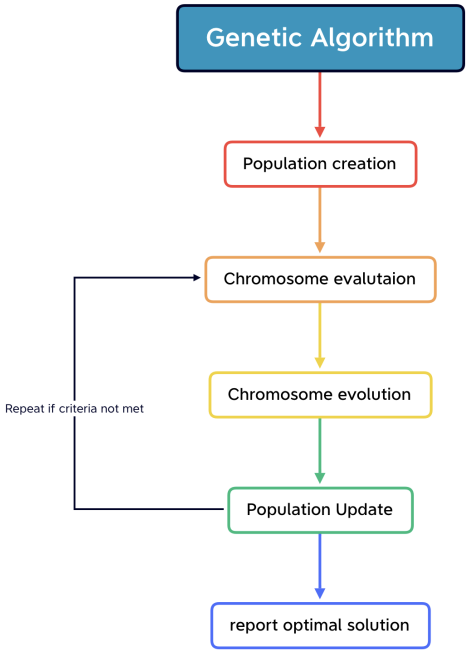
We are going to implement and benchmark different parallelizing strategies, including openMP and CUDA, to accelerate the genetic algorithm for solving the traveling salesman problem.

BACKGROUND

Genetic Algorithm (GA) is a search-based algorithm that is widely used to solve NP-complete problems. Inspired by genetics principles, the algorithm converges to solutions by reflecting the process of natural selection where the fittest chromosomes are chosen for reproduction to produce better offspring of the next generation until an optimized solution is generated.

Genetic Algorithm is an iterative process. After randomly initializing a population, the algorithm executes the following 4 steps recursively, scoring and selecting elite parents from the population for mating, applying crossover to generate next generation chromosomes, conducting mutation to introduce data augmentation, and updating existing population by removing low-score chromosomes, until a predefined criteria is met.

With a large number of population and iterations, the computation required for fitness score evaluation, crossovers, and mutation will increase, resulting in severe performance reduction. Different parallelization strategies can be applied to different phases based on their computation and data dependency nature and may ideally accelerate the process. Data parallelism can be performed during chromosome evaluation by exploiting independencies of chromosomes, and ad-hoc parallelism strategy for chromosome evolution (crossover and mutation phases) can be designed to accelerate the process. The huge population and computation requirements allows parallel programming to scale well.



CHALLENGES

The genetic algorithm is implemented in a multi-phase fashion where each phase has different properties that need to be parallelized differently. For example, chromosome evaluation scores each chromosome individually, which can be easily parallelized, but the following step chromosome evolution requires scores of all chromosomes, which automatically serves as a barrier. Worse still, the evolution process has many sequentially dependent operations that limit parallelization. Hence, we are interested in figuring out how to parallelize the workload efficiently.

After initial analysis, we think the challenges of the project can be concluded as

1. Dependency exists among the stages in the genetic algorithm in each iteration. Therefore, finding the optimal parallelization strategy while preserving the dependencies will be hard. Moreover, because of dependency issues, workload imbalance can lead to high synchronization costs.
2. Data movement can hurt performance if the intermediate results are not stored properly. So careful consideration must be given to where and how to store the data.
3. Being able to develop scalable parallel algorithms as the size of the population and number of iterations increase significantly.

RESOURCES

We will majorly use GHC machines which have OpenMP and CUDA support for development and finally test the OpenMP solution on PSC machines with more cores.

In terms of the codebase, we will start from scratch writing a sequential version and then parallelize it.

Our project is inspired by the following paper

- https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-020-0157-4

GOAL AND DELIVERABLES

PLAN TO ACHIEVE

We hope to complete the sequential version of the genetic algorithm specifically for solving the traveling salesman problem and parallelize it using OpenMP and CUDA. Since the genetic algorithm is a computation-heavy algorithm, we expect to see a significant speedup after parallelization. Next, we plan to compare the performance of speedup of implemented approaches and analyze the reason for the outcome. Eventually, we will scale the population up, benchmark and evaluate our OpenMP version using PSC machines.

HOPE TO ACHIEVE

Since we plan to implement different approaches to the same problem, we hope to have a thorough understanding of the genetic algorithm and its potential bottlenecks, and eventually apply suitable parallel strategies to boost performance. One goal would be to identify the bottlenecks using performance tools, and show how different approaches have different effects on the bottlenecks.

If all goes well, we plan to report our comparative analysis results through graphs that illustrate the speedup of different approaches, and have a demo to show how our code works in action. The showcased result will likely be broken down into times taken by each phase of the algorithm and comparison across different implementations.

PLATFORM CHOICE

C++ will be used as our programming language because of its support for OpenMP and CUDA. Since the population of the genetic algorithm can be really large, it is a good use case for GPUs. However, chromosomes also have many sequentially dependent operations, we are interested to see which one yields a better result and if a heterogeneous approach is feasible to boost performance.

As is mentioned in the previous section, GHC machines with GPU and OpenMP installed will be a good choice for development.

SCHEDULE

Time	Plan
Week 1 11/07 - 11/13	Finish project proposal and start sequential implementation
Week 2 11/14 - 11/20	Finish sequential implementation and start parallelizing using CUDA on GHC machines
Week 3 11/21 - 11/27	Finish CUDA implementation and project milestone report
Week 4 **11/28 - 12/04**	(milestone report checkpoint) Finish OpenMP implementation and profiling on GHC machines with small number of cores
Week 5 12/05 - 12/11	Run OpenMP implementation on PSC machines with up to 128 cores to compare performance
Week 6 **12/12 - 12/18**	(final report due) Report writeup and demo prep