UNIVERSITY OF PISA,
PARALLEL AND DISTRIBUTED SYSTEMS: PARADIGMS AND MODELS

Parallel genetic algorithm for traveling salesman problem

Andrea Rosasco

Computer Science (Artificial Intelligence)

July 14, 2020

Contents

1 Introduction

The aim of the project is to figure out how to efficiently parallelize a Genetic Algorithm (GA) applied to the Traveling Salesman Problem (TSP) and to implement the solution both with the FastFlow framework and the standard c++ parallel mechanisms. The performance of the two implementations (e.g. speedup, scalability) are then compared and discussed.

Before starting discussing how to solve the problem let's lay down a common terminology

1.1 Traveling Salesman Problem

This is how we define the Traveling Salesman Problem that we will try to efficiently solve during the rest of the project.

Definition 1. Given a complete weighted graph $G = \{V, E\}$ and a start node $s \in E$ the Traveling Salesman Problem consists in finding the shortest possible path p_* which starts and finishes at the same node s and visits all other nodes exactly once.

It exists another formulation of the problem which deals with non-complete graphs, but since a graph can be completed without changing the TSP solution by adding arbitrary length edges between the nodes that are not connected, we will stick to Definition 1.

It's important to state that the TSP problem is NP-Complete and that one of the fastest known exact algorithm takes $O(n^2 2^n)$. Also, the number of possible path defined by a TSP of dimension n is $\frac{(n-1)!}{2}$.

1.2 Genetic Algorithm

A genetic algorithm is a stochastic search algorithm which mimics Darwin's theory of evolution. This kind of algorithm is not guaranteed to find the best possible solution but it can efficiently find near optimal solutions. In this particular context a candidate solution is called chromosome. Let us now define its the basic components:

Fitness function
: Function which takes as input a chromosome and return a goodness score.

Crossover operator
: Function which takes as input two chromosomes and recombines them into two new chromosomes.

Mutation operator
: Function which takes as input one chromosomes and generates a new one by applying a random mutation

To create a genetic algorithm for our problem we just have define these three functions. Once we have them we can run the algorithm

Algorithm 1 Generic Algorithm

1: $population \leftarrow$ random population of k chromosomes
2: **for** $i = 0$ `to` g **do**
3: $\quad offspring \leftarrow evolve(population)$
4: $\quad population = select(population, offspring)$
5: **end for**

In Algorithm 1, the function $evolve$ generates k new chromosomes by applying, depending on a given probability, the crossover operator or the mutation operator. After that, the $select$ select takes the best k chromosomes among the old and the new population using the fitness function as a metric.
The main parameter of this algorithm are the population dimension k and the number of generations (iterations) g.

2 Solution

3 Sequential

4 Fastflow

5 Threads