Assignment 1: Genetic Algorithm for the Knapsack Problem

Guilherme Santos `fc62533`

October 12, 2023

Parallelization Strategies Used

My approach to parallelizing this problem was similar to the 'Embarrassingly Parallel Programming' exercises of the first week, where we divided the data, in this case the array of individuals, into chunks and processed each chunk on a separate thread. By dividing the data into smaller chunks, the workload can be distributed evenly across the cores, thereby ensuring that all cores are utilized effectively. First, I parallelized the function `populateInitialPopulationRandomly()` and the *'for loops'* that calculate the fitness, cross-over and mutation and noticed that the algorithm took longer to launch since `populateInitialPopulationRandomly()` ran faster in sequential mode than it did in parallel, so I decided to leave it in sequential mode. There are functions, such as `tournament(int tournamentSize, Random r)` and `bestOfPopulation()`, whose parallelization I don't believe would be beneficial or possible since in `bestOfPopulation()` all the threads would need to access the entire array to find the best individual, and in `tournament(int tournamentSize, Random r)` the parallelization seems complex and has a high risk of introducing bugs. The generations were not parallelized since each generation N requires generation N-1. Used `ThreadLocalRandom` instead of `java.util.Random` because the use of the same `java.util.Random` instance across the threads may encounter contetion and consequent poor performance.

Results

Execution time with `N_GENERATIONS=500` and `POP_SIZE=100000`

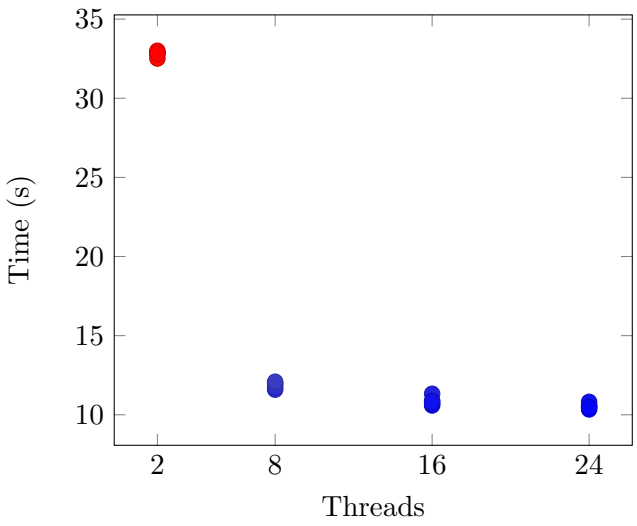

Figure 1: Parallel

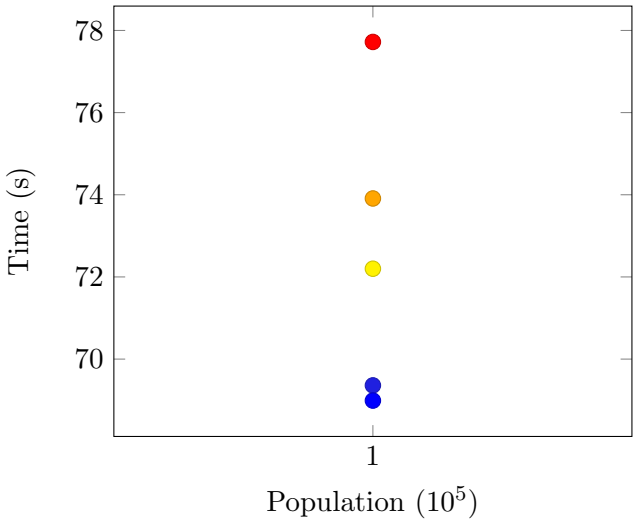

Figure 2: Sequential

According to Figure 1, the best execution time was with 24 threads.

Inprovement in performance:

$$SpeedUp = \frac{\overline{Sequential}}{\overline{Parallel}} = \frac{72.44s}{10.52s} \approx 7$$

As a result, we can say that the parallel version is 7 times faster than the sequential one.

The Mann-Whitney U test

```
[Running] python -u "c:\GitHub\PPC\KnapsackGA\src\scripts\mannwhitneyu.py"
U statistic: 25.0
P-value: 0.007936507936507936
```

Figure 3: Script's output

The U statistic is large and the p-value is small (less than 0.05), which suggests that the difference between the execution times is statistically significant and not due to random chance, indicating that the parallel algorithm is faster.

Environment

CPU	Cores	Threads	RAM	Operating System	Java
Intel Core i7-13700	16	24	32GB 6000MHz	Windows 11	21