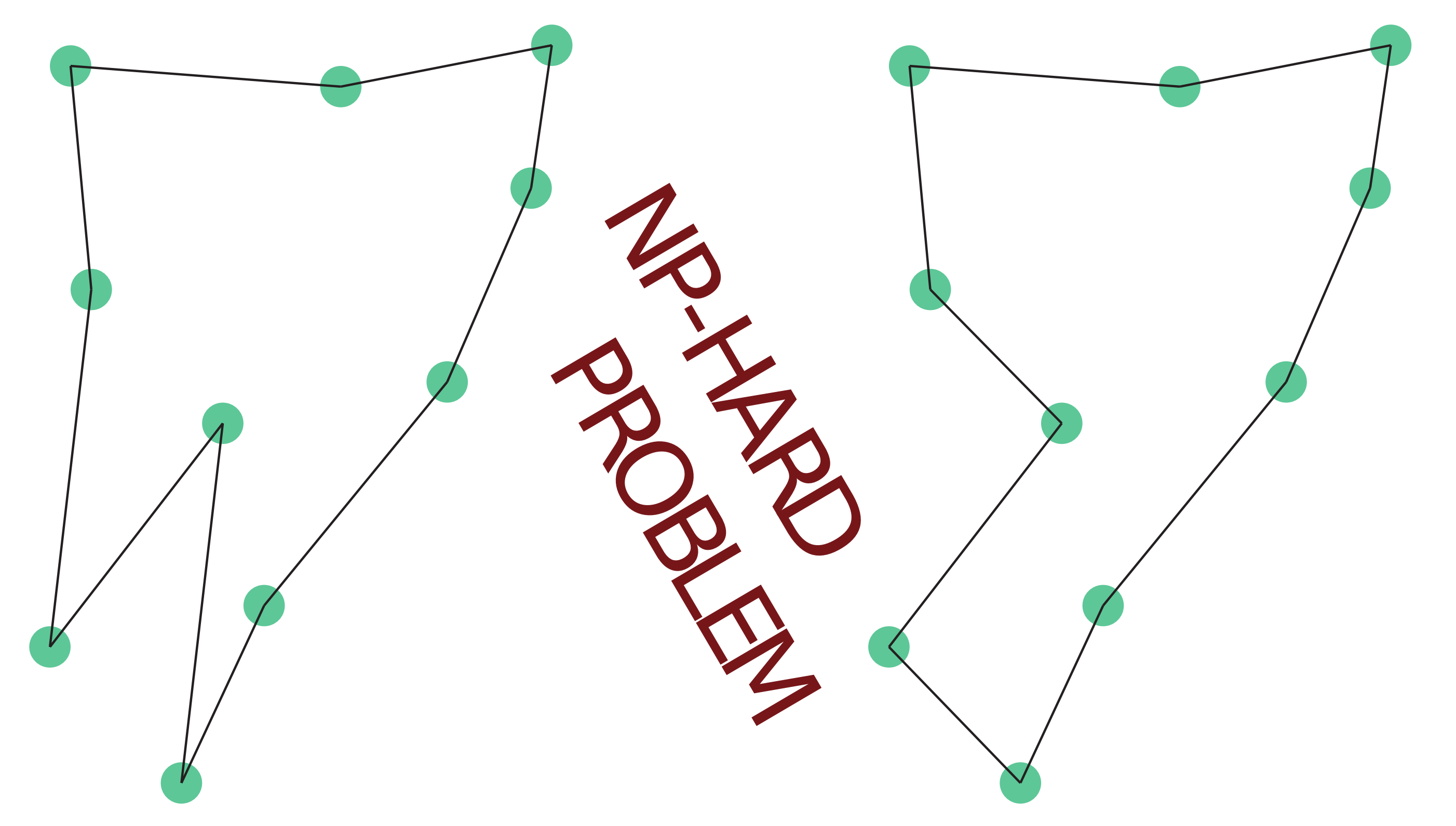
2018-2019
Academic Year

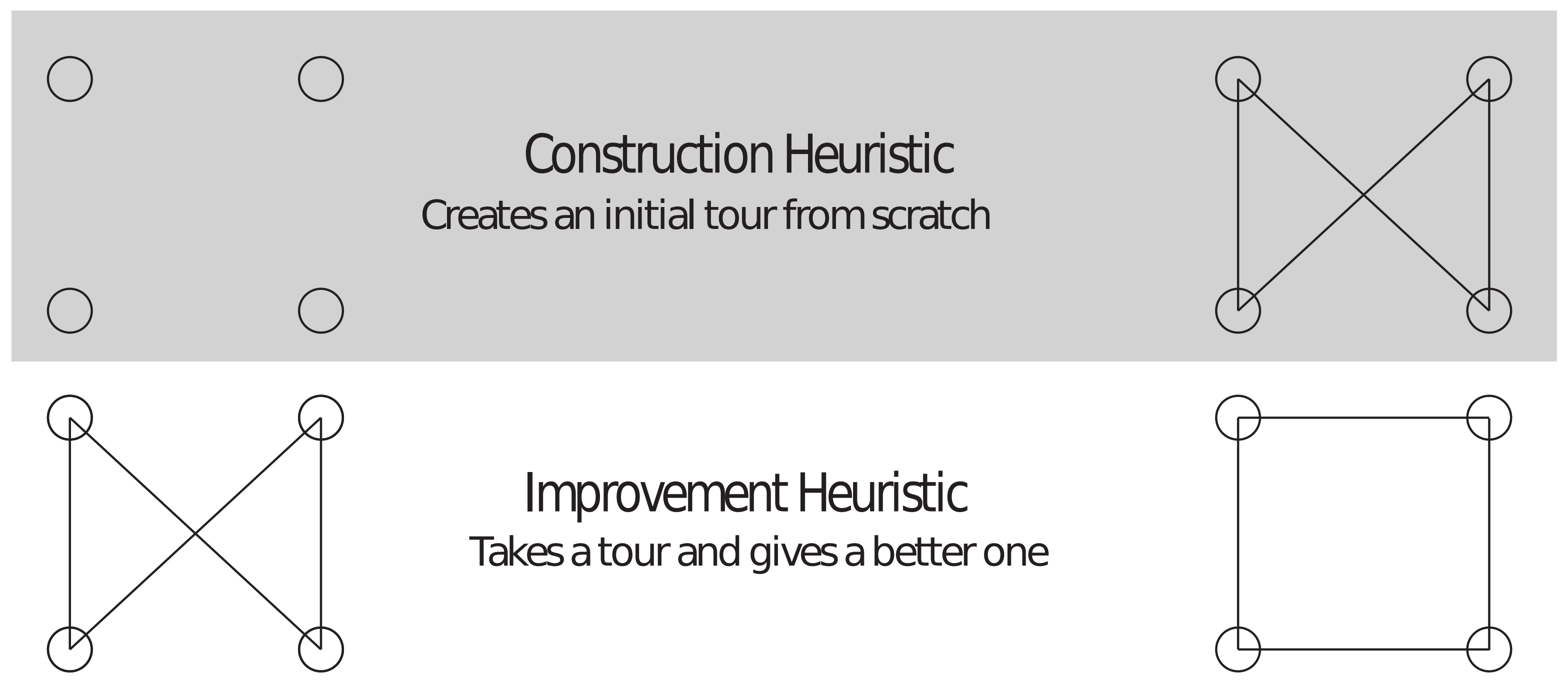
GPU Parallel Iterated Local Search for TSP

Hamdi Burak USUL
Academic Supervisor: Gülay YALÇIN ALKAN

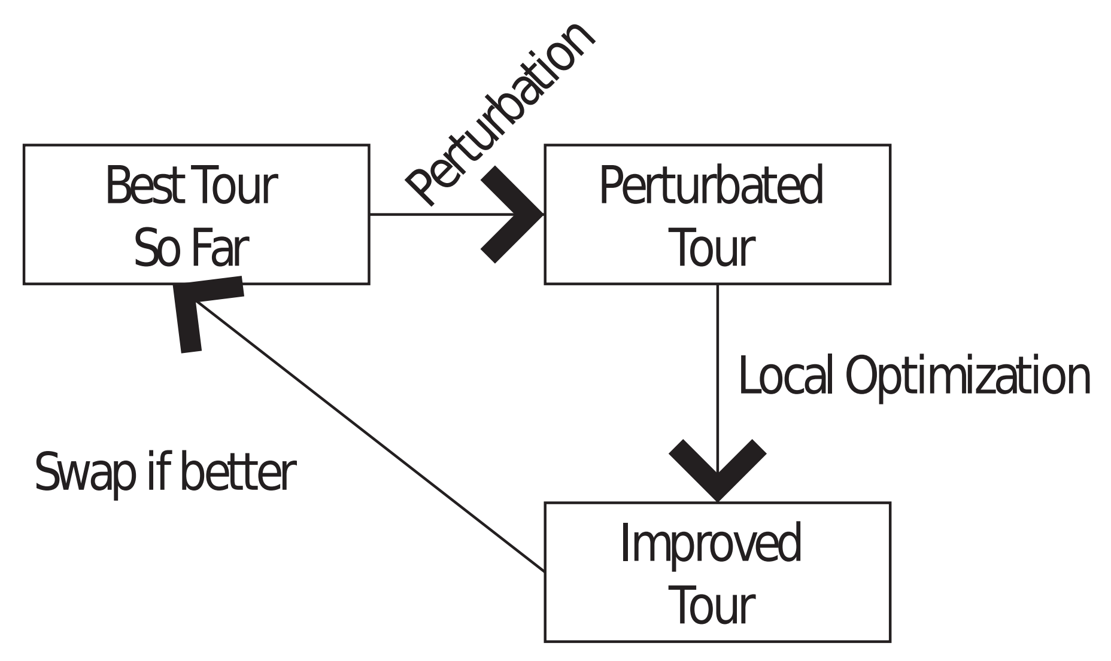
Visit each city once and return to the city you started which path gives the shortest distance?



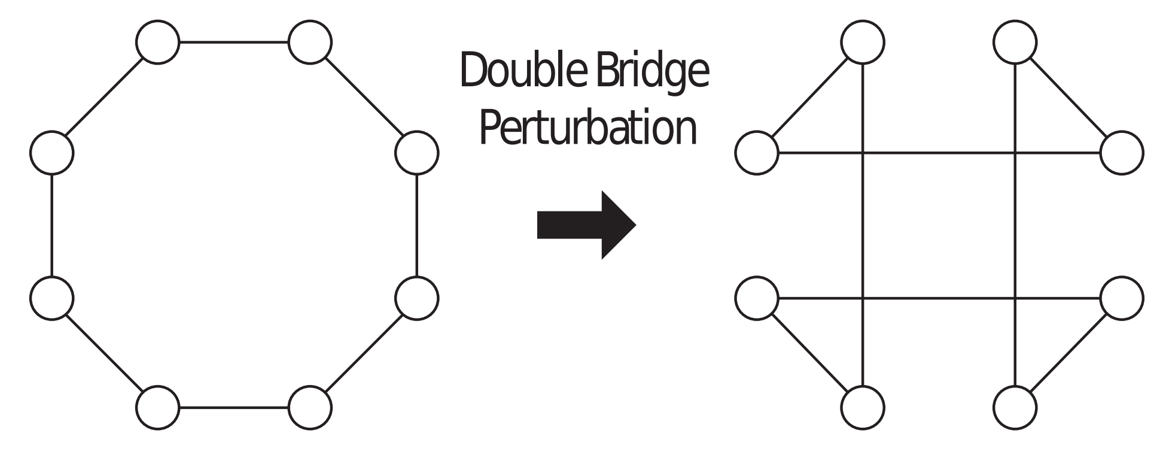
Takes a long time to solve and f nd the exact shortest distance(s). To cope up with this limitation, people settle for good enough solutions. Algorithms that f nd good enough solutions are called heuristics. Heuristics are divided into two groups:


Construction Heuristic
Creates an initial tour from scratch
Improvement Heuristic
Takes a tour and gives a better one

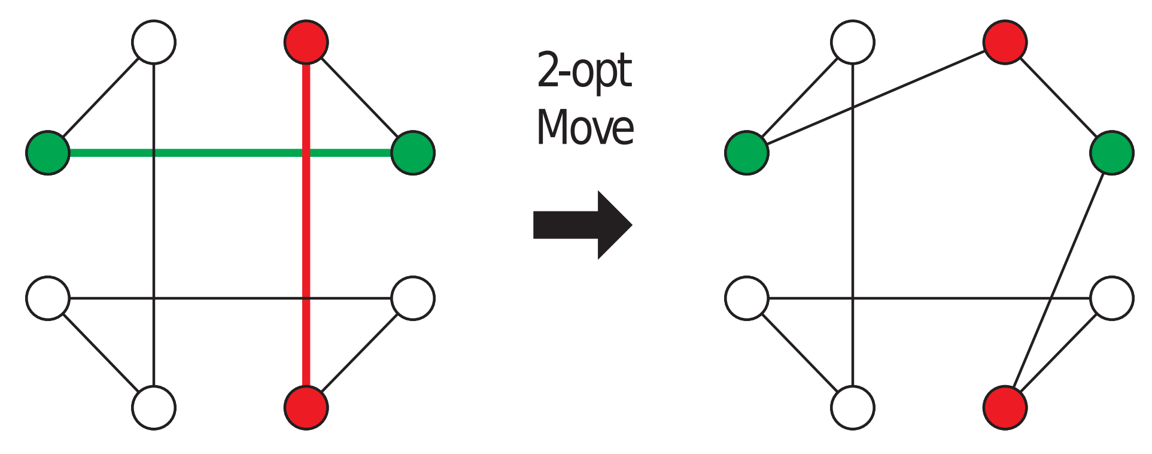
When the result of a heuristic is not good enough, it is a wide practice to run algorithm again with a dif erent initial solution and hope that it will give a better solution. However, when the result gets better, f nding a random initial solution that will lead to a better objective value(shorter distance) will get less likely. Because of this trend, choosing next initial solution randomly will lead to better solutions less and less. Helena et al describes Iterated Local Search, an alternative, by making a connection between the new initial solution and best solution so far. In iterated local search, instead of creating a random tour from scratch, the old tour is changed in a way that it cannot be f xed backed to original in a single move. This operation is called perturbation.



Iterated Local Search Procedure



An example of a perturburbation scheme



An example of a local optimization move

Problem Def nition

Iterated Local Search takes a long time to run.

Current Situation

Most of the time spent in ILS is spent on running local optimization algorithm so many researchers f gured out that it is a good idea to parallelize this section and gain a speed up.

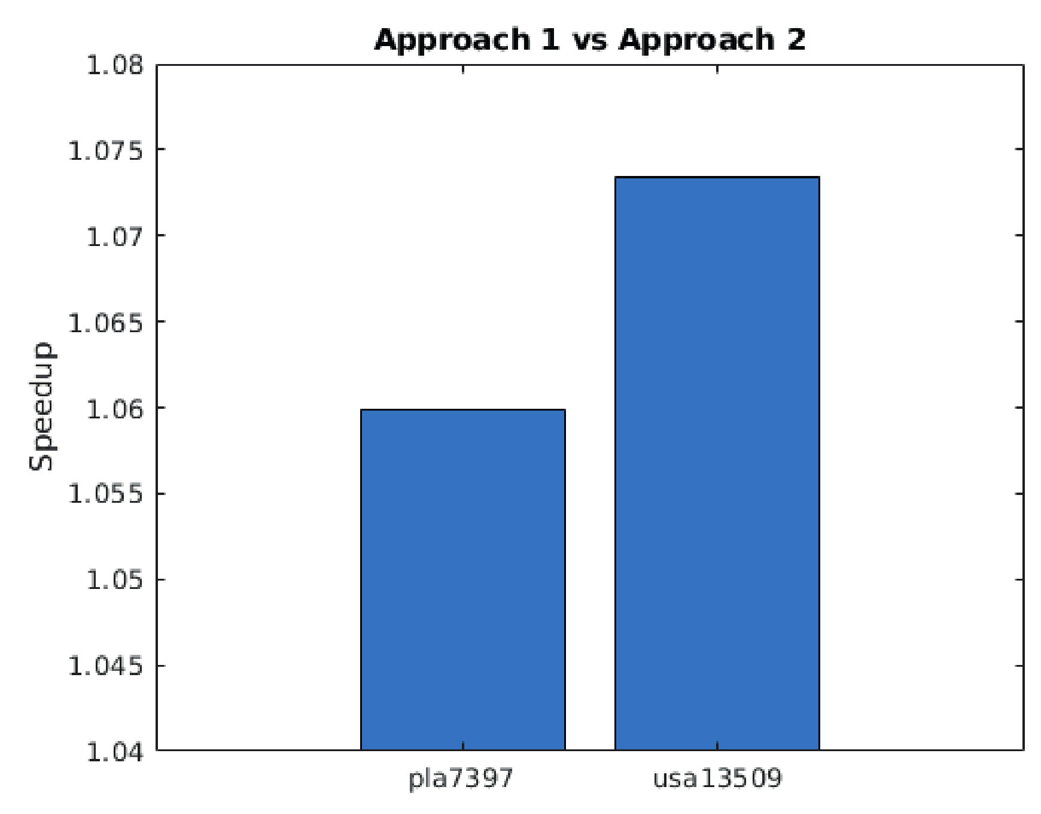
Our Proposition

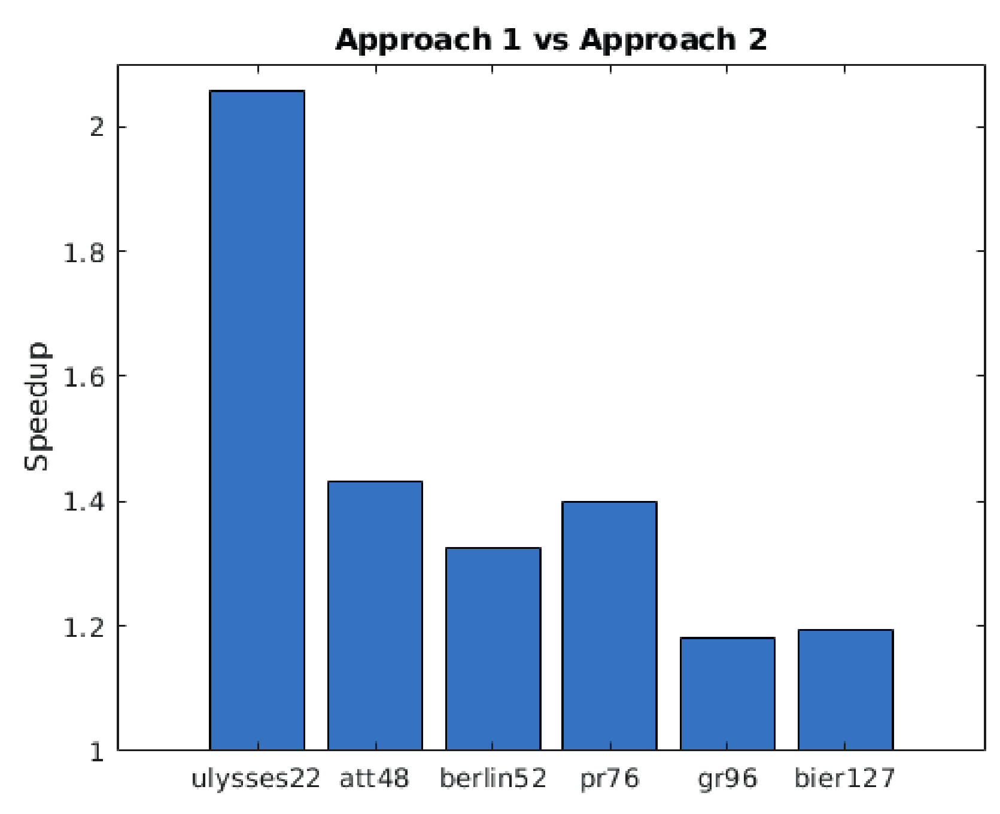
Parallelizing local optimization part is benef cial however, we propose that parallelizing each part yields better performance because of data will stay in GPU if everything is handled in GPU.

In addition to that, CUDA has a feature called dynamic parallelism which enables GPU to call itself instead of requiring CPU to call GPU. We also propose that use of dynamic parallelism will speedup our approach more.

Results

For GPU Parallel versions, CUDA extension for C++ is used. We compared parallelizing only local search and parallelizing the whole procedure. Speedup graphs are shown below. Tests for bigger instances are made possible thanks to Abdullah Gul University providing HPC access during the term. Results for dynamic parallelism are not ready. In the graph x axis represents problem instance names, where the number part determines the number of cities. Our parallel version is 80 times faster than CPU version on average. We did not wish to include those graphs since they do not represent the comparison with a state-of-art method.





Future Work

One of the biggest challenges using the GPU for parallel computing is that it has a small-fast and a big-slow memory. When problem size gets bigger, problems are divided into chunks which we observe to cause a great slowdown. The reason of slowdown is the change of data structure that is kept. We will try to tackle this problem by using compression(k-means) which will help us to f t more coordinates into shared memory(small-fast one).

Computer
Engineering