GPU Ant Colony Optimisation for Traveling Salesman Problem — CUDA Assignment

Otwarto: wtorek, 1 kwietnia 2025, 14:14
Wymagane do: poniedziałek, 28 kwietnia 2025, 23:59

GPU Ant Colony Optimisation for Traveling Salesman Problem — CUDA Assignment

Autor: Adrian Naruszko

Introduction

The **Traveling Salesman Problem (TSP)** is a classical NP-hard optimization problem where a salesman must visit each city exactly once and return to the starting point, minimizing the total travel distance. Even small instances of TSP can be computationally challenging, making it an excellent target for parallel and GPU-based computation.

One popular metaheuristic to solve TSP is **Ant Colony Optimization (ACO)**. It simulates a swarm of artificial ants that explore paths through the graph using both random exploration and accumulated *pheromone trails* to reinforce better solutions. Over many iterations, ants collectively converge toward near-optimal solutions.

In this assignment, you will implement and compare two different GPU-based ACO variants and analyze their performance.

Objectives

- Implement the **Worker Ant baseline version** where one thread simulates one ant independently.
- Implement the **Queen Ant version**, where each block builds a tour cooperatively using one thread per city (see paper below).
- In both versions you must calculate cycle tour length (with reasonable error, explained later) and list the elements of the cycle. Each city must appear exactly once in a cycle. Cycle tour length must equal the sum of distances along the cycle.
- You may (or even should) implement your solution with mutliple kernels and run then in a loop by NUM_ITER value. All core computations must happen on GPU.
- Wrap the core loop of both implementations using **CUDA Graphs** for efficient execution.
- Explore memory and warp-level optimizations where appropriate.
- Compare the correctness, performance, and scalability of all implementations.
- You may use features that come with default nvcc distribution. No external libraries are allowed. If you use anything out of the lecture scope, please remember to explain it in the report.

Reference Paper

This assignment is based on the techniques described in the following paper:

Enhancing Data Parallelism for Ant Colony Optimisation on GPUs
Jose M. Cecilia, Jose M. Garcıa
https://doi.org/10.1016/j.jpdc.2012.01.002 (also attached as PDF)

Recommended Sections to Read:

- **Section 2.1:** Math needed in the task - look at equations 1), 2), 3) and 4).

?

- **Section 3.1.2:** Classic thread-per-ant baseline (matches your first implementation).
- **Section 3.2.2:** Queen Ant model (block = ant, thread = city).
- **Section 3.2.3**: Description of roulette mechanisms.
- **Section 3.3:** Pheromone update description.
- **Section 4:** Experiments — you may use similar input instances for benchmarking, but see constraints below.

Technical Details

- **City count is limited to 1024** to fit within a single CUDA thread block in the Queen Ant model. Thanks to that you can ignore the *tiling* concept introduced in **3.2.2.**
- For performance tests and comparison please use the same datasets as in the article (**Table 4**). Note that the number in the name represents cities count, so use those with appropriate cities count only. Datasets from Table 4 are linked below in the input section. Note, however, that your solution must work correctly even on datasets not listed there.

Input Format

Input files represent a file in a TSPLIB format.

```
NAME : example3
COMMENT : Example TSP input
TYPE : TSP
DIMENSION : 3
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 3 5
2 4.5 -6
3 8.62630e+02 5
EOF
```

First 6 lines contain metadata which you can safely ignore. If you are using input from the internet, make sure that EDGE_WEIGHT_TYPE is EUC_2D, CEIL_2D or GEO. Otherwise the rest of the file will have different format. However, your program will be tested only on supported types. Next lines are of form

```
A B C
```

Where A is the number of the city (it will always be consecutive natural numbers, starting from 1) and (B, C) are A-th city coordinates. Coordinates may be integers, doubles or doubles in scientific format. You should first map those values to a square matrix with distances between each pair of the cities. As this step is a preprocess, you can do this on CPU.
Input ends on EOF or, equivalently, after DIMENSION cities.
You may also assume that no two points overlap.

Output Format

First line of the output should contain single value - the length of the shortest cycle found, answering the TSP problem. Second line should contain space-separated citites numbers on the shortest cycle, starting at 1. Each city must appear exactly once on this list.

Test Data

You may find more examples and test data here. Note that there are also best known solutions in the respository. Please note that some of the inputs there have different format and thus won't work with your solution. Some of the good ones are d198, a280, lin318, pcb442, rat783 or pr1002.

Compilation and Execution

Submit a single `.zip` archive with a directory named `ab123456/` (replace with your login).

We will build and run your program on entropy as follows:

```
unzip <your_login>.zip
cd <your_login>
make clean
make

./acotsp <input_file> <output_file> <TYPE> <NUM_ITER> <ALPHA> <BETA> <EVAPORATE> <SEED>
```

- Input and output files were described above.
- TYPE will be one of WORKER, QUEEN. You should run proper implementation depending on this value.
- NUM_ITER is the number of iterations your program should run.
- ALPHA and BETA are doubles used in the pheromone value transformation (equation 1)), wherease EVAPORATE is a factor used to diminish pheromones in equation 2).
- SEED is a randomness seed, should be used for curand initialization.

In the study reserachers used values NUM_ITER=number of cities, ALPHA=1, BETA=2, EVAPORATE=0.5, so these should be our default values.
As the problem is a heuristic, there is no definite answer to the given input. You may use solutions to the known datasets along with **Figure 9** from the article to estimate the expected range. Solutions will score binarily on each test, thus its enough to be reasonably close to the expected value. No further points will be assigned for a solution that behaves better.

Roulette mechanism

After we assign probabilities to cities, we must pick one of them randomly according to obtained distribution.

In the article the Independent Roulette mechanism is introduced, **but please do not use this mechanism.** It is another heuristic, we do not want to pile them up.

Instead use the simple sequential roulette. You may try to optimise it using warp level reduction.

Another problem is the random number generation in CUDA, that was not mentioned during lectures. Here is an example:

```
#include <stdio.h>
#include <curand_kernel.h>

__global__ void init_rng(curandState* states, unsigned long seed, int n) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;

    // Seed is the random seed, idx is like "subseed"
    // Each thread is thus initialized with different value and has its own currandState variable.
    curand_init(seed, idx, 0, &states[idx]);
}

__global__ void generate_random(float* result, curandState* states, int n) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;

    result[idx] = curand_uniform(&states[idx]);  // generate float in (0, 1]
}

int main() {
    ...
    curandState* d_states;
    cudaMalloc(&d_states, sizeof(curandState) * N);

    init_rng<<<1, N>>>(d_states, SEED, N);
    cudaDeviceSynchronize();

    for (int i = 0; i < 3; ++i) {
        generate_random<<<1, N>>>(d_random, d_states, N);
    }
    ...
}
```

Evaluation Criteria

Component	Points
Correct implementation of Worker Ant version	6
Correct implementation of Queen Ant version	7
CUDA Graph wrapping for both pipelines	1
Performance & Optimisations	6
Report	5

Your solution must score at least 8 points to complete the course. Points for performance and report will ge given only if the solution is *mostly* correct. If only one of the algorithms is implemented those points will be divided accordingly.

Report Requirements

Your report must include:

- Explanation of both implementations (thread/block layouts and synchronization).
- Optimisations applied (e.g. warp-level reductions, memory layout, reuse of data).
- Runtime comparison of both implementations and their CUDA Graph variants.
- Performance measurements on various instances (see Table 4 from the article).
- Explanation of CUDA features outside of the lecture scope.

Deadline

Please do submit your assignment by the due date. If you're late, your score will be reduced by 1 point for every 12 hours (i.e.: if you're late by 2h, we subtract 1 point from your score; if you're late by 25h, we subtract 3 points).

There is a second due date - a week after the first one. Submitting by this due date is very risky. First, very good solutions submitted by this due date receive at most 10 points. Second, there will be less time for patching. Please do submit your assignment by the normal, first due date.

Notes

- There seem to be an error on Figure 2 - Multiplication of previous values is not correct; It's copied from Figure 3. Instead should be multiplication of two rows above.

FAQ

15.04.2025:

- Q: How close should the results be to be treated as "reasonably close"?
  A: Let's assume that for the tests mentioned on the Figure 9 in the article, your program should have relative error up to 50% more than the the worst shown result rounded up to 0.05 (i.e. for d198 acceptable quality is 1.5 * (1.15 - 1) + 1 = 1.23) compared to the solutions.
- Q: Is thrust library allowed? It is distributed with the nvcc.
  A: It is distributed with the nvcc, so yes, you can use it.
- Q: Can we assume that input arguments are correct?
  A: Yes, your program will be tested only on correct arguments (and input files)
- Q: Can both implementations use the same code for pheromones update?
  A: Yes, this part does not differ in both solutions.
- Q: Should number of threads be equal in the pheromone update and build path kernels?
  A: No, we focus on changes in build path kernel, pheromone update may use different threads setting.
- Q: Should we include both optimised and non-optimised versions of the kernels?
  A: You can leave only the optimised versions; In particular it should be clear which version should be scored.
- Q: Do we need to strictly follow the algorithm description from the article?
  A: In general - yes; However, if you note some technical inefficiency and you can express the same thing in other, simpler terms, you can use it as an improvement.
- Q: I found different set of input parameters, that work best for my solution. Can I use them?
  A: Yes, you can even include them in the report so that they will be used for testing. They need to be static, they

cannot change depending on the input files.

- Q: Is Scatter to Gather algorithm (Section 3.3) required or is it an improvement?
  A: It is a possible improvement

jpdc-num2.pdf 31 marca 2025, 21:51

Dodaj pracę

Status przesłanego zadania

Numer próby	To jest próba nr 1.
Status przesłanego zadania	Nie przesłano jeszcze zadania
Stan oceniania	Nieocenione
Pozostały czas	Pozostało 13 dni 10 godzin
Ostatnio modyfikowane	-
Komentarz do przesłanego zadania	Komentarze (0)