My Solution in ICGCA

Team: Drifters, Author: Kohei Ono (DeNA)

Thank you for organizing the competition!

1. Explanation of My Solution

The main strategy of my solution in the ICGCA competition is to construct Zero-suppressed Binary Decision Diagrams (ZDDs) [1] using frontier-based search method (Simpath algorithm [2]), and then count the number of paths.

My method is divided into several steps: preprocessing through filtering (in cases of one_pair), determining variable order through BeamSearch, and building ZDDs with frontier-based search method.

1.1 Preprocessing with Filtering

In one_pair test cases, there may be vertices that are irrelevant to the solution.

Let's denote the distance between two points u and v as $\mathrm{dist}[u][v]$. We can ignore vertices v that satisfy $\mathrm{dist}[s][v] + \mathrm{dist}[t][v] > l$ because they will not be included in the path. Therefore, in my method, such vertices are filtered out in advance to reduce the computational time.

Also, my algorithm pre-calculate the array dist that holds the distances between all pairs of vertices, as it will be used for pruning later (even in cases of all_pairs).

The algorithm to calculate dist is the Warshall Floyd algorithm [3].

1.2 Determining Variable Order with a BeamSearch

In the frontier-based search, the order of the variables to be processed needs to be determined in advance. This variable order significantly affects the performance of the frontier-based search, so it’s very important. I optimized the variable order to minimize the size of the vertex set referred to as the frontier.

Using BeamSearch to determine the variable order in the frontier-based search is a well-researched topic [4], and my method is similar in spirit.

Features of BeamSearch implemented in my method are as follows.

- It optimizes the edge order instead of the vertex order.

- The evaluation function is $\sum_i |F_i|^3$, where F_i is i-th frontier.
- It uses Chokudai Search, a variant of BeamSearch.

Chokudai Search is a well-known technique in the competitive programming community, and to put it simply, it is BeamSearch that increases the beam width sequentially. (The academic source is unknown.)

Compared to ordinary BeamSearch, it is easy to ensure diversity even with a simple evaluation function and can manage execution time easier.

In this competition, I used Chokudai Search the first 100 seconds to optimizing the edge order.

1.3 Constructing ZDDs with the Frontier-Based Search

A ZDD is a data structure that can compress and store a set of graphs. In this competition, I construct a ZDD that compresses and represents all paths that meet the conditions, and obtain the number of paths.

The basic idea is almost the same as the Simpath algorithm by Knuth. I implemented it using TdZdd [4] and turned on the parallelization option using OpenMP.

Features of the frontier-based search implemented in my method are as follows.

- The algorithm stores the number of used edges in states in addition to the ordinary Simpath algorithm.
- Determine the lower bound of the number of edges required to merge multiple paths in a way the traveling salesman problem and prune if it takes more than l edges.

2. Unimplemented techniques

2.1 Parameter Optimization

I could not prepare an execution environment similar to the evaluation environment, so I could not run sufficient tests on the benchmark.

Therefore, although there are several parameters in my solution, I was not able to optimize them sufficiently. (I should have optimized with Optuna or similar if time allowed)

2.2 The Frontier-Based Search without Constructing ZDDs

In this problem, I am constructing ZDDs just to count the total number of paths.

The total number of paths can be found by running dynamic programming on the ZDD, but this operation can be done in the middle of the construction. In other words, ZDD vertices that are no longer needed during the frontier-based search can be discarded.

This improvement should critically work from the perspective of memory consumption. However, since I implemented it using TdZdd and could not find such a function in TdZdd.

[1]: Shin-ichi Minato. Zero-suppressed bdds for set manipulation in combinatorial problems. In Alfred E. Dunlop, editor, Proceedings of the 30th Design Automation Conference. Dallas, Texas, USA, June 14-18, 1993., pages 272–277. ACM Press, 1993.

[2]: Donald E Knuth. The art of computer programming, volume 4A: combinatorial algorithms, part1. Pearson Education India, 2011.

[3]: Robert W Floyd. Algorithm 97: shortest path. Communications of the ACM, 5(6):345, 1962.

[4]: https://github.com/kunisura/TdZdd