Solver Description : PaCo *

Sylwester Swat
Institute of Computing Science, Poznań University of Technology, Poland
sylwester.swat@put.poznan.pl

Abstract

This article briefly describes the most important algorithms and techniques used in the path counting solver called "PaCo", submitted to the second International Competition on Graph Counting Algorithms (ICGCA 2024).

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Path counting, pathwidth decomposition, meet-in-the-middle, backtracking, dynamic programming, reduction rules

Category ICGCA Solver Description

1 Problem description

In the path counting problem we are given a directed or undirected simple graph G, terminal nodes vs, vt and an integer L and the goal is to find the number of simple paths that begin in node vs, end in node vt and have length at most L.

2 Solver description

In this paper we provide a short description of the most important algorithms implemented in solver PaCo. Due to a large variety of used methods, this description does not contain full information about used algorithms, their details and their behaviour in many distinct situations. The workflow of Paco can be described in the following general steps:

1. Reduce the graph using data reduction rules.
2. Create basic graph characteristics and determine a set of algorithms that will be used for result calculation.
3. Run algorithm(s) to calculate the result.

Solver PaCo was designed and implemented to work for directed graphs. It also works for undirected graph, as these can be simply treated as directed ones. Depending on the structure of the graph, however, used algorithms and optimization techniques might differ, resulting in a (sometimes very significant) different performance.

3 Preprocessing

We use several data reduction rules to reduce graph size or simplify the structure in a way that would likely make it easier for one or several of implemented approaches to efficiently calculate desired number of paths. To the most common data reduction rules we can include removal of nodes with degree one (except for terminal nodes) or removal of all vertices v, whose sum of distances $dst(vs, v) + dst(v, vt) > L$, as these clearly cannot belong to a path of length at most L. Similarly, an arc (u, v) can be removed if the condition $dst(vs, u) + 1 + dst(v, vt) > L$ holds. Depending on the graph structure, particularly when dealing with undirected graphs, some data reduction rules must be used with caution or cannot be used at all, as they

* This is a brief description of a solver submitted to the ICGCA 2024 contest.

might make the graph directed - e.g. mentioned distanct-arc-removal rule might remove arc (u, v) but leave arc (v, u). Such behaviour might make the graph's structure smaller, but it also makes it impossible to use optimization techniques (mainly state-trimming) designed specifically for undirected graphs in proper path counting methods.

4 Path counting methods

There are several implemented approaches for the problem. Each of these approaches can be classified into one of the three following, general groups:

1. backtracking
2. meet-in-the-middle dynamic programming
3. dynamic programming on pathwidth decomposition

PaCo contains implementation of several backtracking algorithms for the problem, but one of them is clearly dominant over the others. Several optimization techniques are used to speed up result calculation. To these techniques we can include distance trimming, result caching and on-the-fly graph structure reduction (using some of the reduction rules). For each of these techniques there are variations differing in approach and general performance for graphs with distinct characteristics. For example, distance trimming can be done using any approach to SSSP problem. In PaCo BFS was used for graphs where distances could be calculated using this method (edges had no weight or weights of all edges are equal), SPFA algorithm for quick calculation of distances from single source node or Dijkstra algorithm for calculation of number of shortest paths, whenever need to calculate those occured. Additionally, Customizable Contraction Hierarchies algorithm was used for faster distance-queries n dynamically chaning graph, but from our tests it follows that its advantage over SPFA starts to occur only for graphs too large to consider for path counting problem (or at least too large to expect that any algorithm could solve instances of that size in general in feasible time).

A dynamic programming based on the meet-in-the-middle approach was implemented and included in PaCo. This approach seems to be better than the backtracking algorithm for some instances, especially those where the maximum considered path length is small. What is worth mentioning is that structure of the graphs plays a significant role here. There are instances, even of very of very large graphs, that can be solved fairly quickly by this approach due to applied state-trimming techniques. Those are, however, very sensitive to the graph structure and do not seem to exhibit any particular easily-observable nature (e.g. graph G might be easy for the meet-in-the-middle method, graph $G[E \bigcup \{e_1\}]$ might be hard to solve, and graph $G[E \bigcup \{e_1, e_2\}]$ might again be quickly solvable. The main disadvantage of of the meet-in-the-middle approach compared to the bracktracking one is its much higher memory consumption, as it needs to create and keep in memory states (subsets of vertices) and counts of paths for those states.

Third group contains algorithms that use dynamic programming on pathwidth decomposition (or at least we think this is it). The idea is as follows: starting with a set S initially containing a single node vs, we iteratively add or remove nodes from S in such a way, that S is a $vs - vt$ separator, dynamically keeping track of the intersection of set S with a set of all possible $vs - vt$-paths of length at most L and cumulative numbers of those paths. At the end, the cumulative value for set $S = \{vt\}$ is the final answer.

5 Additional information and acknowledgements

All algorithms created and implemented within PaCo are of our own design and invention. We are aware that many distinct approaches for the path counting problem exist, but we treated the contest as a chance to have some fun and abstract from daily duties, without need for a more "scientifically correct" approach. Our knowledge of the literature in the path counting field is therefore limited to the report from ICGCA 2023 [1] and brief descriptions of solvers submitted by participants [TLDC, diodrm, TAG, Drifters, NaPS+GPMC, KUT_KMHT, asprune, castella, dimitri, Cipher, PCSSPC]. We would like to offer here particular thanks to the authors of [TAG] solver for including table with results of their methods in their description - these results made it very easy and pleasant for us to test our implemented approaches without need of conducting additional experiments and provided us a great motivation to keep improving. Without these tables we would most probably cease at some point early on and lose a lot of fun we had while designing and improving our solver.

Some algorithms implemented within PaCo solver have been significantly improved since the end of the contest (especially version for undirected graphs). We would be very happy to take part in future editions of ICGCA contest with the hope to have a motivation to further improve our solver and make it more competitive and (hopefully) make some progress in the state-of-the-art in the field of path counting problem, however small it might be.

Many thanks to the Organizers for organizing ICGCA 2024 - we had a great time taking part in the competition!

References

1 Takeru Inoue, Norihito Yasuda, Hidetomo Nabeshima, Masaaki Nishino, Shuhei Denzumi, and Shin-Ichi Minato. International Competition on Graph Counting Algorithms 2023. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, 107(9):1441–1451, September 2024. arXiv:2309.07381, doi:10.1587/transfun.2023DMP0006.