**Research on Efficient Hop-Constrained s-t Simple Path Enumeration**

Author: Shi Wenlan

**Abstract:**

We study the hop constraint s-t path enumeration problem: Given a graph G, enumerate all simple paths p from a source vertex s to a target vertex t with the number of hops not larger than k. The state-of-the-art still leaves some interesting future work. And we want to further develop a better algorithm.

**Acknowledgement:**

This project is supported by the grant “Undergraduate Research Awards” awarded by Undergraduate Research Awards Selection Committee in 2021.

1. **Introduction**

The graph is a data structure used in various areas. It represents information of different entities(vertices) and their relationships(edges). To evaluate the relationship between two entities, we need to enumerate simple paths from one entity (vertex s) to another entity (vertex t). Intuitively, the longer the path, the weaker the relationship between two entities, and enumerating long paths often requires a lot of calculation. So, it is natural to impose a hop constraint k to s-t path enumeration.

To enumerate all simple paths from s to t with the number of hops less than or equal to k in polynomial delay, Peng et al. (2019) proposed Barrier-based constrained DFS (BC-DFS). However, barrier update incurs high overhead, so it is still not fast enough. To meet the rigid time constraint in real-world applications, Sun et al. (2021) proposed the PathEnum algorithm. This algorithm proposed new research ideas such as a lightweight index and join optimizer, and there is still a lot of room for optimization.

If this problem can be solved, this technology can be used in many areas such as detecting money laundering and build knowledge networks, and have good development prospects.

1. **Problem Statement and Formulation**

Hop constraint s-t path enumeration: Given a graph G, enumerate all simple paths p from a source vertex s to a target vertex t with the number of hops not larger than k.

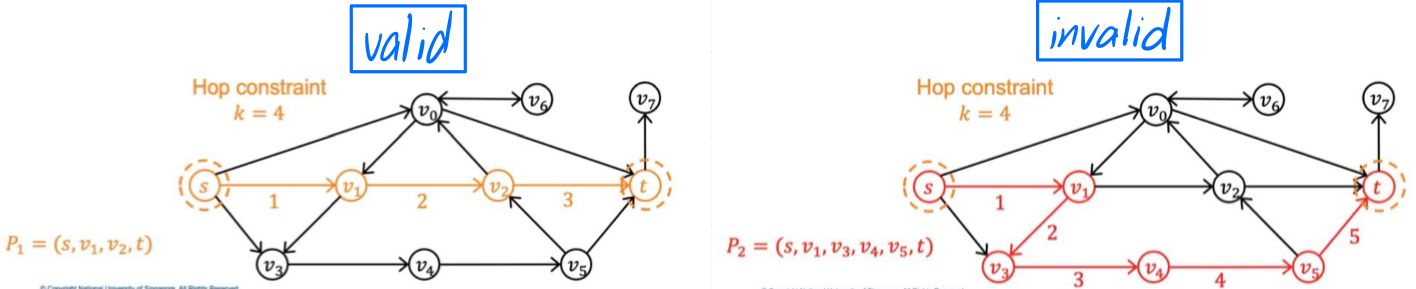


Figure 1: path sample

Figure 1 shows the sample of path that we need.

In this example, for given k = 4, the path we need to enumerate are: s-v0-t, s-v1-v2-t, s-v1-v2-v0-t, s-v3-v4-v5-t and s-v0-v1-v2-t.

And we want to enumerate these paths as fast as possible.

1. **Progress Made (Intermediate Results)**

In the past cycle, I met with professors every week to discuss research.

I measured the time-consuming of PathEnum when k=5. At this time, the construction of index accounted for the main time-consuming. So, during this period, my main focus was to build an index offline to provide additional information to the process of building an index online, thereby accelerating the process of building an index online.

During the period, I studied the papers and source code about pruned-2-hop-labeling. I modified the source code by Akiba et al. (2013) originally used for undirected graphs into a version suitable for directed graphs, and incorporated it into the subgraph construction part of PathEnum to speed up the construction of the index.

Pruned-2-hop-labeling is based on the notion of distance labeling or distance-aware 2-hop cover. The most basic idea is to store the distance to other points in the label of each point, and then compare the label data stored in the starting point s and the end point t one by one, find a point a in the point set of the intersection part, where distance from s to a plus distance from a to t is the shortest. Then we get the length of the shortest path from s to t, which is distance from s to a plus distance from a to t. In addition, some techniques are used to make each point save less labels without affecting the correctness of the results, so that the memory usage of the label is relatively acceptable.

My idea of speeding up is to use this label to quickly prune some branches when constructing the subgraph with linear time complexity, so that the number of points detected during the construction of the subgraph is reduced and thus speeded up. But the result of the experiment is that on average, only 3% of the points can be pruned, and the accelerated part fails to cover the overhead that was brought when the label was checked.

Later, when I increased k to 7, I found that the time to construct the subgraph did not change significantly, but the time to search for the path on the subgraph increased exponentially and occupied the main time-consuming, so I decided to turn the direction of the focus to use parallel algorithms to accelerate path enumeration part. This direction can definitely accelerate the process and get promising result.

1. **Research Plan & Expected Outcome**

Month 1: Master parallel algorithms and modify the code.

I plan to create threads in the style of breadth first search in the first few steps of path enumeration, and then each thread executes the depth first search branch independently.

Month 2: Experimental evaluation and algorithm optimization.

Dataset plan to use: US Patents, DBpedia, Web-google, Web-standford, Twitter-social, Baidu-baike, Wiki-trust, Soc-Epinsion1, Web-uk-2005, WikiTalk, Soc-Slashdot0922, LiveJournal, Rec-dating, Bio-grid-yeast and Twitter-mpi.

Month 3: Algorithm optimization and paper writing.

1. **Appendix (if necessary)**

(Supporting documents can be put here)

1. **References**

Peng, Y., Zhang, Y., Lin, X., Zhang, W., Qin, L., & Zhou, J. (2019). Hop-constrained st Simple Path Enumeration: Towards Bridging Theory and Practice. Proc. VLDB Endow., 13(4), 463-476.

Sun, S., Chen, Y., He, B., & Hooi, B. (2021, June). PathEnum: Towards Real-Time Hop-Constrained st Path Enumeration. In Proceedings of the 2021 International Conference on Management of Data (pp. 1758-1770).

Akiba, T., Iwata, Y., & Yoshida, Y. (2013, June). Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (pp. 349-360).