N-stella

Kyosuke Nishizawa[1] and Yu Nakahata[2]

[1]National Institute of Technology (KOSEN), Nara College
[2]Nara Institute of Science and Technology

1 Introduction

The length-limited path counting problem is defined as follows.

Input A undirected graph $G = (V, E)$ with vertex set V and edge set $E \subseteq V \times V$, terminals $s, t \in V$, and a maximum length $0 \leq l$.

Output The number of simple s-t paths in G whose length is at most l.

First, we process the input graph to reduce the graph size. After the edge ordering optimization, we construct a ZDD (Zero-suppressed Decision Diagram) that represents s-t paths and count the number of them.

2 Reducing Graph Size

To reduce the size of the input graph, we convert the input graph $G = (V, E)$ to a weighted graph $G' = (V', E', w, c)$. Each edge e in G' has two weights: $w(e)$ and $c(e)$. $w(e)$ represents the length of the edge e and $c(e)$ represents the number of multiple edges represented by e. We consider the input graph G as a weighted graph $G' = (V', E', w, c)$ where every edge $e \in E'$ has weights $w(e) = 1$ and $c(e) = 1$.

We processed the input graph G' to reduce its size as follows. Using these algorithms, we could reduce the number of vertices by 72% and the number of edges by 58% on average.

Length Constraint

First, compute the shortest distance between all vertices $v \in V'$ and s, t. Then, delete the vertices $v \in V'$ that $d(s, v) + d(v, t) > l$.

Contracting Degree-2 Vertices

We contract the degree-2 vertices except for s, t. Let $N(v)$ be the neighbourhood of $v \in V'$. If $N(v) = \{u, w\}$, we delete v, $e = (u, v)$, and $e' = (v, w)$. Then, we add a new edge $e'' = (u, w)$ whose weights are $w(e'') = w(e) + w(e')$ and $c(e'') = c(e) \times c(e')$.

Reducing Multiple Edges

When multiple edges exist between u and v and they have the same length, we reduce them to one edge. Specifically, when $e = e' = \{u, v\}$ and $w(e) = w(e')$, add a new edge $e'' = \{u, v\}$ whose weighs are $w(e'') = w(e') = w(e)$ and $c(e'') = c(e') + c(e)$.

Block-cut tree

Block-cut tree is a tree of biconnected components. Any connected graph can be decomposed into them. Any biconnected component that is not on the path from the biconnected component including s to the biconnected component including t can be deleted. We use Tarjan's algorithm for constructing block-cut tree.

3 Optimizing Edge Ordering

The complexity of the frontier-based search depends on the frontier size. So, optimizing edge ordering is important to reduce the frontier size and memory consumption. We use Chokudai Search code written by 'Drifters'[1], a contestant in the ICGCA 2023.

4 Constructing ZDD and Counting s-t Paths

We construct a ZDD $\mathcal{Z} = (\mathcal{N}, \mathcal{A})$ representing all s-t paths in G' whose length is at most l using frontier-based search. After the construction, we count the number of s-t paths by bottom-up dynamic programming as follows.

Each node $\alpha \in \mathcal{N}$ stores $C(\alpha)$. The values of 0-terminal and 1-terminal are initialized to $C(0) = 0, C(1) = 1$. We process the nodes in the reverse topological order (i.e., from the terminals to the root). For each non-terminal node $\alpha \in \mathcal{N} \setminus \{0, 1\}$ whose label is $e \in E'$, $C(\alpha)$ is computed as follows:

$$C(\alpha) = C(\alpha_0) + C(\alpha_1) \times c(e),$$

where α_0 and α_1 are the 0-child and 1-child of $\alpha \in \mathcal{N}$. After the computation, we obtain the answer: $C(root)$.

[1] `https://afsa.jp/icgca2023/files/user02/02.pdf`