Counting Paths with Specified Length Based on a Path Enumeration Algorithm

Kazuhiro Kurita

1 Overview

We aim to implement an algorithm for fast counting s-t paths with length constraints. Our implementation is based on a path enumeration algorithm [1]. This algorithm is a standard backtrack-based enumeration algorithm for s-t paths, and its design technique is known as a binary partition or the flashlight approach. In this short abstract, we propose an algorithm that combines this algorithm with a shortest path counting algorithm.

The algorithm by Read and Tarjan divides the set of s-t paths based on edges incident to s $e_1, \ldots, e_k$. This division is repeated recursively to enumerate the s-t paths. In our algorithm, in addition to this division, we also divede based on shortest s-t paths. In the algorithm by Read and Tarjan [1], for each edge $e_1 = \{s, v_1\}, \ldots, e_k = \{s, v_k\}$ incident to s, the set of s-t paths, $\mathcal{P}(G, s, t)$, is divided into $\mathcal{P}(G - \{s\}, v_1, t), \ldots, \mathcal{P}(G - \{s\}, v_k, t)$. In our algorithm, we divide these divisions into sets of shortest paths $\mathcal{P}_s(G - \{s\}, v_i, t)$, and sets of non-shortest paths $\mathcal{P}_n(G - \{s\}, v_i, t)$. Since counting the shortest paths can be done in linear time using a simple dynamic programming approach, counting $\mathcal{P}_s(G - \{s\}, v_i, t)$ can be done in liner time. Furthermore, the set of non-shortest paths can also be recursively divided similarly. Finally, we consider the decision problem of whether $\mathcal{P}_n(G, \{s\})$ is empty. This can be determined in linear time by constructing a directed acyclic graph (DAG) D consisting only of shortest paths from the graph and checking if there are paths that pass through edges not included in D.

In our implementation, if many shortest paths can be found simultaneously, fast counting of paths can be expected. However, preliminary experiments have shown that there are inputs for which this method may not work well. In our preliminary experiments, we found that this method did not perform well for an $n \times n$ grid graph. When we applied the algorithm to this grid graph, the number of paths was enormous compared to the number of paths that could be computed using the shortest path counting algorithm in a single run. We expect that fast counting for long paths using this approach is not straightforward unless many shortest paths can be discovered simultaneously, as it would be equivalent to simple path enumeration. On the other hand, we expect that this method may be powerful for counting paths of almost the same length as the shortest paths.

References

[1] R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5(3):237–252, 1975.