CS260: Project Proposal
Algorithms for Solving Maximum Flow Problem

Lu Yu, Siqing Hou, Zihao Wang

September 20, 2016

1 Introduction

Network flow problem is a kind of classical network optimization problem. We could find its applications in network packets routing, transportation scheduling, bridges destroy with minimum cost etc. Usually, a maximum flow problem needs to be solved in order to optimize practical cost for certain task. Formally, maximum flow problem [4] can be formulated as finding a flow from source node s to target node t with a given directed graph $G = (V, E)$, where each edge e is associated with its capacity $c(e) > 0$. There are two constraints: (i) flow on each edge doesn't exceed $c(e)$; (ii) for every node v non-s and t, incoming flow is equal to outgoing flow. Many pioneering research have provided efficient solutions to maximum flow problem. In this project, we are interested in different aspects of two popular solutions, Edmonds–Karp algorithm [1] and Dinic's Blocking flow algorithm [2]. More specially, we will conduct a series of experiments with different settings to examine their performances in completeness, optimality, time complexity and space complexity. We will further discuss their differences based on the experimental results.

2 Algorithms

The two algorithms we are interested in are both variations of Ford-Fulkerson method [3], which is based on the max-flow min-cut theorem and the idea of augmenting the current flow on an augmenting path until no more augmenting path can be found in the residual graph. The basic Ford-Fulkerson algorithm [5] is only guaranteed to terminate if all capacities are rational. When all the capacities are integral and the maximum flow of the flow network is f^*, it has a time complexity of $O(|E| \cdot |f^*|)$ [5], which is not polynomial. Edmonds-Karp algorithm and Dinic's algorithm use two different strategies of how to find an augmenting path or a blocking flow in the residual graph and thus ensure the completeness and improve the time complexity to polynomial.

Edmonds–Karp algorithm : Compared to the basic Ford-Fulkerson algorithm, Edmonds-Karp algorithm find the shortest augmenting path each iteration. This can be done by a bread-first search on the residual graph. It can compute the maximum flow in a flow network in $O(|V| \cdot |E|^2)$ time [5].

Dinic's algorithm : The introduction of the concepts of the level graph and blocking flow enable Dinic's algorithm to achieve its performance. In each iteration, it will build a level graph from the residue graph via a bread-first search, find the blocking flow with backtracking algorithm and augment the current flow with the blocking flow. The original implementation of this algorithm [2] has time complexity $O(|V|^2 \cdot |E|)$. By using a data structure called dynamic trees in

the procedure of finding a blocking flow, the running time of Dinic's algorithm can be improved to $O(|V| \cdot |E| log|V|)$ [6].

References

[1] Jack Edmonds, Richard M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM)19.2 (1972): 248-264.

[2] Dinitz Y. Dinitz'algorithm: The original version and even's version[M]//Theoretical computer science. Springer Berlin Heidelberg, 2006: 218-240.

[3] Ford L R, Fulkerson D R. Maximal flow through a network[J]. Canadian journal of Mathematics, 1956, 8(3): 399-404.

[4] Harris, T. E.; Ross, F. S. (1955). "Fundamentals of a Method for Evaluating Rail Net Capacities" (PDF).Research Memorandum. Rand Corporation.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press.

[6] Sleator D D, Tarjan R E. A data structure for dynamic trees[J]. Journal of computer and system sciences, 1983, 26(3): 362-391.