CS 270 Project Proposal

Comparing Practical Implementations of Max-Flow

Chirag Sharma: shchirag@berkeley.edu,
Vint Lee: vint@berkeley.edu,
Xuandi Ren: xuandi_ren@berkeley.edu

March 2023

1 Project Implementation Overview

The maximum flow problem, first proposed by [HR55], is not only of great theoretical interest, but also has high practical value in many applications.

Our project is centered on the implementation of 5 different algorithms for solving max flow, with different theoretical worst-case runtimes. All of these algorithms have been covered either in this class or in an introductory algorithms class. The algorithms and runtimes are listed in the table below.
We let n denote the number of vertices in the input graph, m the number of edges, and U the largest edge capacity:

Algorithm	Runtime	Runtime with Capacity Scaling
Ford-Fulkerson [FF56]	$O(mnU)$	$O(m^2 \log U)$
Edmonds-Karp [EK72]	$O(m^2 n)$	–
Dinic's Blocking Flow Algorithm [Din70]	$O(mn^2)$	$O(mn \log U)$
Dinic's Algorithm with Link-Cut Trees	$O(mn \log n)$	–
LP Solver using Simplex Method [Sch99]	–	–

The performance of LP solvers in practice can vary a lot from theoretical worst-case bounds so we will use this only as a benchmark for comparison of practical performance of these algorithms. Note that of the 6 runtimes listed in the table, 3 are strongly polynomial, 2 are weakly polynomial, and 1 is pseudo-polynomial. In particular, this means that our choice of edge capacities in input graphs can significantly affect our performance comparisons.

2 Comparison and Analysis

For each of the implemented algorithms in the table above, we plan to compare practical performance to the listed theoretical guarantees. We can then characterize constant-factor and lower-order runtime contributions and compare them across the different implementations. In particular, we will set up a curriculum of inputs that grow in size in order to determine the best performing algorithm for each regime. Since the practical performance can vary based on graph properties, each curriculum will maintain the same 'class' of graph while scaling n, m (e.g. curriculum of fully connected graphs of varying n).

We aim to also further explore different classes of graphs to see whether any of these algorithms perform significantly better when presented with a specific graph structure. To do this, we will identify 3-4 classes of graphs, each of which has some specified property/structure, and look at the practical performance of these 5 algorithms on each. If we observe any significant runtime speedup/slowdown, we will attempt to analyze the reason for it.

Finally, we also aim to characterize the expected runtime of these 5 algorithms over the choice of input graph by, for example, sampling uniformly at random from $\mathcal{G}_{n,m} := \{G = (V, E) : |V| = n, |E| = m\}$. For algorithms that depend heavily on graph structure, as mentioned above, we expect them to have high variance over the choice of input, and thus a larger gap between expected and worst-case runtime.

3 Further Exploration Directions

If time permits, we may explore the followings regarding max-flow algorithms:

- Look at more recent max-flow algorithms (e.g., the almost-linear time algorithm by Chen et. al [CKL+22])
- Look at randomized algorithms (e.g., [CH95] which runs in $O(nm + n^2 \log^2 n)$ time with high probability)
- Look at heuristics for improving practical performance, (e.g., [BK04] which applies heuristics on top of Dinic's algorithm to improve empirical performance on problem instances found in computer vision)
- Look at dynamic network flow algorithms (e.g. [vdBLS22] which can maintain a $(1 - \varepsilon)$-approximate maximum flow in a dynamic, capacitated graph undergoing edge additions)

References

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[CH95] Joseph Cheriyan and Torben Hagerup. A randomized maximum-flow algorithm. *SIAM Journal on Computing*, 24(2):203–226, 1995.

[CKL+22] Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 612–623. IEEE, 2022.

[Din70] Yefim Dinitz. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 01 1970.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, apr 1972.

[FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[HR55] T. E. Harris and Frcpc Susan D. Ross. Fundamentals of a method for evaluating rail net capacities. 1955.

[Sch99] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

[vdBLS22] Jan van den Brand, Yang P. Liu, and Aaron Sidford. Dynamic maxflow via dynamic interior point methods, 2022.