Efficient Maximum Flow Maintenance on Dynamic Networks

Sergio Greco
University of Calabria, Italy
greco@dimes.unical.it

Cristian Molinaro
University of Calabria, Italy
cmolinaro@dimes.unical.it

Chiara Pulice
University of Maryland, USA
cpulice@umiacs.umd.edu

Ximena Quintana
University of Calabria, Italy
x.quintana@dimes.unical.it

ABSTRACT

While many efficient algorithms for the maximum flow problem have been proposed over the years, they are designed to work with static networks, and thus they need to recompute a new solution from scratch every time an update occurs. Such approaches are impractical in many current application where updates are frequent. To overcome these limitations, this paper proposes efficient *incremental* algorithms for maintaining the maximum flow in dynamic networks.

Keywords

Maximum flow problem; Incremental algorithms

1. INTRODUCTION

The maximum flow problem is nowadays successfully applied in social network analysis [14, 15] for link spam detection [17] and for the identification of web communities [4]. In such applications, flow networks are highly *dynamic*, that is, subject to frequent updates.

Even if many efficient algorithms for the maximum flow problem have been proposed over the years, they are designed to work with *static* networks, and thus they need to recompute a new solution from scratch every time the network is modified.

To overcome these limitations, we propose novel *incremental* algorithms for maintaing the maximum flow in dynamic networks. Incremental algorithms have proved to yield significant benefits in many domains [8, 9, 10, 11, 2]. Experimental results showed that our approach is very efficient and outperforms state-of-the-art algorithms.

2. INCREMENTAL MAXIMUM FLOW COMPUTATION

In this section, we discuss algorithms for the incremental maintenance of the maximum flow (MF) after edge insertions and deletions. We point out that the same approach can be used to handle capacity increases and decreases, as well as insertions and deletions of vertices.

The following two examples illustrate how edge insertions and deletions can be handled.

Example 1 (Edge insertion). *Consider the flow network and the MF in Figure 1a. Each edge label x/y states that y is the edge capacity and x is its current flow. Suppose the edge (c, d) with capacity 15 is added to the network.*

First, we look (in the residual network) for both an augmenting path from s to c and an augmenting path from d to t. Both searches are done in a breadth-first fashion.

Figure 1b shows the paths found in this way: the blue-colored edge from the source to c, whose residual capacity is 10, and the green-colored edge from d to the sink, whose residual capacity is 5. The red edge is the new inserted one and its residual capacity is 15. The flow across the resulting augmenting path is increased by 5, yielding the flow network and the maximum flow in Figure 1c.

Then, as both (c, d) and the path from the source to c have still residual capacity after updating the flow, we look for a new path in the residual network from d to the sink, and finds the one consisting of the green-colored edges in Figure 1c. This allows us to identify a new augmenting path, namely the one consisting of the colored edges in Figure 1c. Along such a path, the flow is increased by 5, yielding the flow network and the maximum flow in Figure 1d.

After that, the path from the source to c is saturated, and since there is no other path from the source to c in the residual network, we stop. Figure 1d shows the updated network with its new maximum flow.

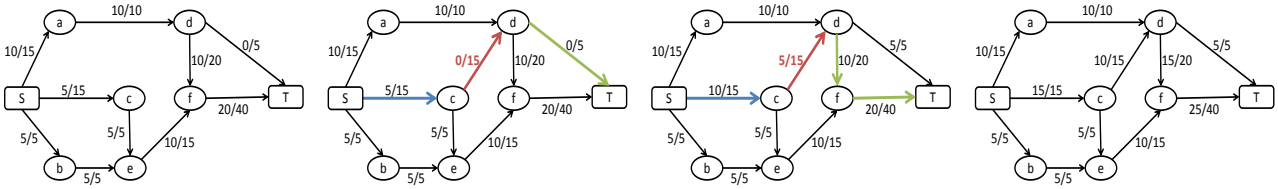
Example 2 (Edge deletion). *Consider the flow network and the MF in Figure 1d. Suppose we delete edge (e, f).*

As a consequence, vertex f has a negative excess, as it is receiving less flow than that being sent. This exceeding flow is sent back to f from the sink, through the only path connecting them—see green-colored edge in Figure 2a. The result is shown in Figure 2b.

Then, we try to push the excess at e toward the sink, looking for a path in the residual network from e to t. By searching in a breadth-first fashion, the blue-colored path in Figure 2b is found; 5 units are pushed along such a path, yielding the flow network in Figure 2c.

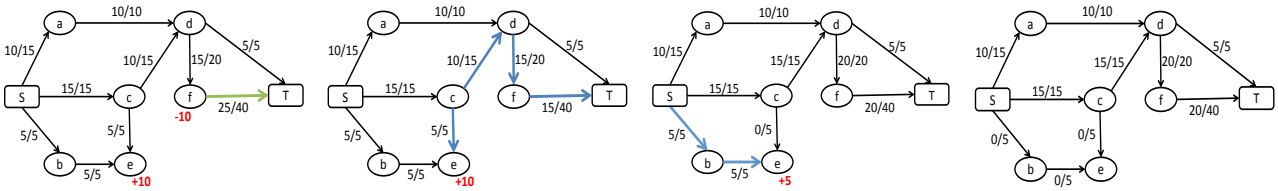
After that, as there is still some excess at e, we look for other paths from e to the sink in the residual network, but none is found. As a result, the exceeding flow at e has to be

WWW 2017 Companion, April 3–7, 2017, Perth, Australia.
ACM 978-1-4503-4914-7/17/04.
http://dx.doi.org/10.1145/3041021.3051152



(a) Flow network and a MF. (b) First iteration. (c) Second iteration. (d) Flow network and a MF.

Figure 1: Incremental maximum flow computation after edge insertion.



(a) First step. (b) Second step. (c) Third step. (d) Flow network and a MF.

Figure 2: Incremental maximum flow computation after edge deletion.

pushed back to the source. This can be done along the blue-colored path highlighted in Figure 2c. The updated network with its new optimal flow are shown in Figure 2d.

Experimental Evaluation. We evaluated our algorithms over a variety of networks taken from the repositories *The Maximum Flow Project Benchmark*[1] and *Computer Vision Datasets*[2], comparing against state-of-the-art algorithms for the maximum flow computation, namely HIPR [3], PAR [5], P2R [6], HPF [12], BK [1], E-BK [16], and E-IBFS [7].

As for edge insertions, our algorithm was the fastest one in all but one dataset, being more efficient than the second-fastest algorithm by at least one order of magnitude in around 75% of the datasets.

Our deletion algorithm was the fastest one in most of the cases, even if handling edge deletions showed to be more expensive than handling edge insertions.

3. REFERENCES

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[2] M. Calautti, S. Greco, and I. Trubitsyna. Detecting decidable classes of finitely ground logic programs with function symbols. In *PPDP*, pages 239–250, 2013.

[3] B. V. Cherkassky and A. V. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

[4] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *KDD*, pages 150–160, 2000.

[5] A. V. Goldberg. The partial augment-relabel algorithm for the maximum flow problem. In *ESA*, pages 466–477, 2008.

[1] `http://www.cs.tau.ac.il/~sagihed/ibfs/benchmark.html`

[2] `http://vision.csd.uwo.ca/data/`

[6] A. V. Goldberg. Two-level push-relabel algorithm for the maximum flow problem. In *AAIM*, pages 212–225, 2009.

[7] A. V. Goldberg, S. Hed, H. Kaplan, P. Kohli, R. E. Tarjan, and R. F. Werneck. Faster and more dynamic maximum flow by incremental breadth-first search. In *ESA*, pages 619–630, 2015.

[8] S. Greco, C. Molinaro, and C. Pulice. Efficient maintenance of all-pairs shortest distances. In *SSDBM*, pages 9:1–9:12, 2016.

[9] S. Greco, C. Molinaro, C. Pulice, and X. Quintana. All-pairs shortest distances maintenance in relational DBMSs. In *ASONAM*, 2016.

[10] S. Greco and F. Parisi. Efficient computation of deterministic extensions for dynamic abstract argumentation frameworks. In *ECAI*, pages 1668–1669, 2016.

[11] S. Greco and F. Parisi. Incremental computation of deterministic extensions for dynamic argumentation frameworks. In *JELIA*, pages 288–304, 2016.

[12] D. S. Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research*, 56(4):992–1009, 2008.

[13] N. Imafuji and M. Kitsuregawa. Finding web communities by maximum flow algorithm using well-assigned edge capacities. *IEICE Transactions*, 87-D(2):407–415, 2004.

[14] C. Kang, S. Kraus, C. Molinaro, F. Spezzano, and V. S. Subrahmanian. Diffusion centrality: A paradigm to maximize spread in social networks. *Artificial Intelligence*, 239:70–96, 2016.

[15] C. Kang, C. Molinaro, S. Kraus, Y. Shavitt, and V. S. Subrahmanian. Diffusion centrality in social networks. In *ASONAM*, pages 558–564, 2012.

[16] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2079–2088, 2007.

[17] H. Saito, M. Toyoda, M. Kitsuregawa, and K. Aihara. A large-scale study of link spam detection by graph algorithms. In *AIRWeb*, 2007.