Intermediary Report

A**New Approach to the Maximum Flow Problem**

Praveen Kumar Shanmugam and Shridharan Chandramouli

University of Utah

October 31, 2014

Problem

Problem Definition

The problem of finding the maximum flow in a given graph is solved using preflow concept of Karzanov. By incorporating the dynamic tree data structure [6] of Sleator and Tarjan the new approach acheives a running time of $\mathcal{O}(nm \log{(n^2/m)})$ on an *n-vertex*, *m-edge* graph.

A minimum cut is a cut of minimum capacity. The max-flow, min-cut theorem of Ford and Fulkerson states that the value of a maximum flow is equal to the capacity of a minimum cut.

Graph G = (V, E) is a directed graph with vertex set V and edge set E. Size of V is denoted by *n* and size of E by *m*. G is a network if it has two distinct distinguished vertices, a *source s* and a *sink t*, and a positive capacity *c(v,w)* on each directed edge (v,w). A flow *f* on G is a real-valued function on vertex pairs satisfying. The value of a flow *f* is the net flow into the sink,

$$|f| = \sum_{v \in V} f(v,t) \tag{1}$$

A maximum flow is a flow of maximum value. A *cut* S, $\bar{S}$ is a partition of the vertex set $(S \cup \bar{S} = V, S \cap \bar{S} = 0)$ with $s \in S$ and $t \in \bar{S}$. The capacity of the cut is

$$c(S,\bar{S}) = \sum_{v \in S, w \in \bar{S}} c(v,w) \tag{2}$$

The flow across the cut is

$$f(S,\bar{S}) = \sum_{v \in S, w \in \bar{S}} f(v,w) = |f| \tag{3}$$

Main Result

The new approach makes the algorithm as fast as any known method for any graph density and faster on graphs of moderate destiny. And the algorithm is efficient in distributed and parallel implementations. The parallel implementation running in $\mathcal{O}(n^2 \;\log n)$ time and uses only $\mathcal{O}(m)$ space with a time bound $\mathcal{O}(n^3)$, and achieves a $\mathcal{O}(nm\;\log{(n^2/m)})$ run time in a sequential implementation.

Importance of Result

TODO:Explain why this result is important (provide some background with references)
The below table shows the algorithms running time complexity which performs better in either in dense

graphs or in sparse graphs. But there is no clear winner which performs better in both the cases. This is where our approach surpasses all. By using the dynamic tree data structure [6] of Sleator and Tarjan it achieves a $\mathcal{O}(nm\ \log{(n^2/m)})$ run time in a sequential implementation, and $\mathcal{O}(n^2\ \log(n))$ in a parallel version of implementation.

S. No	Date	Discoverer	Running Time
1	1956	Ford and Fulkerson	-
2	1969	Edmonds and Karp [1]	$\mathcal{O}(nm^2)$
3	1970	Dinic	$\mathcal{O}(n^2m)$
4	1974	Karzanov	$\mathcal{O}(n^3)$
5	1977	Cherkasky	$\mathcal{O}(n^2\sqrt(m))$
6	1978	Malhotra, Pramodh Kumar, and Maheshwari	$\mathcal{O}(n^3)$
7	1978	Galil	$\mathcal{O}(n^{5/3}m^{2/3})$
8	1978	Galil and Naamad [3]; Shiloach [4]	$\mathcal{O}(nm\ (\log n)^2)$
9	1980	Sleator and Tarjan [6]	$\mathcal{O}(nm\ \log(n))$
10	1982	Shiloach and Vishkin [5]	$\mathcal{O}(n^3)$
11	1983	Gabow [2]	$\mathcal{O}(nm\ \log(N))$
12	1984	Tarjan	$\mathcal{O}(n^3)$

Impact

TODO:Explain what impact (if any) this result has had (or might have)
The approach gives a generalized best solution for either density or a sparse graph dealt with. This simplifies the usage of the algorithm for both type of graph inputs.

Outline

Outline, in a page, your best understanding of how the paper does what it does, focusing on key techniques. For example, it might be something like

- previous papers did X
- they hit a bottleneck Y
- This paper uses key idea Z to get improvement A1, and then key idea Z' to get improvement A2, and so on.

References

[1] EDMONDS, J., AND KARP, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM 19*, 2 (Apr. 1972), 248–264.

[2] GABOW, H. N. Scaling algorithms for network problems. *J. Comput. Syst. Sci. 31*, 2 (Sept. 1985), 148–168.

[3] GALIL, Z., AND NAAMAD, A. An O(EVIog2V) algorithm for the maximal flow problem. *Journal of Computer and System Sciences 21*, 2 (Oct. 1980), 203–217.

[4] SHILOACH, Y. An o(n . i log$\hat{2}$ i) maximum-flow algorithm. Tech. rep., Stanford, CA, USA, 1978.

[5] SHILOACH, Y., AND VISHKIN, U. An o(n2 log n) parallel max-flow algorithm. *J. Algorithms 3*, 2 (Feb. 1982), 128–146.

[6] SLEATOR, D. D., AND TARJAN, R. E. A data structure for dynamic trees. *J. Comput. Syst. Sci. 26*, 3 (June 1983), 362–391.