# A Survey on Max-flow Problem

BY XIHAN LI 312, LIANGZU PENG 3130000858

*June 21, 2016*

**Abstract**

In this survey we had some discussions to solve the problem for calculating the maximal flow of an directed edge-weighted graph. We introduced the maximal flow problem itself first, and then discussed various canonical algorithms. Some necessary comparisons of those algorithms were done and some simple analyses were introduced. Finally, we showed the results by implementing some of those algorithms.

## Definition [2][4][8]

Let $G=(V,E)$ be a network with $s,t\in G$ being the source and the sink of $G$, respectively. The ***capacity*** of an edge is a mapping $c: E\to\mathbb{R}^+$, denoted by $c(u,v)$. A ***flow*** in $G$ is a mapping $f: E\to\mathbb{R}^+$, denoted by $f(u,v)$, that satisfies:

1.For all $u,v\in V, 0\leqslant f(u,v)\leqslant c(u,v)$ (capacity constraint)

2.For all $u\in V-\{s,t\}, \sum_{v\in V}f(v,u)=\sum_{v\in V}f(u,v)$ (flow conservation)

The ***value*** $|f|$ of a flow f is defined as $|f|=\sum_{v\in V}f(s,v)-\sum_{v\in V}f(v,s)$. The ***maximum flow problem***[1] is to maximize $|f|$, that is, to route as much flow as possible from $s$ to $t$.

The ***residual capacity*** $c_f(u,v)$ is defined as:

$$c_f(u,v)=\begin{cases} c(u,v)-f(u,v) & \text{if } (u,v)\in E,\\ f(v,u) & \text{if } (v,u)\in E,\\ 0 & \text{otherwise}\end{cases}$$

The ***residual network*** of $G$ induced by $f$ is $G_f=(V,E_f)$, where $E_f=\{(u,v)\in V\times V: c_f(u,v)>0\}$.

An ***argumenting path*** p is a simple path from s to t in the residual network $G_f$.

***preflow*** is a function: $f: V\times V\to\mathbb{R}$ that satisfies the capacity constraint and the following relaxation of flow conservation:

$$\sum_{v\in V}f(v,u)-\sum_{v\in V}f(u,v)\geqslant 0$$

for all vertices $u\in V-\{s\}$, the ***excess flow*** of vertice $u$ is defined as: $e(u)=\sum_{v\in V}f(v,u)-\sum_{v\in V}f(u,v)$

***labeling*** h assigns a non-negative integer label $h(v)$ to all $v$ in $V$.

***preflow*** $f$ and ***labeling*** l are ***compatible*** if:

1.$h(s)=n, h(t)=0$

2.For all edges $(v,w)$ in $G_f, h(v)\leqslant h(w)+1$

## Introduction

It has been a couple of decades since the maximum flow problem was first formulated in 1954 by T.E.HARRIS and F.S.ROSS[7]. Its description is as follows:

> "Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other."

1. https://en.wikipedia.org/wiki/Maximum_flow_problem

Later on, in 1956, the Ford-Fulkerson method[3] was created. This method depends on three important ideas, which later became the basis of many other maximum flow algorithms: *residual networks, argumenting paths,* and *cuts*.

Over the years, more and more algorithms are introduced[1]. The basic methods for the maximum flow problem include the network simplex method of Dantzig, the augmenting path method of Ford and Folkerson, and bloking flow method of Dinitz, and the push-relabel method of Goldberg and Tarjan.

The below table serves as a record of the research on maximum flow problem:[4]

| # | Discoverer | Running Time | Date |
|---|---|---|---|
| 1 | Ford & Fulkerson | $O(E \max \|f\|)$ | 1956 |
| 2 | Edmonds & Karp | $O(\mathrm{VE}^2)$ | 1969 |
| 3 | Dinic | $O(V^2E)$ | 1970 |
| 4 | Karzanov | $O(V^3)$ | 1974 |
| 5 | Cherkasky | $O(V^2E^{1/2})$ | 1977 |
| 6 | Malhotra & Pramodh kumar & Maheshwari | $O(V^3)$ | 1978 |
| 7 | Galil | $O(V^{5/3}E^{2/3})$ | 1978 |
| 8 | Galil & Naamad & Shiloach | $O(VE\,(\log V)^2)$ | 1978 |
| 9 | Sleator & Tarjan | $O(VE \log V)$ | 1980 |
| 10 | Shiloach & Vishkin | $O(V^3)$ | 1982 |
| 11 | Gabow | $O(VE \log U)$ | 1983 |
| 12 | Tarjan | $O(V^3)$ | 1984 |
| 13 | Goldberg | $O(V^3)$ | 1985 |
| 14 | Goldberg & Tarjan | $O(VE \log(V^2/E))$ | 1986 |
| 15 | Ahuja & Orlin | $O(VE + V^2 \log U)$ | 1986 |

In this survey, we will study only two of them, which both are basic but classic: Edmonds & Karp algorithms and push-relabel algorithms.

# Algorithms

The Edmonds & karp algorithm is based on the Ford-fulkerson method:

**Algorithm Ford-fulkerson:**

initialize flow f to 0
**while** there exists an augmenting path p in the residual network $G_f$
    augment flow f along p
**return** p

Edmonds & karp algorithm uses ***breadth-first search*** to find the augmenting path in $G_f$. The length of the augmenting path will be non-decreasing with the increase of flow. There always exists a critical edge $(u, v)$, where $C_f(u, v)$ - the flow of the augmenting path, which disappears afterwards. Each edge can become critical at most $O(|V|)$ times because the length of the augmenting path can increase at most $O(|V|)$ time. The total number of flow augmentations is $O(|V||E|)$, hence the total time complexity is $O(|V||E|^2)$.[2]

The push-relabel algorithm was designed by Goldberg and Tarjan[4], implemented by using the concept of a ***preflow***, which was originally designed by Karzanov[6]. The below is the algorithm itself:

2. borrowed from Chin's ppt.

**Algorithm Push-Relabel**

Start with ***labeling***: $h(s)=n, h(t)=0, h(v)=0$, for all other $v$
Start with ***preflow*** $f$: $f(e)=c(e)$ for $e=(s,v), f(e)=0$, for all other edges $e$

**While** there is a node (other than $t$) with positive ***excess***:
Pick a node v with excess($v$) > 0
If there is an edge $(v,w)$ in $E_f$ such that push$(v,w)$ can be appied
Push$(v,w)$
Else
Relabel(v)

Push$(v,w)$: Applies if excess($v$) > 0, h($w$) < h($v$), $(v,w)$ in $E_f$
q = min (excess($v$), $c_f(v,w)$)
Add q = $f(v,w)$
Relabel($v$): Applies if excess($v$) > 0, for all w s.t $(v,w)$ in $E_f$, h($w$) >= h($v$)
Increase h($v$) by 1

The correctness of this algorithm is not straightforward. There are three steps we need to prove.
First, we can prove ***preflow*** $f$ and ***labeling*** h are always ***compatible*** over the push-relabel algorithm by induction:
Initially, ***compatible*** of course, as $G_f$ has no $(s,v)$ edges.
Suppose f and h are ***compitible*** at time t. At time t + 1:
- Relabel: Labels increase only if no downward edges in $G_f$
- Push: Edges in $G_f$ may be reversed. If so, as we push from high to low h a downwards edge will become an upwards edge

Then, we should realize that:

*If preflow f and labeling h are compatible, then there is no s-t path in $G_f$.*

the proof is as follows:
**Proof:** Suppose there is an $s-t$ path$(s, v_1, v_2, \ldots\ldots, v_{k-1}, t)$ in $G_f$, due to the ***compatibility***,

$$h(v_1) \geqslant h(s) - 1 = n - 1$$
$$h(v_2) \geqslant h(v_1) - 1 \geqslant n - 2$$
$$\ldots$$
$$h(t) = h(v_{k-1}) - 1 \geqslant n - k > 0 (\text{as } k < n)$$

This is a contradiction because of the ***compatibility***$(h(t)=0)$.

We should also realize that(because a flow is surely a preflow):

*if flow f and labeling h are compatible, then f is max flow*.(due to no $s-t$ path)

Finally, we will show:

*a preflow "becomes" a flow on termination of the algorithm.*

It should be noted that, when this algorithm terminates, there is no node $v$ with excess$(v)>0$, and hence, $f$ is a flow.
Consequently, Push-Relabel algorithm works correctly on maximum flow problem.

## Key Results

(finished by XIHAN LI).

## Conclusion

Though maximum flow algorithms have a long history, revolutionary progress is still being made[5].

Maximum flow problem has already become one of the cornerstone problems in combinatorial optimization, and it has many practical applications and nontrivial reductions, for example in network reliability[3], data mining, distributed computing, bipartite matching, and image segmentation.

Studying maximum flow algorithm not only strengthens our ability of modeling and abstracting the real-world problems, but also tells us that "the maximum flow problem is far from being completely understood, and new and improved algorithms continue to be discoverd[5]". We may do some research on maximum flow problem or meet it in the real-world someday!

## Bibliography


**[1]** Boris V Cherkassky and Andrew V Goldberg. On implementing the push—relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997.

**[2]** Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

**[3]** Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

**[4]** Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

**[5]** Andrew V Goldberg and Robert E Tarjan. Efficient maximum flow algorithms. *Communications of the ACM*, 57(8):82–89, 2014.

**[6]** Alexander V Karzanov. Determination of maximal flow in a network by method of preflows. *Doklady Akademii Nauk SSSR*, 215(1):49–52, 1974.

**[7]** Alexander Schrijver. On the history of the transportation and maximum flow problems. *Mathematical Programming*, 91(3):437–445, 2002.

**[8]** Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-Wesley Professional, 4th edition, 2011.


3. http://www.cse.unr.edu/~mgunes/papers/JNCA13.pdf, a survey of network flow applications