Maximum Flow Problem in Graph

Ford-Fulkerson Algorithm / Method

Ford–Fulkerson Algorithm (FFA) is a greedy algorithm that computes the maximum flow in a flow network. It is sometimes called a "method" instead of an "algorithm" because it encompasses several implementations with differing running times. A path with available capacity is called an *augmenting path*.

Important concepts in FFA

Residual networks

Given a flow network G and a flow f, the residual network G_f consists of edges with capacities that can admit an amount of additional flow equal to the edge's capacity minus the flow on that edge.

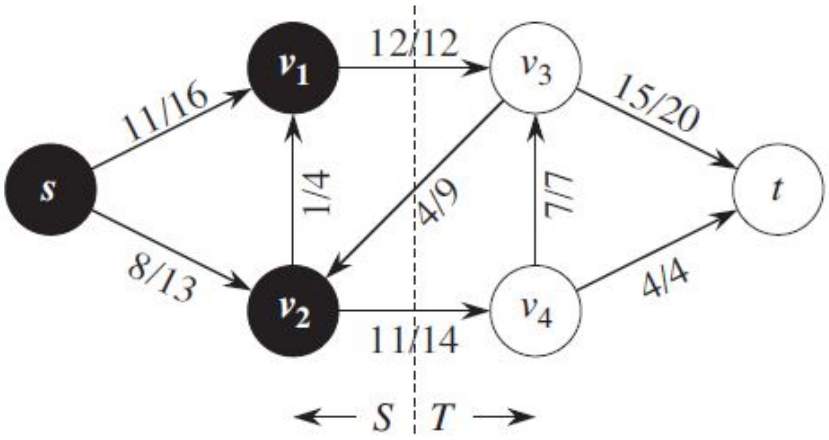
Augmenting paths

Given a flow network $G = (V, E)$ and a flow f , an augmenting path p is a simple path from s to t in the residual network G_f. We call the maximum amount by which we can increase the flow on each edge in an augmenting path p the residual capacity of p, given by $c_f(p) = min\{c_f(u, v) : (u, v)\ is\ on\ p\}$

Cuts of flow network

How do we know that when the algorithm terminates, we have actually found a maximum flow? The *max-flow min-cut theorem*, tells us that a flow is maximum if and only if its residual network contains no augmenting path.

A cut (S, T) of flow network $G = (V, E)$ is a partition of V into S and T = V - S such that $s \in S$ and $t \in T$.



A cut (S, T) in the flow network of above figure, where $S = \{s, v_1, v_2\}$ and $T = \{v_3, v_4, t\}$ The vertices in S are black, and the vertices in T are white. The net flow across (S, T) is $f(S, T) = 19$, and the capacity is $c(S, T) = 26$.

A *minimum cut* of a network is a cut whose capacity is minimum over all cuts of the network.

Max-flow min-cut theorem

If f is a flow in a flow network $G = (V, E)$ with source s and sink t , then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$ for some cut (S, T) of G.

Algorithm

Let $G(V, E)$ be a graph, and for each edge from u to v, let $c(u, v)$ be the capacity and $f(u, v)$ be the flow. We want to find the maximum flow from the source s to the sink t. After every step in the algorithm the following is maintained:

Capacity constraints	$\forall (u, v) \in E : \ f(u, v) \leq c(u, v)$	The flow along an edge can not exceed its capacity
Skew symmetry	$\forall (u, v) \in E : \ f(u, v) = -f(v, u)$	The net flow from u to v must be the opposite of the net flow from v to u

Flow conservation	$\forall u \in V : u \neq s \text{ and } u \neq t \Rightarrow \sum_{w \in V} f(u,w) = 0$	The net flow to a node is zero, except for the source, which "produces" flow, and the sink, which "consumes" flow
Value(f)	$\sum_{(s,u) \in E} f(s,u) = \sum_{(v,t) \in E} f(v,t)$	The flow leaving from s must be equal to the flow arriving at t.

After each round in the algorithm. We define the residual network $G_f(V, E_f)$ to be the network with capacity $c_f(u,v) = c(u,v) - f(u,v)$ and no flow. Notice that it can happen that a flow from v to u is allowed in the residual network, though disallowed in the original network: if $f(u,v) > 0$ and $c(v,u) = 0$ then $c_f(v,u) = -f(v,u) \Rightarrow f(u,v) > 0$.

Algorithm Ford–Fulkerson
Inputs Given a Network $G = (V, E)$ with flow capacity c, a source node s, and a sink node t
Output Compute a flow f from s to t of maximum value

1. $f(u,v) \leftarrow 0$ for all edges (u,v)
2. While there is a path p from s to t in G_f, such that $c_f(u,v) > 0$ for all edges $(u,v) \in p$:
    1. Find $c_f(p) = \min\{c_f(u,v) : (u,v) \in p\}$
    2. For each edge $(u,v) \in p$
        1. $f(u,v) \leftarrow f(u,v) + c_f(p)$ (Send flow along the path)
        2. $f(v,u) \leftarrow f(v,u) - c_f(p)$ (The flow might be "returned" later)

The path in step 2 can be found with a breadth-first search (BFS) or a depth-first search in $G_f(V, E_f)$. If BFS is used, the algorithm is called Edmonds–Karp.

Complexity

By adding the flow augmenting path to the flow already established in the graph, the maximum flow will be reached when no more flow augmenting paths can be found in the graph. However, there is no certainty that this situation will ever be reached, so the best that can be guaranteed is that the answer will be correct if the algorithm terminates. The situation where this algorithm runs forever, only occurs with irrational flow values. When the capacities are integers, the runtime of Ford–Fulkerson is bounded by $O(Ef)$, where E is the number of edges in the graph and f is the maximum flow in the graph. This is because each augmenting path can be found in $O(E)$ time and increases the flow by an integer amount of at least 1, with the upper bound f.

A variation of the Ford–Fulkerson algorithm with guaranteed termination and a runtime independent of the maximum flow value is the Edmonds–Karp algorithm, which runs in $O(VE^2)$ time.