SAPIENZA
UNIVERSITÀ DI ROMA

Achieving Max Flow in Strongly Polynomial Time for Sparse Networks: Beyond the Edmonds-Karp Algorithm

Faculty of Information Engineering, Computer Science and Statistics
Bachelor's Degree in Computer Science

Armando Coppola
ID number 2003964

Advisor
Prof. Paul Joseph Wollan

Academic Year 2023/2024

Achieving Max Flow in Strongly Polynomial Time for Sparse Networks:Beyond the Edmonds-Karp Algorithm
Bachelor's Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: ArmandoCoppola24@gmail.com

DA DEDICARE.

Contents

Chapter 1

Preliminary notions

1.1 Network and flow

Before starting, we need to establish some essential preliminary notions. In particular, we need to define what a network is and what it is composed of.

Definition 1.1 (Network)**.** A network is a structure composed of a graph G such that $G = (N, E)$ such that:

- N = the set of nodes
- E = the set of edges such that $(i, j) \in E \implies i, j \in N$
- $n = |N|$
- $m = |E|$

and a function $u : E \to \mathbb{N}^+ \cup \{+\infty\}$ which denotes the capacity of each edge.

$$u(i, j) = \text{capacity of the edge } (i, j)$$

we will denote the capacity $u(i, j)$ *below with the abbreviation* u_{ij}

In each network exists two special nodes, s *the source* and t *the sink*. The network aims to send a certain flow from the source to the sink.

Definition 1.2 (U_{min}, U_{max})**.** In each Network we define:

- U_{min}: the smallest non zero capacity associated to an edge:

$$U_{min} = u_{ij} | (i, j) \in E \ \wedge \ u_{ij} > 0 \ \wedge \ \nexists (k, l) \in E : 0 < u_{kl} < u_{ij}$$

- U_{max}: the largest finite capacity

$$U_{max} = u_{ij} | (i, j) \in E \wedge u_{ij} \neq +\infty \ \wedge$$
$$\nexists (k, l) \in E : u_{ij} < u_{jl} < +\infty$$

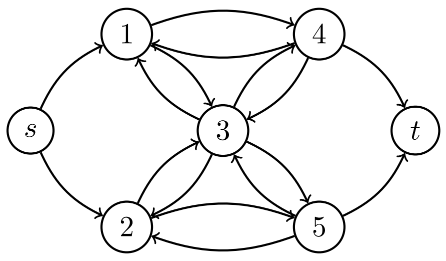


Figure 1.1. A classic example of a network

Moreover, we divide the edge into two categories:
External Arcs $:= \{(x,y) | (x,y) \in E \wedge (x = s \vee y = t)\}$
Internal Arcs $:= \{(x,y) | (x,y) \in E \wedge x \neq s \wedge y \neq t\}$ i.e. $E \setminus External\ edges$

For our simplicity, we assume that for each internal edge $(i,j) \in E$ exists the edge $(j,i) \in E$.
The same thing is true for any internal node for which there always exists an edge that link it with s and t, even if it has zero capacity.

$$\forall i \in N \implies \{(s,i),(i,t)\} \subseteq E$$

Observation 1.1. *Node s has no incoming arcs just as node t has no outgoing arcs.*

Definition 1.3 (Flow)**.** We define the flow as the function $f : E \to \mathbb{R}_+ \cup \{0\}$ which satisfies the **flow conservation role**:

$$\sum_{j:(i,j)\in E} f_{ij} - \sum_{j:(j,i)\in E} f_{ji} = 0 \quad \forall i \in N \setminus \{s,t\}$$

We call a flow *feasible* if it respects the **capacity constraint**:

$$\forall (i,j) \in E,\ f_{ij} \leq u_{ij}$$

The value of a flow is given by the sum of all the outgoing edges of s (or by the sum of all the incoming edges of t; it is the same)

Definition 1.4 (Residual capacity)**.** The residual capacity of an edge (i,j) means the amount of flow we can route in this edge before we saturate it.

$$r_{ij} = u_{ij} + f_{ji} - f_{ij}$$

When we talk about residual capacity according to different flows we could also use the notation:

$$u_f(i,j)$$

that means the residual capacity of the edge (i,j) which has routed the flow f.

We will often talk later about the residual function or the array of residual capacities, in fact we are referring to any function or structure that associates each arc with its residual capacity.

Definition 1.5 (Residual Graph)**.** Given a network G and a flow f, we can define a residual graph as follows

$$G[r] := (N(\mathcal{N}), \{(i,j)|(i,j) \in E(\mathcal{N}) \wedge r_{ij} > 0\})$$

The notation $G[r]$ refers to a graph designed from the residual capacity function r. We will refer to the residual graph also using the notation G_f that underlines the representation of the original network under the effect of the routed flow f

Definition 1.6 (s-t Cut)**.** Given a network G we define an *s-t* cut on G as a partition into two subsets (T, S) such that:

1. $s \in S$
2. $t \in T$
3. $S \cap T = \varnothing$
4. $S \cup T = N$

The *cutting capacity* is defined as:

$$u(S,T) = \sum_{i \in S \wedge j \in T} u_{ij}$$

the *residual of a cut* is defined as:

$$r(S,T) = \sum_{i \in S \wedge j \in T} r_{ij}$$

Lemma 1.1 (Max residual flow min residual cut)**.** *Given a residual graph $G[r]$ and a cut (S,T) then $r(S,T)$ represents the upper bound of the flow from $s \to t$. In particular, the maximum increase in flow with respect to r is the smallest residual capacity of an s-t cut.*

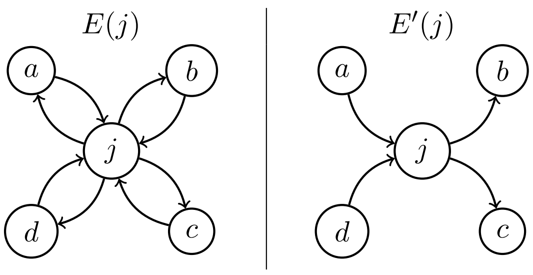
Proof. Omitted.

The lemma states that the problem of finding the maximum flow on a network is **dual** to that of finding a minimum capacity cut on the same network since this will represent the bottleneck that acts as an upper bound to the increase in flow.

Definition 1.7. (Anti-symmetric subset) Let $E(j)$ be the set of edges incident to a node j, we define the *Anti-symmetric* subset of j as:

$$E'(j) := \{(x,y)|(x,y) \in E(j) \wedge (x,y) \in E'(j) \iff (y,x) \notin E'(j)\}$$

Example:



Lemma 1.2 (Anti-symmetriy lemma)**.** *Given $E'(j)$ an anti-symmetric subset of $E(j)$ and a flow f on G with $r = r[f]$ then is true that:*

$$\sum_{(i,j)\in E'(j)} r_{ij} - \sum_{(j,i)\in E'(j)} r_{ji} = \sum_{(i,j)\in E'(j)} u_{ij} - \sum_{(j,i)\in E'(j)} u_{ji}$$

Proof.

$$\sum_{(i,j)\in E'(j)} r_{ij} - \sum_{(j,i)\in E'(j)} r_{ji} - \sum_{(i,j)\in E'(j)} u_{ij} + \sum_{(j,i)\in E'(j)} u_{ji} = 0 \implies$$

$$\sum_{(i,j)\in E'(j)} (u_{ij} - r_{ij}) + \sum_{(j,i)\in E'(j)} (u_{ji} - r_{ji}) = 0$$

since $r_{ij} = u_{ij} - f_{ji} + f_{ij} \implies u_{ij} - r_{ij} = f_{ji} - f_{ij}$

$$\sum_{(i,j)\in E'(j)} (f_{ji} - f_{ij}) + \sum_{(j,i)\in E'(j)} (f_{ij} - f_{ji}) = \sum_{(i,j)\in E(j)} (f_{ji} - f_{ij}) = 0$$

so we deduce the conservation flow constraint. □ □

1.2 Flow decomposition and transferring

Definition 1.8 (Flow decompositon)**.** Given f an s-t flow on a Network $\mathcal{N}$, we define a *flow-decomposition*, as a collection of s-t directed path

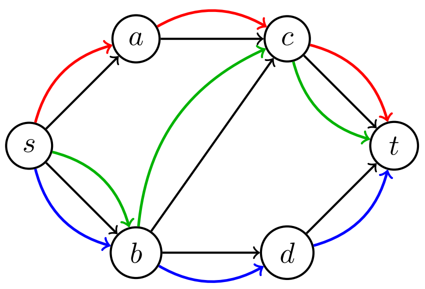
$$P_1, ..., P_k \quad where\ k < m$$

To each path P_i corresponds a value $\phi_i \in \mathbb{N}^+ | \phi_i > 0$ that is the value of the path flow.

In a flow decomposition the following rules must be respected:

1. $\forall P_i, P_j,\ |P_i \cap P_j| \neq |P_i| \land |P_i \cap P_j| \neq |P_i|$ So each path in the decomposition must differ for at least one edge
2. $val(f) = \sum_{i=1}^{k} \phi_i$

An intuitive observation is that the maximum number of decompositions of any flow is m.



Example of a decomposed flow

Once we establish what decomposing a flow means, we can talk about capacity transfer

Definition 1.9 (Tranfer)**.** Given an edge $(i, j) \in E$ and a *path* P $i \to j$ with $|P| \geq 2$, **to transfer** δ unity of capacity from P to (i, j) means subtracting δ unity of residual capacity from each edge in P and incrementing the (i, j) residual capacity of the same δ unity

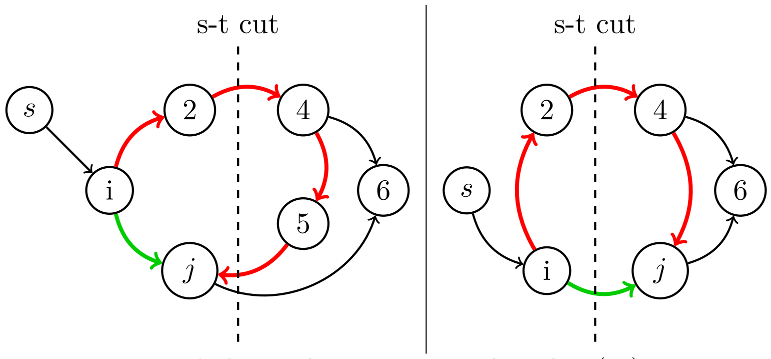
Lemma 1.3 (Capacity tranfer lemma)**.** *Let P be path in G from node i to node j and let (S, T) be an s-t cut. If we transfer delta capacity from the path P to the edge (i, j) and r and r' are respectively the residual capacity of the network before and after the transfer then is true that:*

$$r'(S, T) \leq r(S, T)$$

Proof. The proof is trivial if $i, j \in S \vee i, j \in T$ since $u'(P) \leq u(P) \implies u'(S, T) \leq u(S, T)$.
Otherwise if $i \in S \wedge j \in T$, if we consider $(l, k) \in P$ such that $l \in S \wedge k \in T$ we estimate

$$u'(S, T) - u(S, T) \leq (u'_{kl} + u'_{ij}) - (u_{kl} + u_{ij}) = -\delta + \delta = 0$$

$\square$



in red the path P, in green the edge (i,j)

At the end of this consideration, we can deduce that transferring capacity from a path to an edge doesn't increase the maximum routable flow in a network

1.3 Distance based

We usually think about graphs composed of nodes linked by edges and measure the distance between two nodes i and j as the sum of the edges on the shortest path that brings from i to j. That is true just because we don't specify the length of an edge then we assume that it is one. Instead, we can specify the length of each edge and still divide the nodes by labels, i.e. by the distance from a specific node. But in doing this we have to pay attention to some rules that allow us to achieve our goal. First of all we need to establish what a valid distance labeling is:

Definition 1.10 (Valid distance labeling)**.** Let N be a Network, f a feasible flow on N and l a function that takes as input an edge in G and returns its *length*.
The **distance function** $d : N(G) \to \mathbb{N}$ is said **valid** with respect to the residual graph $G[r]$ if it satisfies the following properties

1. $d(t) = 0$
2. $d(i) \leq d(j) + l((i, j))$

Observation 1.2 (Valid distance label property)**.** *A valid distance label, d, preserves the following properties:*

1. *$d(i)$ represents the lower bound of the length of the shortest path from $i \to t$ in the residual graph*
2. *$d(s) \geq n \implies \nexists\ p$ path $\in G[r] \mid p = s \to t$*

Another point of view of the second property that a valid distance label has to respect $(d(i) \leq d(j) + l((i, j)))$ is that:

$$\neg(d(v) > d(w) + l(v, w))$$

This means that can not exist a node i that is more distant from t than any node j adjacent to i, plus the length of the edge (i, j).

Definition 1.11 (Admissible graph)**.** Let G be a Network with a feasible flow f, a valid distance label $d : N(G_f) \to \mathbb{N}$ and a length function l. A *residual arc* is called **Admissible arc** if it satisfies:

$$d(v) = d(w) + l(v, w) \quad \forall (v, w) \in E(G(f))$$

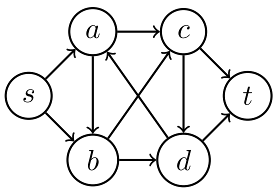
i.e.

$$d(v) > d(w) \vee (d(v) = d(w) \wedge l(v, w) = 0)$$

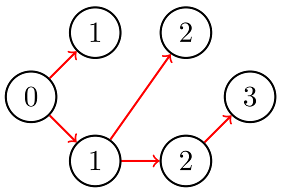
We will represent the admissible graph with the notation $A(f, l, d)$. The admissible graph is the graph formed to all the edges that are admissible concerning the distance d and the length l in the residual graph given by the flow f

Observation 1.3. *Let G_f be a residual graph, let B_s be an arborescence given by a BFS on the graph G_f from node s and let A l'admissible graph of G_f then*

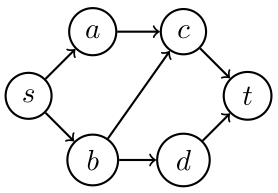
$$E(B_s) \subsetneq E(A)$$



grafo originale



arborescenza



Admissible graph

Given the distance label definition, we can recall the notions about s-t cut to define the **canonical cut**

Definition 1.12 (Canonical cut)**.** Given a network $\mathcal{N}$ and a distance label d on $\mathcal{N}$, a canonical cut is defined by a partition made as follows

$$(S_k, T_k) = (S_k := \{v \in V(\mathcal{N}) | d(v) \geq k\},\ T_k := V(\mathcal{N}) \setminus S_k)$$

1.4 Max flow

To find the maximum flow in the network, we can use the **Edmonds-Karp algorithm**.

This algorithm finds the shortest path from the source to the sink P and augments the flow on the edges of the path by the value x s.t.

$$x = \min_{\forall (i,j) \in P} r_{ij}$$

In this way at each increment at least one edge is deleted from the residual graph and in at most $O(nm)$ increments the algorithm terminates. Since we need $O(n+m)$ time to use the BFS to find the shortest s-t path and other $O(n)$ time tu augments flow in this path, Edmonds-Karp algorithm take $O(nm^2)$ time to find a maximum flow in any network.

Up to here, all notations that we need to recognize a network and its properties were given. The Edmonds-Karp algorithm represents the first step in a series of improvements that will lead us to find the max flow in $O(nm)$. From here on, each algorithm will bring a modification of the previous one while preserving the original intuition. The last algorithm shows how to reach the desired cost even for sparse graphs.